
Spine Toolbox Documentation

Release 0.7.0-final.0

Spine project consortium

Aug 25, 2023

CONTENTS:

1	What's New?	3
2	Getting Started	19
3	How to Set up SpineOpt.jl	33
4	Setting up Consoles and External Tools	37
5	Main Window	47
6	Project Items	51
7	Links	55
8	Tool Specification Editor	59
9	Executing Projects	63
10	Settings	67
11	Welcome to Spine Database Editor's User Guide!	77
12	Plotting	119
13	Parameter Value Editor	123
14	Spine Metadata Description	133
15	Importing and Exporting Data	135
16	Spine Engine Server	151
17	Terminology	155
18	Contribution Guide	157
19	Developer Documentation	163
20	API Reference	175
21	Indices and tables	643
	Bibliography	645

Python Module Index	647
Index	651

Spine Toolbox is an application, which provides means to define, manage, and execute complex data processing and computation tasks, such as energy system models.

If you are new to Spine Toolbox, *Getting Started* section is a good place to start. If you want to run **SpineOpt.jl** using Spine Toolbox, *How to Set up SpineOpt.jl* provides step-by-step instructions on how to get started. For information on how to set up Python, Julia, or Gams for Spine Toolbox, see *Setting up Consoles and External Tools*. Please see *Settings* chapter for information on user customizable Spine Toolbox settings. If you need help in understanding the terms we use throughout the app and this User Guide, please check the *Terminology* section. If you want to contribute to this project, please see the *Contribution Guide*. The last section contains the complete code reference of Spine Toolbox.

WHAT'S NEW?

Here's the Changelog for Spine Toolbox.

```
# Changelog
All **notable** changes to this project are documented here.

The format is based on [Keep a Changelog](http://keepachangelog.com/en/1.0.0/)

## [0.7.0] - 2023-08-25

### Added
- Support for version 11 Spine Toolbox projects.
- Executable Tool Specifications can be used to run any (shell) command. This enhancement duplicates the functionality of Gimlet project items and makes them obsolete.
- There is now an option to select if new scenarios or tools are automatically used for filtering in Link properties.
- The new "Filer validation" button in Link properties allows forcing at least one ↵ scenario or tool filter to be checked.
- Python 3.11 support.
- PySide6 support. The app has been ported from PySide2 to PySide6.
- In addition to dragging and dropping, it is now possible to copy alternatives from Alternative tree and paste them to the ↵ scenario tree in Database editor.
- Scenarios can now be copied and pasted between databases in Database editor.
- Scenarios can now be duplicated in Database editor.
- Tool project item's Python and Julia consoles can now be killed from the right-click ↵ context menu.
  A killed console can be restored by restarting it.
- A new option "Kill consoles at the end of execution" has been added to Tool properties, that kills Python and Julia console processes after execution saving some memory and ↵ computing resources.
- A new option "Log process output to a file" has been added to Tool properties, that logs all the 'console' output of the Tool to a file in the logs subfolder.
- You can now open a 'Detached' Jupyter Console on any kernel you have installed in your ↵ system from the *Consoles* main window menu.
- "Make Julia Kernel" and "Make Python Kernel" buttons in Settings->Tools page. Clicking ↵ them creates a new Julia or Python kernel based on selected Julia/Python executable on the same page if ↵
```

(continues on next page)

(continued from previous page)

↪ the kernel does not exist.
 If the kernel already exists, it is selected automatically.

- ``project.json`` now has an experimental option ["project"]["settings"]["enable_execute_all"] which disables the Execute Project button when set to ``false``. The option is currently not settable in the UI.

Changed

- The console settings of Python tools as well as the command and shell settings of executable tools are now treated as project-specific data and stored in ``<project root>/spinetoolbox/local/specification_local_data.json``. This should make it easier to share projects over e.g. Git.
- Scenario and tool filters are now ON by default.
 If new scenario or tool is added to a database it will be automatically selected in outgoing connections.
 NOTE: In existing projects, if a connection did not have any scenarios/tools selected, they will all be selected when opening a project for the first time after this change. If this is not desired, the scenarios/tools need to be deselected manually before saving the project on disk.
- Project name is now the project directory name and cannot be changed unless by moving the project to another directory manually or by using the File -> Save project as... menu item.
- Scenario filters now filter unrelated scenarios and alternatives as well.
- Alternative/Scenario tree in Database editor has been split into separate Alternative tree and Scenario tree docks.
 This will hopefully make dragging and dropping alternatives to scenarios easier.
- Logic of executing selected project items. URL's passed by unselected Data Transformers are not included automatically into selective execution anymore. When selecting project items for execution, make sure to select also preceding Data Transformers if needed.
- Names for duplicate items/scenarios/etc now use the format `prefix (xx)` instead of `prefix xx`.
 Duplicating a previously duplicated item now has the number `xx` incremented instead of having a new number appended.
- "Open kernel spec editor" buttons in Settings->Tools page have been changed "Make Julia kernel" and "Make Python Kernel" buttons

Deprecated

Removed

- Python 3.7 support
- GdxExporter project item. Please use Exporter instead.
- Gimlet project item. Please use Tool instead.
- The possibility to import parameter values when importing object groups using Importer. Existing mappings will not import group parameters anymore. Please add explicit parameter import mappings if needed.
- The possibility to export parameter values when exporting object groups using Exporter. Existing mappings will not export group parameters anymore.

(continues on next page)

(continued from previous page)

```

Please add explicit parameter export mappings if needed.
- Execute in work setting in Tool Specification Editor. Please use Execute in work
  ↳ setting
  in Tool Properties instead.
- Kernel Spec Editor widget and the "Open Kernel Spec Editor" button in Tool
  ↳ Specification Editor

#### Fixed
- Deleting the last item of a mapping in the importer specification editor no longer
  ↳ disables the 'add' button.
- Group Id's for Jupyter Consoles
- Kill consoles at end of execution for Jupyter Consoles
- Crash when typing exit, exit(), quit, or quit() into Jupyter Console

#### Security

## [0.6.5] - 2021-09-08

#### Added
- Support for loops has been added. Loops can be created using a special *loop link*
  ↳ which is initiated by
  holding down the **Alt** key while drawing a new link in Design view.
- Performance boost for Spine Db Editor when working with large databases.

#### Changed
- Installation does not require cx-Oracle and pycopg2 packages anymore.
- install_requires and requirements.txt files have been revised.

#### Fixed
- Spine Db Editor Graph View works on Python 3.7 again.
- Spine Db Editor Graph View 'Add objects' and 'Save position' actions.
- Export to SQLite in Spine Db Editor

## [0.6.3] - 2021-09-03

#### Added
- Plenty of stability improvements and bug fixes

#### Fixed
- Fixed 'ImportError: DLL load failed while importing win32api' on (Conda) Python 3.8
  ↳ when trying to execute Tools
  in Jupyter Console

## [0.6.1] - 2021-06-28

#### Added
- Data Transformer now supports parameter value transformations.
- Project execution shortcuts: F5 to execute all DAGs, F6 to execute selected items
  and F7 to stop execution.
- Time series, maps and other compound values have gained the ability to have names for
  ↳ their indexes.
  Index names can be edited in parameter value editors, and they are also supported by

```

(continues on next page)

(continued from previous page)

```

↳ Importer and Exporter items.
- Support for running Python Tools (specifications) in a Conda environment
- Execution mode (kernel spec, console, interpreter) can now be selected individually.
↳ for each Python
  Tool specification

#### Changed
- Data Transformer's specification editor has now a new interface.
- Parameter renaming in Data Transformer requires now entity class names to identify the.
↳ parameters.
  Data Transformer's icon will show a notification if class names are missing.
- Installation instructions advice to install directly from PyPI.
- Stand-alone DB Editor is now opened with the `spine-db-editor [URL]` command
- Python settings on the *Tools* page of *File->Settings* are now the default settings.
↳ for new Python Tool
  specifications. I.e. they are not global settings anymore.

#### Deprecated
- GdxExporter has been deprecated. Use the general purpose Exporter item instead.
  GdxExporter will be removed in a future release. Please replace existing items by.
↳ Exporter.

## [0.6.0-final.2] - 2021-06-03

#### Fixed
- [win-x64] Running Python or Julia Tools does not open an extra console window anymore

#### Security
- urllib3 v1.26.5 now required because of a security vulnerability in earlier versions

## [0.6.0-final.1] - 2021-06-01

#### Fixed
- Event Log and Item Execution Logs now automatically scroll to the bottom when there.
↳ are new messages
- [win-x64] Resolve correct GAMS, Python, and Julia paths in Settings->Tools

## [0.6.0-final.0] - 2021-05-07

#### Added
- Support for parallel/multicore processing
- New project item: Data Transformer. Can be used to configure Spine database filters.
↳ for successor items.
  Currently, it supports renaming entity classes.
- New project item: Exporter. A general-purpose tabular data exporter.
- Support for version 3 Spine Toolbox projects and an automatic upgrade of version 2.
↳ projects to version 3.
- Support for version 4 Spine Toolbox projects.
- Support for version 5 Spine Toolbox projects.
- Support for version 6 Spine Toolbox projects.
- Support to create sysimages for Julia tools.
- New requirement: jill, for installing Julia.

```

(continues on next page)

(continued from previous page)

- The SpineOpt configuration assistant has been moved from File->Configuration assistants, to File->Settings->Tools->Julia, and renamed to SpineOpt Installer.
- New wizard to install Julia, accessible from File->Settings->Tools->Julia.
- File->Close project option
- Support for Python 3.8
- Automated kernel creation, if the user selects to run tools in console without having created a kernel.
- Option to pack CSV resource files into one datapackage.json file for advertising, available from Link properties.
- Option to color project item icons in the toolbar, available from File->Settings->General.
- Reorganize project item icons in the toolbar with drag and drop.

Changed

- Project Item (Tool, Data Store, Importer, etc.) code has been removed from Spine Toolbox. Project Items are now in a separate package called spine_items, which is upgraded at Spine Toolbox's startup.
- Importer item now applies the same mapping to all input files. If the user needs to apply different mappings, they need to create different Importers. The specification can be shared using the json file.
- The .gdx exporter project item is now called GdxExporter.
- [win-x64] Installer does not require admin rights anymore
- [win-x64] Installer always asks for an installation directory, even if a previous installation exists
- [win-x64] Installer wizard style changed to modern

Removed

- Combiner project item. The same functionality can be achieved by connecting a Data Store to another Data Store.
- Upgrade support for original (.proj file based) Spine Toolbox projects.
- Python 3.6 is no longer supported.
- The Spine Datapackage Editor is gone. There wasn't enough reason to keep this widget
- The app no longer checks that Spine dependencies are up to date. Users are asked to follow the upgrade procedure which involves manually upgrading requirements after pulling the latest master branch.

Fixed

- [win-x64] returning_process.py when frozen
- Traceback in GdxExporter when there are indexing settings for a parameter that is not in the database
- Bug in installing Plugins
- Traceback when removing Plugins

[0.5.0-final.1] - 2020-02-03

Added

- Tutorial for case study A5 in the documentation

(continues on next page)

(continued from previous page)

```

#### Fixed
- [win-x64] Fixed /tools/python.exe by adding sitecustomize.py and a missing python37.dll

## [0.5.0-final.0] - 2020-12-14

#### Added
- Exporting graphs as PDF files from the *Graph* menu in the Data Store form.
- Pruning entire classes from the graph view. The option is available both from the
  ↳ *Graph* menu and
  ↳ from *Entity Graph* context menus. Also, pruned items can be restored gradually.
- A new Input type *Indexed parameter expansion* is now available in Data Store view's
  ↳ Pivot table.
  ↳ In this Input type the indexes, e.g. time stamps of time series get expanded as a new
  ↳ dimension in the table.
- Import editor now has a new Source type: Table name. It can be used e.g. to pick an
  ↳ Excel sheet's
  ↳ or GAMS domain's name as the object class name.
- Import editor now supports multidimensional maps. The number of dimensions can be set
  ↳ using the
  ↳ *Map dimensions* spin box in mappings options.
- Executing a project from the command line without opening the Toolbox GUI (i.e.
  ↳ headless execution).
  ↳ The headless execution is enabled by the new command line option ``--execute-only``.
- Toolbox now supports scenarios and alternatives. They can be accessed via Data store
  ↳ view's new Alternative tree.
- New Project Item: Gimlet. Can be used to run any command as part of the workflow
  ↳ with or without a shell. Supported shells at the moment are cmd and powershell for
  ↳ Windows and bash for other OS's.
- Python and Julia Kernel spec Editor. Provides the means to make new kernel specs for
  ↳ Python Console and Julia
  ↳ Console without leaving Spine Toolbox. Kernel (spec) Editor can be found in Settings->
  ↳ Tools tab.
- [win-x64] Includes Tools/python.exe for running Python Tools for systems that do not
  ↳ have a Python installation.
  ↳ Also, pyvenv.cfg and path.pth files for configuring the included python.exe.

#### Fixed
- Signal disconnection issue in Graph View
- Bugs in removing objects and object classes in Spine db editor's Graph View

#### Changed
- Data Store Form is now called 'Spine database editor'
- Spine db editor Graph View behavior. Now selecting objects in the object tree not only
  ↳ shows those objects but also
  ↳ all the cascading relationships. One can still go back to the previous behavior in
  ↳ Settings.
- Moving object items in the graph view also causes relationship icons to follow. This
  ↳ behavior can be disabled in the
  ↳ Settings.
- Required PySide2 version is now 5.14. The version is checked at startup.
- Indexed parameter handling has been overhauled in Exporter allowing parameters to
  ↳ share indexing domains.

```

(continues on next page)

(continued from previous page)

```

**Note**: Due to numerous changes in the backend, Exporters in old project files will
↳ not load properly
  and need to be re-configured.
- The way Exporter handles missing parameter values and None values has changed. The
↳ item now ignores missing
  values instead of replacing them by the default value. Further, there is a new option
↳ to replace None values by
  the default value and another option to replace Nones by not-a-numbers or skip
↳ exporting them.
- The numerical indicator on the upper left corner of project items no longer indicates
↳ the execution order for
  each individual item because the exact order is not know before the Execute button is
↳ actually clicked.
  The number still indicates the execution order but may show the same numbers for items
↳ in different parallel
  branches.
- Project.json file format has been upgraded to version 2. Version 1 project.json files
↳ are upgraded to version 2
  automatically when a project is opened.
- Default Python interpreter is now {sys.executable} i.e. the one that was used in
↳ launching the app.
  This affects the Python used by Python Tool specifications and the PyCall used by
↳ SpineOpt.jl configuration
  assistant.
- [win-x64] Default Python interpreter is the Python in user's PATH if available. If
↳ Python is not defined in
  user's PATH, the default Python interpreter is the <app_install_dir>/Tools/python.exe.
- User's now need to explicitly choose the kernel specs for the Python Console and the
↳ Julia Console. They are
  not chosen (nor created) automatically anymore. The kernel specs can be selected in
↳ the drop-down menus
  in application Settings->Tools.
- Database revision handling has been improved. Id est, the app does not offer to
↳ upgrade databases
  that are more recent than the current version of Spine Toolbox can handle.
- Links to User Guide and Getting Started documents open only the online versions. The
↳ docs have been
  published on readthedocs.org.
- Clearing the line edits for Julia executable and Python Interpreter (in Settings->
↳ Tools) shows
  the full paths to their respective files as placeholder text.

### Deprecated
- CustomQtKernelManager class

### Removed
- python_repl_widget.py
- julia_repl_widget.py

## [0.4.0-final.0] - 2020-04-03

### Added

```

(continues on next page)

(continued from previous page)

- A small notification icon is painted next to project items in the design view whenever [they](#) are missing some configuration. Hovering the icon shows tips for completing the configuration.
- A small icon is painted next to the project items in the design view to show the order [in](#) which they will be executed
- Main Window menu 'File -> Open recent'. Shortcut for opening a recent project.
- A new project item *Exporter* allows a database contained in a *Data Store* to be [exported](#) as GAMS *.gdx* file.
- It is now possible to copy and paste project items for example between projects.
- It is now possible to duplicate project items.
- Changes made in the tree view are also seen in the graph view and viceversa.
- New Setting: *Sticky selection in Graph View*. Enables users to select if they want to [use](#) multi-selection or single selection in the Graph view Object tree when selecting items [with](#) the *left-mouse button*.
- Projects can be saved to any directory
- Project name can be changed in Settings
- The graph view features a short live demonstration that new users can follow to [discover](#) the basic functionality.
- New Setting: *Curved links*. When active, links on the Design View follow a smooth [curve](#) rather than a straight line.
- When execution traverses a link, a small animation is played to denote the flow of [data](#). Users can set how quick they want this animation to be in Settings. The fastest setting effectively disables the animation.
- Special 'tag' command line arguments are now available in Tool Specification which [expand](#) to, for example, input database URLs or paths to optional input files when a Tool is [executed](#).
- It is now possible to undo/redo database changes in the Data Store form.
- It is now possible to visualize the history of database changes in the Data Store form. [The](#) option is available in the Session menu.
- Support for Tool command line arguments. You can now give Tool (project item) command [line](#) arguments in addition to Tool Specification command line arguments.
- Undo/Redo in Design View
- It is now possible to add new plots to existing plot windows in Data Store View.
- Objects in Data Store's Tree View are sorted alphabetically
- A new parameter value type, *map* has been added. There is now a dedicated editor to modify maps. Plotting of non-nested maps is supported, as well.
- [win-x64] importer_program.py has been built as an independent application. This [program](#) is now distributed with the Spine Toolbox single-file installer. When installing Spine [Toolbox](#), the Importer Program app can be found in the Spine Toolbox install directory

(continues on next page)

(continued from previous page)

```
(/importer_program).
```

- Import preview window now supports copy-pasting mappings and options from a source table to another
- Import preview window header context-menus for the preview table which, allows users to change all data types at once.
- Provide data for EditRole for nicer editor experience in MappingSpecModel.
- Red background is displayed for invalid values in MappingSpecModel
- Object tooltips now show the descriptions Data Store view's

Fixed

- Data advertised by a project item during execution is only accessible by its direct successors. In other words, resources are passed to the next items in line but not beyond.
- [win-x64] Executing the Importer project item has been fixed on Windows release version
- Bug fixes for Data Store View
 - Disappearing object names in entity graph
 - Spine db manager error dialogs
- Tool configuration assistant for SpineModel.jl
- [win-x64] A problem with displaying special characters in Process Log when executing the Importer project item.
- The context menu in Graph view's pivot table resulted in a traceback when an entity class did not have parameter definitions.
- Combobox delegate in Import preview window had wrong list of choices.
- Don't set mapping to NoneMapping if user gives a None value.
- Exporter now also exports empty object classes and empty parameters into GDX files
- A bug that sometimes made duplicate entries to File->Open recents menu
- Bug where an Excel import with empty rows would return a None in it's get_data_iterator
- [win-x64] Executing Julia tools in the embedded Julia Console
- [win-x64] Setting up Python Console. Installing ipykernel and kernel specs now works.
- [win-x64] Setting up Julia Console. Installing IJulia and kernel specs now works.
- Column indexing in Import Editor. When entering a time or time pattern index column manually in Import Editor's lower right corner table, the colors in the preview table failed to update. This should fix the bug and allow column selection both by column index or column header text.

Changed

- spinetoolbox is now a Python package. To start the app, use command ``python -m spinetoolbox`` or ``python spinetoolbox.py`` as spinetoolbox.py has been moved to repository root.
- Tool templates are now called Tool specifications
- File->Open Project opens a file dialog, where you can open projects by selecting an old `<project_name>.proj` file or a Spine Toolbox Project directory. Valid Spine Toolbox projects are decorated with the Spine logo.
- Old style projects (.proj files) cannot be opened anymore. The old style projects need to be upgraded before opening. You can upgrade your .proj file projects into new ones with

(continues on next page)

(continued from previous page)

```

*Project Upgrade Wizard* found in `File->Upgrade project` menu item.
- Project information is not saved to a <project_name>.proj file anymore. This
  information
  is now located in file <project_dir>/spinetoolbox/project.json. Every Spine Toolbox
  project
  has this file.
- Work directory is now a global setting instead of a project setting
- Renamed *Data Interface* project item to *Importer*. The corresponding category
  *Data Importers* was renamed to *Importers*.
- Tree, graph, and tabular views have been merged into one consolidated view. You can
  choose
  your preferred style from the Data Store View's `View` menu.
- The graph view behavior has changed. Now selecting objects in the object tree not only
  shows those objects but also all the cascading relationships. This is to facilitate
  exploring
  the system without a previous knowledge.
- importer_program.py now uses the Python interpreter that the app was started with and
  not
  the one that is given by user in Settings -> Tools.
- Importer now uses QProcessExecutionManager for running the importer_program.py

#### Removed
- The status bar of the Data store view is gone. Instead, notifications are printed in a
  box
  on the right side of the form.
- Saving project information to .proj files is not happening anymore

## [0.3] - 2019-09-06

#### Added
- Welcome message including a link to Getting Started guide.
- Zooming (by using the mouse wheel) is now enabled in Design View. You can also select
  multiple project
  items by pressing the Ctrl-key down and dragging the mouse.
- New project item icons on Design View.
- Two options for the Design View background (grid or solid). See Settings (F1).
- Python Console (menu View -> Dock Widgets -> Python Console).
- Python interpreter can be selected in Settings (Also Python 2.7 supported).
- Support for Python Tool templates
- Python Tool execution using the command line approach.
- Python Tool execution using the embedded Python Console (checkbox in Settings)
- The Data Store treeview now also has a relationship tree.
- Support for reordering columns in the treeview.
- Selecting 'edit object classes' in the treeview now also allows the icon to be
  selected.
- Play button to main window Toolbar that executes all Directed Acyclic Graphs in the
  Project one after another.
- Another Play button to main window Toolbar to execute selected Directed Acyclic Graph.
  Selecting a DAG happens by
  selecting one or more project items belonging to the wanted DAG in the Design View or
  in Project Items list.
- Stop button to main window Toolbar, which terminates execution.

```

(continues on next page)

(continued from previous page)

- Possibility to specify a dedicated Julia project or environment for Spine Toolbox in the settings.
- Feature to Export project to GraphML format. Each graph in Design View is written to its own file. To do this, just select *Export project to GraphML* from the Project Item list context-menu or from *File -> Export project to GraphML*.
- New project item: *Data Interface*
- Parameter and relationship parameter values can now be edited in a dedicated editor window in Tree, Graph and Tabular views. The editor is accessible by right-clicking a value and selecting `Open in editor...`.
- It is now possible to plot parameter and relationship parameter values in Tree, Graph and Tabular views.

Fixed

- There is now an upper limit on how much text is logged in Event Log and Process Log. The oldest lines are removed if the limit is exceeded. This fixes a problem with excessive memory usage when running long processes.
- Mouse click problems in Design View
- Mouse cursor problem in Tool Configuration Assistant
- Welcome message is now shown for first time users
- NameError: SpineDBAPIError when executing Data Stores.
- .py files in *spinedb_api\alembic\versions* are now copied to the installation bundle without compiling them to .pyc files first. Also, if there's a previous version installed, *spinedb_api\alembic* directory is deleted before installation begins [Win-x64].
- Added missing modules from *spinedb_api\alembic\versions* package into installation bundle [Win-x64].
- *numpy* is now deleted before installation begins so installing over an existing installation of Spine Toolbox works again [Win-x64].
- *packaging* and *appdirs* packages are now included in the installation bundle [Win-x64].

Changed

- Selecting the Julia environment in Settings now requires picking the Julia interpreter (**file** (e.g. julia.exe on Windows) instead of the directory where the Julia interpreter is located.
- Selecting the GAMS program (**file**) in Settings now requires picking the GAMS program (e.g. gams.exe on Windows) instead of the directory where the GAMS program is located.
- All application Settings are now saved using Qt's QSettings class. Old configuration file, *conf/settings.conf* file has been removed.
- New Spine databases can be created in any backend supported by spinedb_api. The DataStore item properties have been changed to allow for this.
- Executing Directed Acyclic Graphs instead of just Tools.

(continues on next page)

(continued from previous page)

```
- Executing project items does not happen from the Tool Properties anymore. It happens.
↳ instead by pressing the
  Play button in the main window Toolbar.

#### Removed
- An unnecessary error dialog in Import Preview widget.
- ConfigurationParser class in configuration.py.
- Execute button in Tool Properties.
- Stop button in Tool Properties.
- Console window that opened in addition to the application window is not shown anymore.
↳ [Win-x64].

## [0.2] - 2019-01-17

#### Added
- New Setting. You can now select whether to delete project item's data directory
  when removing a project item.
- Graph View is also available from Data Store items, allowing to insert new
  relationships more graphically.
- You can now execute Tools in the work directory or the source directory (where the
  main program file is located). The Setting is in the Tool template editor and in Tool
  properties.
- Tabular view: New view for Data Store items, allowing to view and edit data in a
  pivot table.
- Tool Properties now shows all information about the attached Tool template
- Context-menus for Data Connection Properties, Tool Properties, and View Properties
- Support for optional input files for Tool templates. You can now use Unix style.
↳ wildcards (`*` and `?`)
  to specify the optional files that a Tool may exploit, e.g. `*.csv`.
- Wildcard support for Tool template output files
- Tool template output files now support subdirectories
- You can now create a new (blank) main program file in the Tool template editor by.
↳ pressing the the `**` button
  and selecting `Make new main program`
- A shortcut to Tool template main directory, accessible e.g. in Tool Properties context-
↳ menu
- New Setting. You can select whether the zoom in the Graph view is smooth or discrete
- Windows installer default location is /Program Files/SpineToolbox/. When new versions.
↳ are released, the
  installer will automatically upgrade Spine Toolbox to a new version in that directory.
- Support for executing Executable type tools in Linux&Mac. (Windows support was added.
↳ previously)
- Tool configuration assistant. Available in menu `File->Tool configuration assistant`.
↳ Checks automatically
  if the Julia you have selected in Settings supports running Spine Model. If not, the.
↳ required packages are
  automatically installed.

#### Fixed
- Better support for scaling screen resolutions
- Exporting a datapackage to Spine format failed if datapackage.json was not explicitly.
↳ saved
```

(continues on next page)

(continued from previous page)

Changed

- Importing items with names into a spinedatabase moved to spinedatabase_api to allow for easier adding of new import formats in future versions.
- Tool template list is now located in the Project dock widget below the list of Project_ items
- New look for Spine Datapackage editor

Removed

- Templates tab in `Project` dock widget.

[0.1.75] - 2018-11-23

Added

- New Setting (F1). You can now select whether to delete project item's data directory when removing a project item.
- Application icon (Spine symbol)
- New installer for 64-bit Windows:
 - Installation file extension is now *.exe* instead of *.msi*
 - Show license file before installation (users must agree to continue)
 - Default install folder is now `C:\Program Files\`.
 - ****No**** need to ***Run as Administrator*** even if installed to the default location because write permissions for sub-folders that need them (\conf, \projects, \work) are set automatically
 - Create a shortcut on desktop (if wanted)
 - Create a Start Menu folder (if wanted)
 - Uninstaller. Available in the Start Menu folder or in Windows Add/Remove Programs
 - Remove app related registry entries when uninstalling (if wanted)

Fixed

- Data Package editor. Some files were missing from the tabulator package.
- Bug when exiting the app when there is no project open in close_view_forms() when exiting the application without a project open

Changed

- settings.conf file is now in /conf directory

[0.1.71] - 2018-11-19

Added

- Added PyMySQL package, which was missing from the previous release
- Improved Graph View for the View project item (work in progress)

Changed

- Some main window components have been renamed
 - Main view is now called ***Design View***
 - Julia REPL is now called ***Julia Console***
 - Subprocess Output is now called ***Process Log***
 - Project item info window is now called ***Properties***

[0.1.7] - 2018-11-01

(continues on next page)

(continued from previous page)

Added

- New option to refresh the data store tree view and get latest changes from the database.
- Several performance enhancements in tree view form (accessing internal data more efficiently, optimizing queries to the database.)
- Now the data store tree view offers to commit pending changes at exit.
- Better support for batch operations in data store tree view.
- data store tree view can be fully operated by using the keyboard.
- New options to edit object tree items in the data store tree view, including changing the objects involved in a relationship.
- The dialogs to add/edit tree view items are not closed in case of an error, so the user can adjust their choices and try again.
- Stop button now terminates tool execution.
- New context menu options to fully expand and collapse object tree items in the data store tree view.
- The autofilter in the data store tree view now also filters work in progress rows.
- In the Data store item controls, the path to the SQLite file can be specified by dropping a file.
- Parameter and parameter value tables in the data store tree view now have an empty row at the end, which can be used to enter data more easily.
- JSON data can be visualized and edited more easily in the data store tree view.
- Tools can now execute (Windows) batch files and other executables (.exe). Linux support pending.
- About Qt dialog added to Help menu

Fixed

- Clicking on the open treeview button while the data store tree view is open now raises it, rather than opening a second one.
- Work folder is not created for Tools if the Tool template requirements are not met.
- Result folder is not created if the Tool template fails to start.
- The embedded Julia REPL now uses the Julia that is given in application Settings (F1). Previously, this used the default Julia of your OS.

Removed

- Connections tab has been removed (actually, it is just hidden and can be restored with a keyboard shortcut)
- Refresh Tools button on Templates tab has been removed as it was not needed anymore
- Set Debug message level in Settings has been removed

[0.1.5] - 2018-09-28

Added

- Advanced copy/pasting and multiple selection support for the data store tree view.
- Import data from Excel files into the data store tree view.
- Export Spine database from the data store tree view into an Excel file.
- Save at exit prompt.

(continues on next page)

(continued from previous page)

- Import data from datapackage into the data store tree view.
- Restore Dock Widgets in the main window.
- Parameter tables can be filtered by clicking on their headings in the data store tree.
→view.
- Parameter and parameter values are added and edited directly in the data store tree.
→view,
without need for an additional dialog.
- On-the-fly creation of necessary relationships when entering parameters in data store.
→tree view.
- View item feature for visualizing networks from a Spine database. A view item can.
→visualize databases
from all data store items connected to it.
- Packages numpy, scipy, and matplotlib are now mandatory requirements.
- Drag files between data connections. File items can be dragged from the references and.
→data lists.
Data connection items can be selected by hovering them while dragging a file. Dropping.
→files onto a Data Connection
item copies them to its data directory.
- datapackage.json files in data connections are now opened with the Spine datapackage.
→form. This is a dedicated
interface to prepare the datapackage for importing in the data store tree view.
- The data store tree view does not lock the database when there are uncommitted changes.
→anymore.

Changed

- Changed DBAPI package mysqlclient (GPL license, not good) to pymysql (MIT license,
→good)
- spinedatabase_api is not included in Spine Toolbox repository anymore. It is a required package from now on.
- Data Store item can have only one database, not many. When opening a project created.
→with a
previous version, the first database in the list of saved references will be loaded.
→for each Data Store.
- In the data store tree view, the object tree presents all relationship classes at the.
→same level,
regardless of how many object classes are involved. The same applies for relationships.
→and objects.
- In the data store tree view, the relationship parameter value view now has different.
→columns for each object
in the relationship.

[0.1] - 2018-08-20

Added

- Basic functionality

GETTING STARTED

Welcome to the Spine Toolbox’s getting started guide. In this guide you will learn two ways of running a “Hello, World!” program on Spine Toolbox. If you need help on how to run **SpineOpt.jl** using Spine Toolbox, see chapter *How to Set up SpineOpt.jl*. For small example projects utilizing **SpineOpt.jl**, see *SpineOpt tutorials*.

This chapter introduces the following topics:

- *Spine Toolbox Interface*
- *Creating a Project*
- *Creating a Tool Specification*
- *Adding a Tool Item to the Project*
- *Executing a Tool*
- *Editing a Tool Specification*
- *Adding a Data Connection Item to the Project*
- *Adding Data Files to a Data Connection*
- *Connecting Project Items*

2.1 Spine Toolbox Interface

The central element in Spine Toolbox’s interface is the **Design View**, which allows you to visualize and manipulate your project workflow. In addition to the **Design View** there are a few *dock widgets* that provide additional functionality:

- **Project** provides a more concise view of your project, including the *Items* that are currently in the project, grouped by category: Data Stores, Data Connections, Tools, Views, Importers, Exporters and Manipulators.
- **Properties** provides an interface to interact with the currently selected project item.
- **Event Log** shows relevant messages about user performed actions and the status of executions.
- **Console** allows Spine Toolbox to execute Tools written in Python, Julia or GAMS and provides an interface to interact with the aforementioned programming languages. Also shows a list of parallel executions available in the project.

In addition to the **Design View** and the dock widgets, the main window contains a **Toolbar** split into two sections. The **Items** section contains the project items that you can drag-and-drop onto the **Design View** while the **Execute** section has buttons related to executing the project.

Tip: You can drag-and-drop the dock widgets around the screen, customizing the interface at your will. Also, you can select which ones are shown/hidden using either the **View -> Dock Widgets** menu, or the main menu **Toolbar**'s context menu. Spine Toolbox remembers your configuration between sessions. Selecting **Restore Dock Widgets** from the **View -> Dock Widgets** menu restores the widgets back to their default location.

Tip: Most elements in the Spine Toolbox's interface are equipped with *tool tips*. Leave your mouse cursor over an element (button, checkbox, list, etc.) for a moment to make the tool tip appear.

2.2 Creating a Project

To create a new project, please do one of the following:

- From the application main menu, select **File -> New project...**
- Press **Ctrl+N**.

The *Select project directory (New project...)* dialog will show up. Browse to a folder of your choice and create a new directory called 'hello world' there. Then select the 'hello world' directory and click Select Folder. Spine Toolbox will populate the selected directory with some files and directories it needs to store the project's data.

Congratulations, you have created your first Spine Toolbox project.

2.3 Creating a Tool Specification

Note: Spine Toolbox is designed to run and connect multiple tools, which are specified using **Tool specifications**. You may think of a Tool specification as a self-contained program specification including a list of source files, required and optional input files, and expected output files. Once a Tool specification is added to a project, it can then be associated to a Tool item for its execution as part of the project workflow.

Note: Just like the main window, the **Tool specification editor** consists of dock widgets that you can reorganize however you like.

In the **Toolbar**, click on the icon next to the Tool icon , to reveal the Tool specification list. Since there are none in the project yet, click on the button to open the **Tool specification editor**. Follow the instructions below to create a minimal Tool specification:

- Type 'hello_world' into the *Name:* field.
- Select 'Python' from the *Tool type* dropdown list,
- Click on the button next to the *Main program file* text in the **Program files** dock widget. A *Create new main program file* file browser dialog opens. Name the file `hello_world.py` and save it e.g. directly to the 'hello world' project directory or to a folder of your choice.

We have just created a `hello_world.py` Python script file, but at the moment the file is empty. Spine Toolbox provides an mini **IDE** where you can view and edit the contents of Tool specification files. Let's try it out.

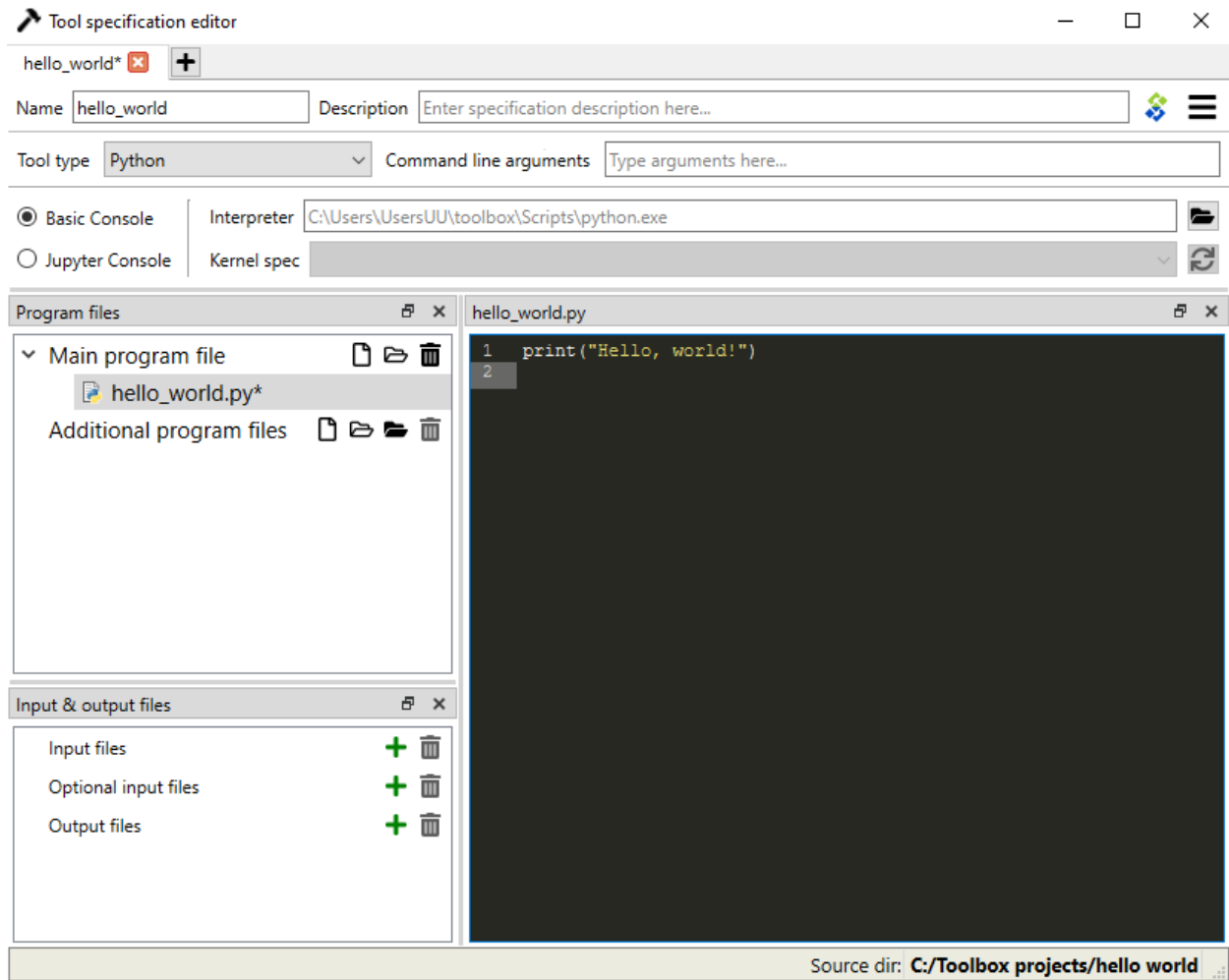
Select `hello_world.py` below the *Main Program File*. Click on the (black) editor dock widget with the title 'hello_world.py'.

Type in the following:

```
print("Hello, world!")
```

Now, whenever `hello_world.py` is executed, the sentence 'Hello, World!' will be printed to the standard output.

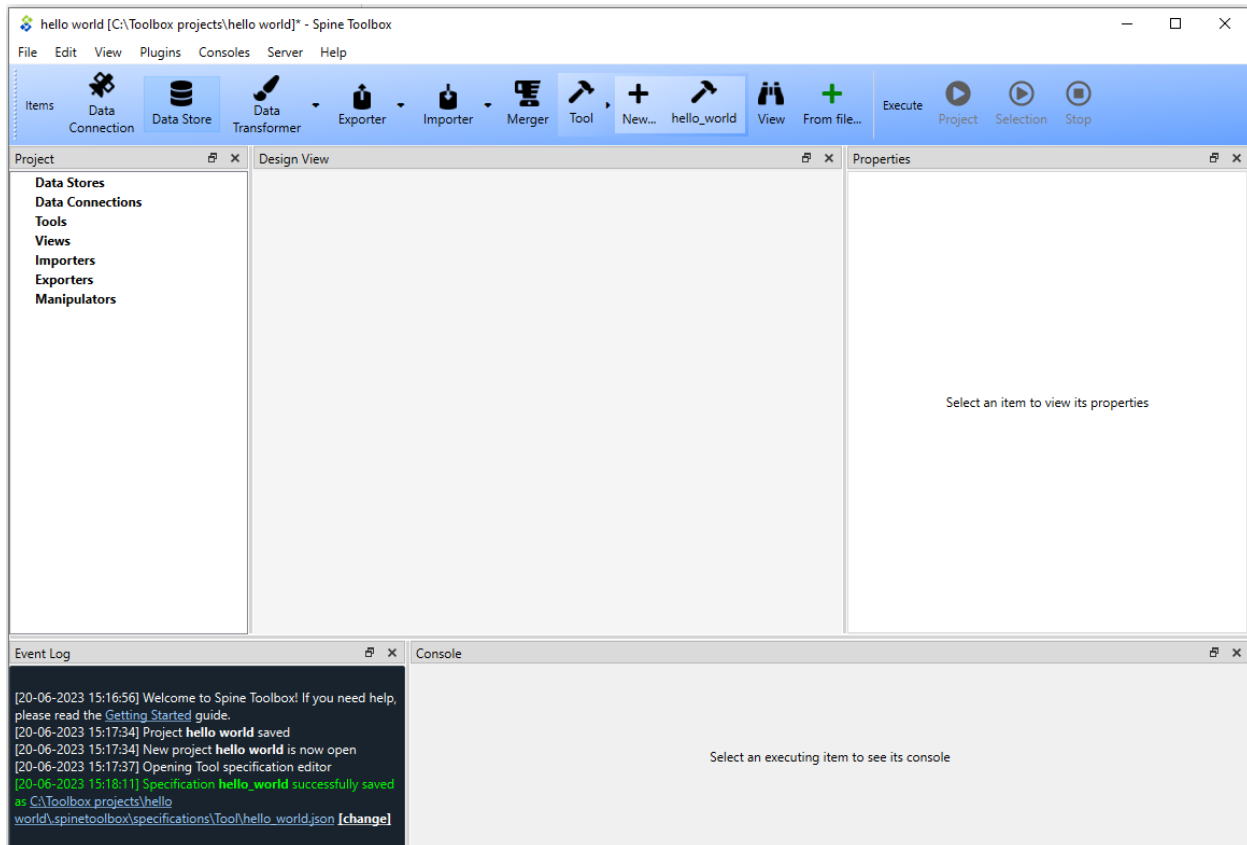
The **Tool specification editor** should be looking similar to this now:



Note that the program file (`hello_world.py`) and the Tool specification (`hello world`) now have unsaved changes. This is indicated by the star (*) character next to `hello_world.py*` and the Tool specification name in the tab bar (`hello_world*`).

- Save changes to both by either pressing **Ctrl+S** or by mouse clicking on **Save** in the hamburger menu in the upper right hand corner.
- Close **Tool specification editor** by pressing **Alt+F4** or by clicking on 'X' in the top right hand corner of the window.

Your main window should look similar to this now.



Tool specifications are saved by default in JSON format into a dedicated directory under the project directory. If you want, you can open the newly created `hello_world.json` file by clicking on the file path in the Event log message. The file will open in an external editor provided that you have selected a default program for files with the `.json` extension (e.g in Windows 10 you can do this in **Windows Settings -> Apps -> Default apps**). In general, you don't need to worry about *the contents* of the JSON Tool specification files. Editing these is done under the hood by the app.

If you want to save `hello_world.json` somewhere else, you can do this by clicking the white `[change]` link after the path in the Event Log.

Tip: Saving the Tool specification into a file allows you to add and use the same Tool specification in another project. To do this, you just need to click the *From file...* button () in the **Toolbar** and select the Tool specification file (`.json`) from your system.

Congratulations, you have just created your first Tool specification.

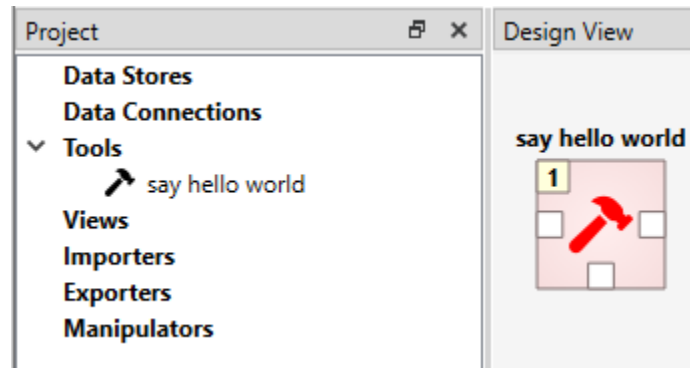
2.4 Adding a Tool Item to the Project

Note: The Tool project item is used to run Tool specifications.

Let's add a Tool item to our project, so that we're able to run the Tool specification we created above. To add a Tool item drag-and-drop the Tool icon from the **Toolbar** onto the **Design View**.

The **Add Tool** form will popup. Change name of the Tool to 'say hello world', and select 'hello_world' from the dropdown list just below, and click **Ok**. Now you should see the newly added Tool item as an icon in the **Design View**,

and also as an entry in the **Project** dock widget, under the 'Tools' category. It should look similar to this:

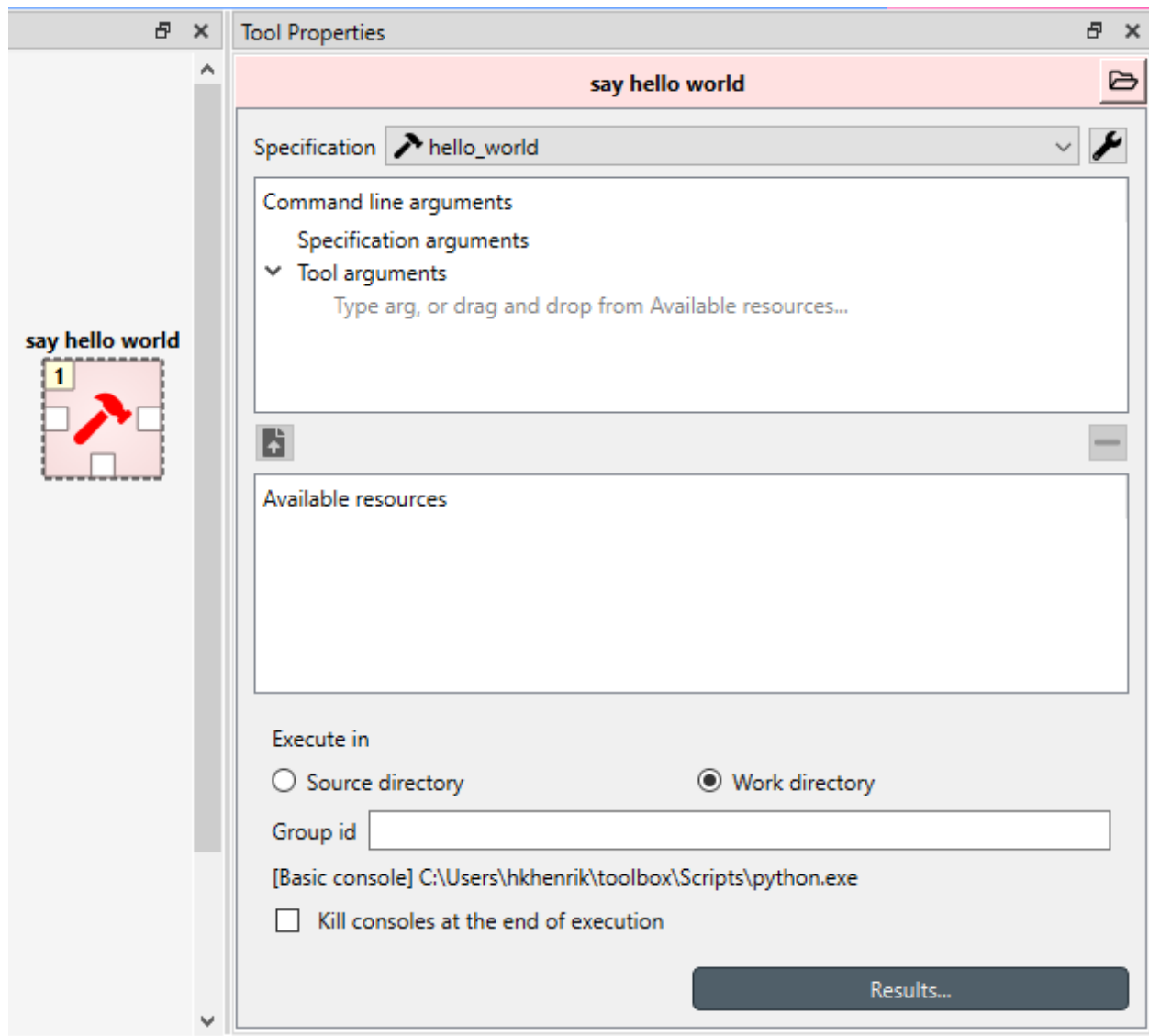


Another way to do the same thing is to drag the with the 'hello world' text from the **Toolbar** onto the **Design View**. Similarly, the **Add Tool** form will popup but the 'hello world' tool specification is already selected from the dropdown list.

Note: The Tool specification is now saved to disk but the project itself is not. Remember to save the project every once in a while when you are working. You can do this by selecting **File -> Save project** from the main window or by pressing **Ctrl+S** when the main window is active.

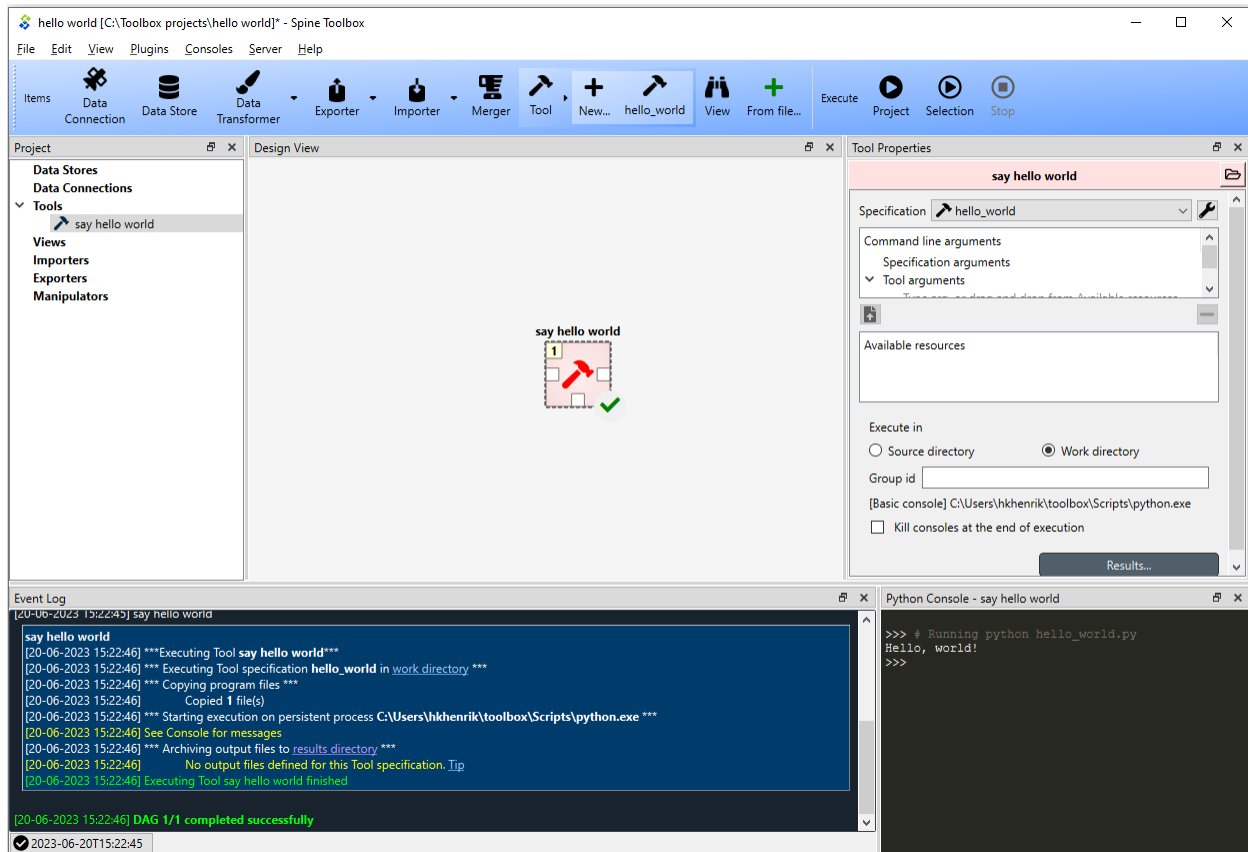
2.5 Executing a Tool

Select the 'say hello world' Tool on **Design View**, and you will see its *Properties* in the dedicated dock widget. It looks similar to this:



Press **execute project** button on the **Toolbar**. This will execute the 'say hello world' Tool project item which now has the 'hello world' Tool specification associated to it. In actuality, this will run the main program file `hello_world.py` in a dedicated process.

Once the execution is finished, you can see the details about the item execution as well as the whole execution in **Event Log**. The **Console** contains the output of the executed program file.



Note: For more information about setting up Consoles in Spine Toolbox, please see [Setting up Consoles and External Tools](#) for help.

Congratulations, you just executed your first Spine Toolbox project.

2.6 Editing a Tool Specification

To make things more interesting, we will now specify an *input file* for our ‘hello_world’ Tool specification.

Note: Input files specified in the Tool specification can be used by the program source files, to obtain input data for the Tool’s execution. When executed, a Tool item looks for input files in **Data Connection**, **Data Store**, **Exporter**, and **Data Transformer** project items connected to its input.

Open the Tool specification editor for the ‘hello world’ Tool spec. You can do this for example, by double-clicking the ‘say hello world’ Tool in **Design View**, or by right clicking the ‘say hello world’ -item in the **Project** dock widget and selecting **Specification... -> Edit specification**, or from the **Tool Properties** by clicking the Tool specification options button () next to the specification and selecting **Edit specification**.

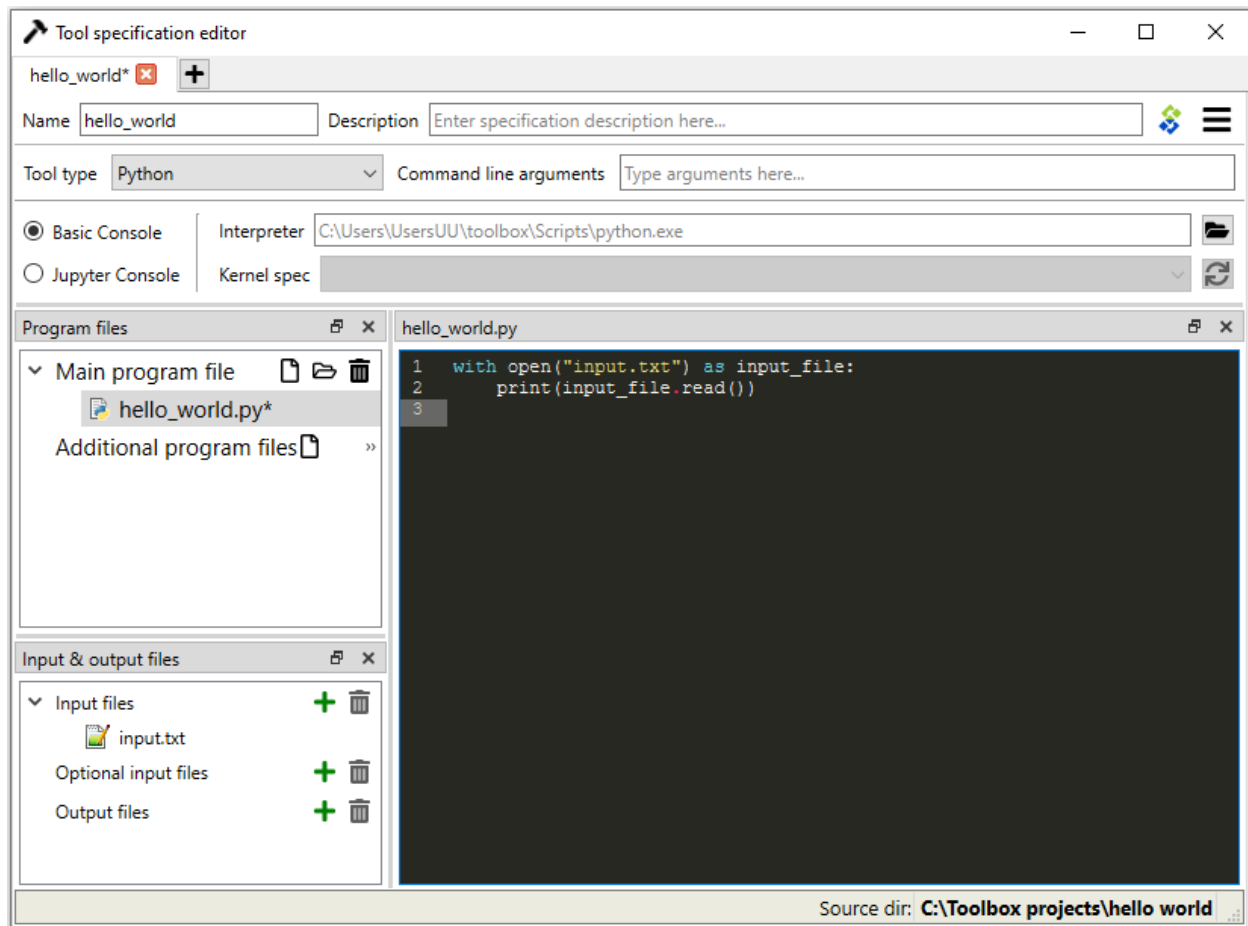
In **Input & Output files** dock widget, click the button next to the *Input Files* text. A dialog appears, that lets you enter a name for an input file. Type 'input.txt' and press Enter.

So far so good. Now let's use this input file in our program. Still in the Tool specification editor, replace the text in the main program file (`hello_world.py`), with the following:

```
with open("input.txt") as input_file:
    print(input_file.read())
```

Now, whenever `hello_world.py` is executed, it will look for a file called `input.txt` in the current directory, and print its content to the standard output.

The editor should now look like this:

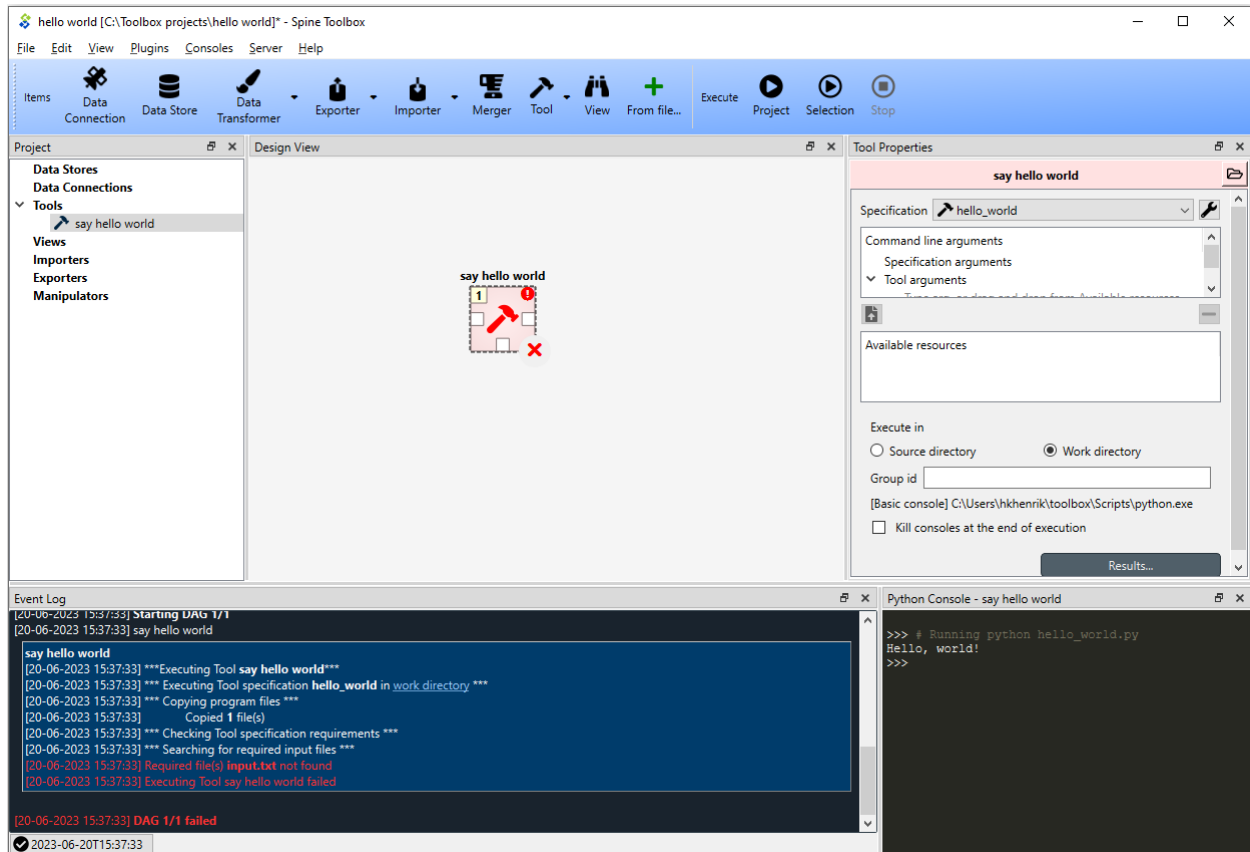


Save the specification and close the editor by pressing **Ctrl+S** and then **Alt+F4**.

Note: See *Tool Specification Editor* for more information on editing Tool specifications.

Back in the main window, note the exclamation mark on the Tool icon in **Design View**, if you hover the mouse over this mark, you will see a tooltip telling you in detail what is wrong. If you want you can try and execute the Tool anyway

by pressing in the **Toolbar**. *The execution will fail* because the file `input.txt` is not made available for the Tool:



2.7 Adding a Data Connection Item to the Project

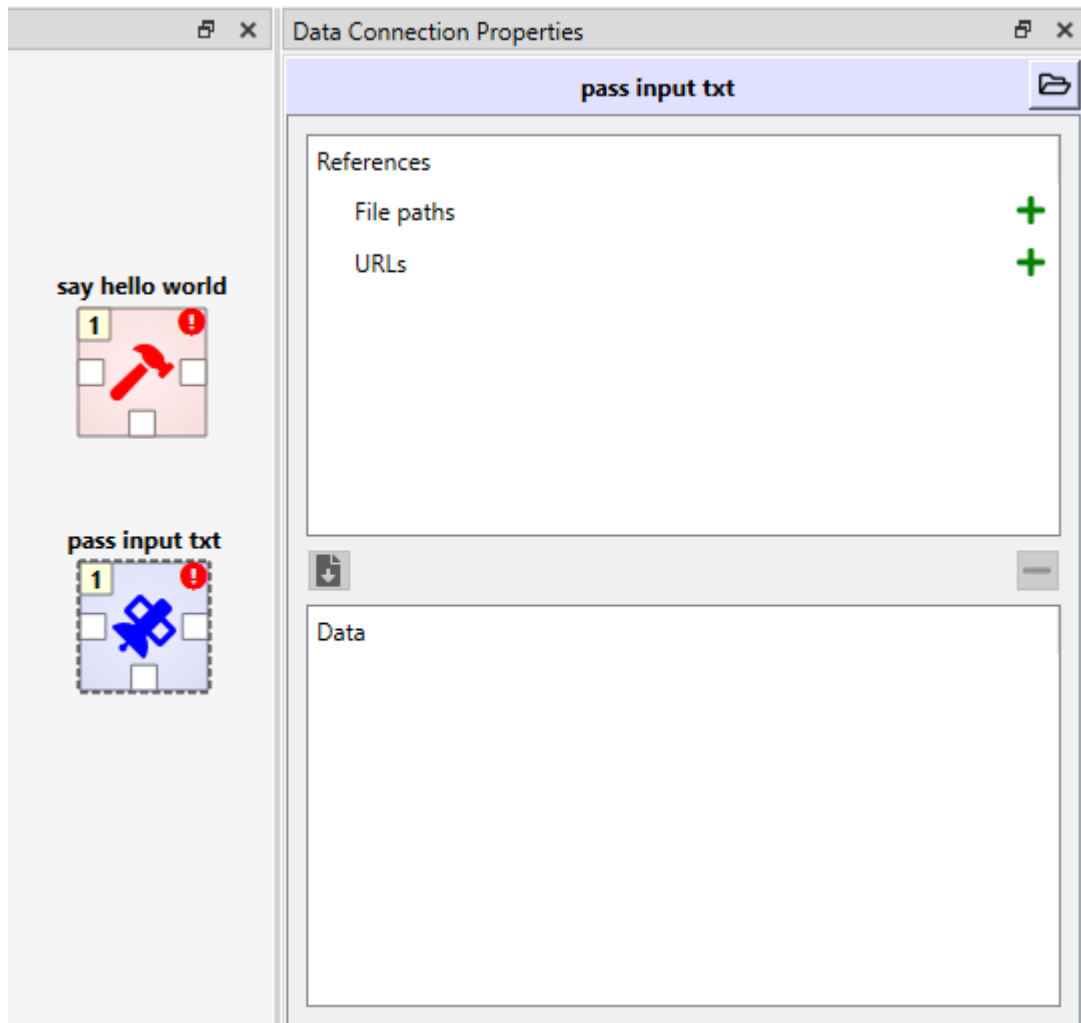
Note: The Data Connection item is used to hold generic data files, so that other items, notably Importer and Tool, can make use of that data.

Let's add a Data Connection item to our project, so that we're able to pass the file `input.txt` to 'say hello world'. To add a Data Connection item, drag-and-drop the Data Connection icon () from the **Toolbar** onto the **Design View**.

The *Add Data Connection* form will show up. Type 'pass input txt' in the name field and click **Ok**. The newly added Data Connection item is now in the **Design View**, and also as an entry in the **Project** dock widgets items list, under the 'Data Connections' category.

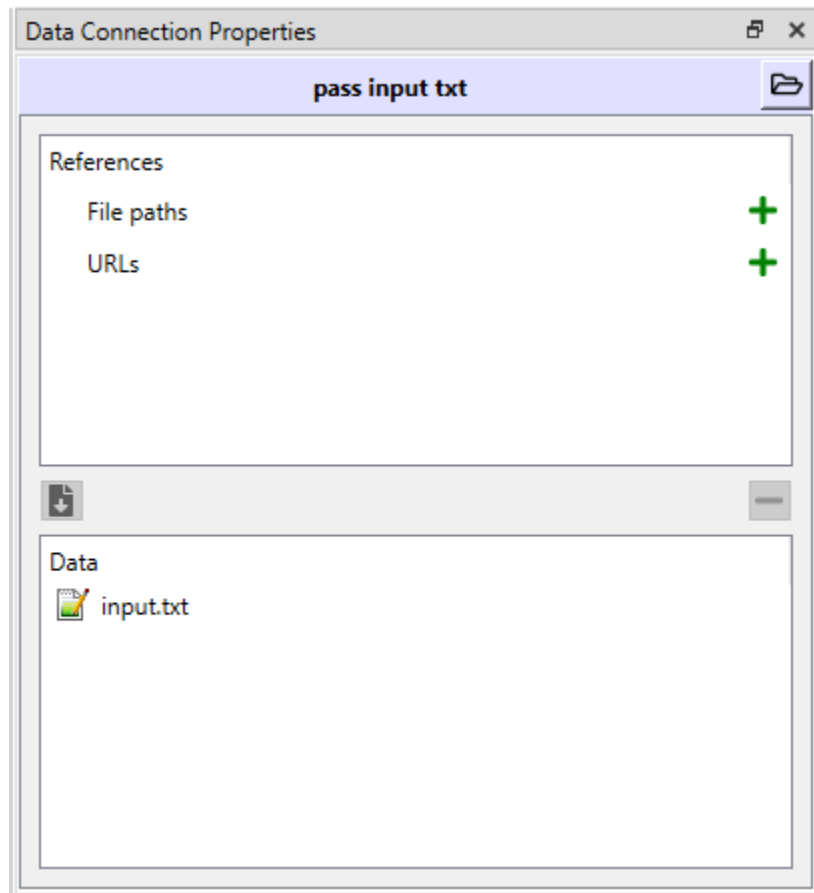
2.8 Adding Data Files to a Data Connection

Select the ‘pass input txt’ Data Connection item to view its properties in the *Properties* dock widget. It should look similar to this:



Right click anywhere within the **Data** box and select **New file...** from the context menu. When prompted to enter a name for the new file, type ‘input.txt’ and click **Ok**.

There’s now a new file in the *Data* list:



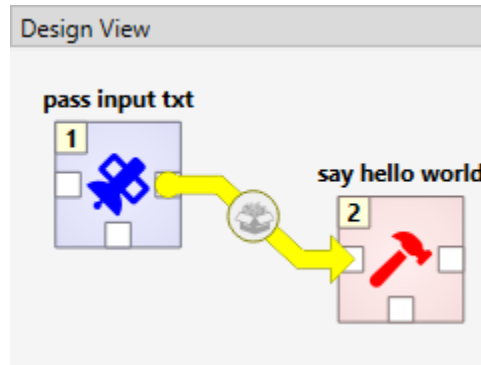
Double click this file to open it in your default text editor. Then enter the following into the file's content:


Hello again, World!

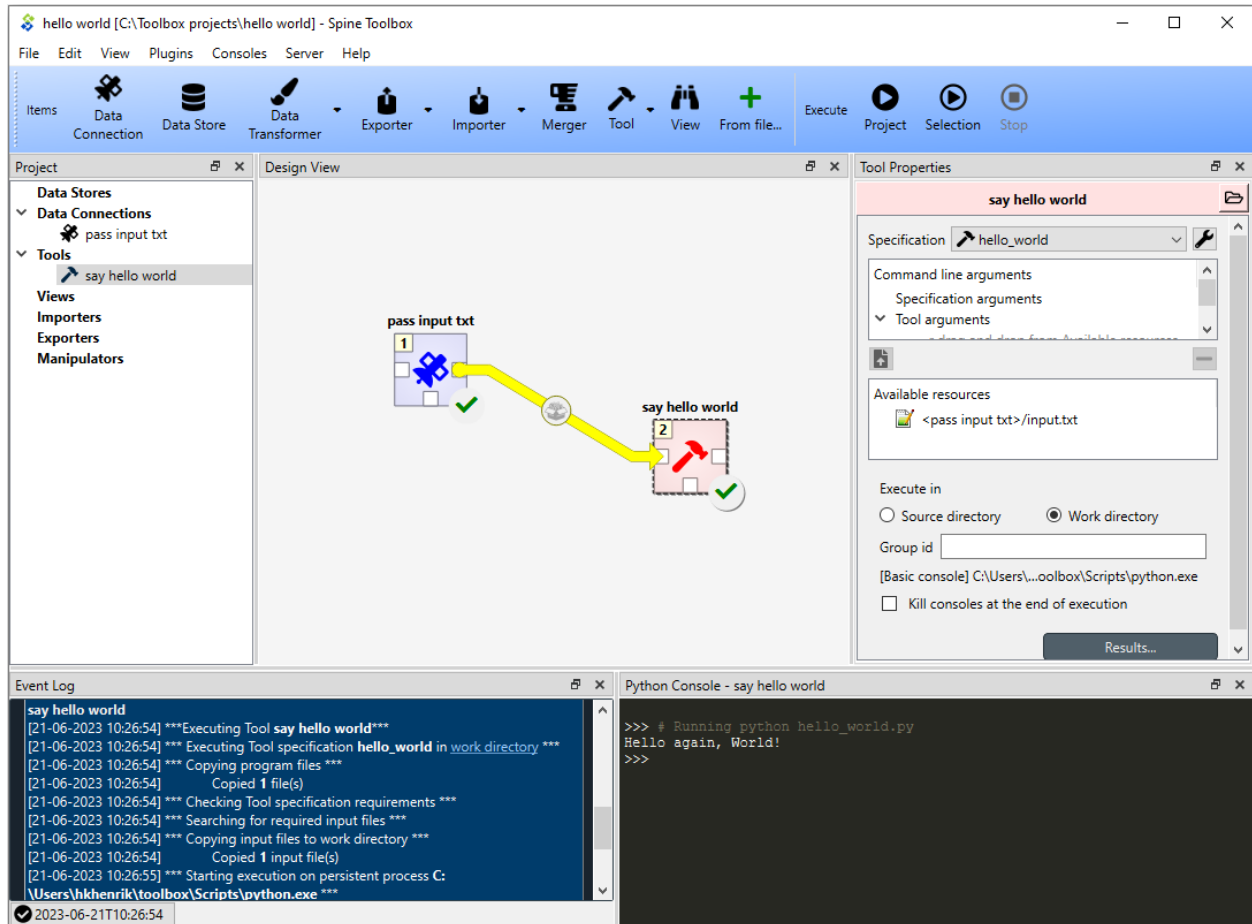
Save the file.

2.9 Connecting Project Items

As mentioned above, a Tool item looks for input files in Data Connections or other items connected to its input. Thus you now need to create a connection from 'pass input txt' to 'say hello world'. To do this, click on one of the *connector* slots at the edges of 'pass input txt' in the **Design view**, and then on a similar slot in 'say hello world'. This will create an arrow pointing from one to another, as seen below:



Press  once again. The project will be executed successfully this time:

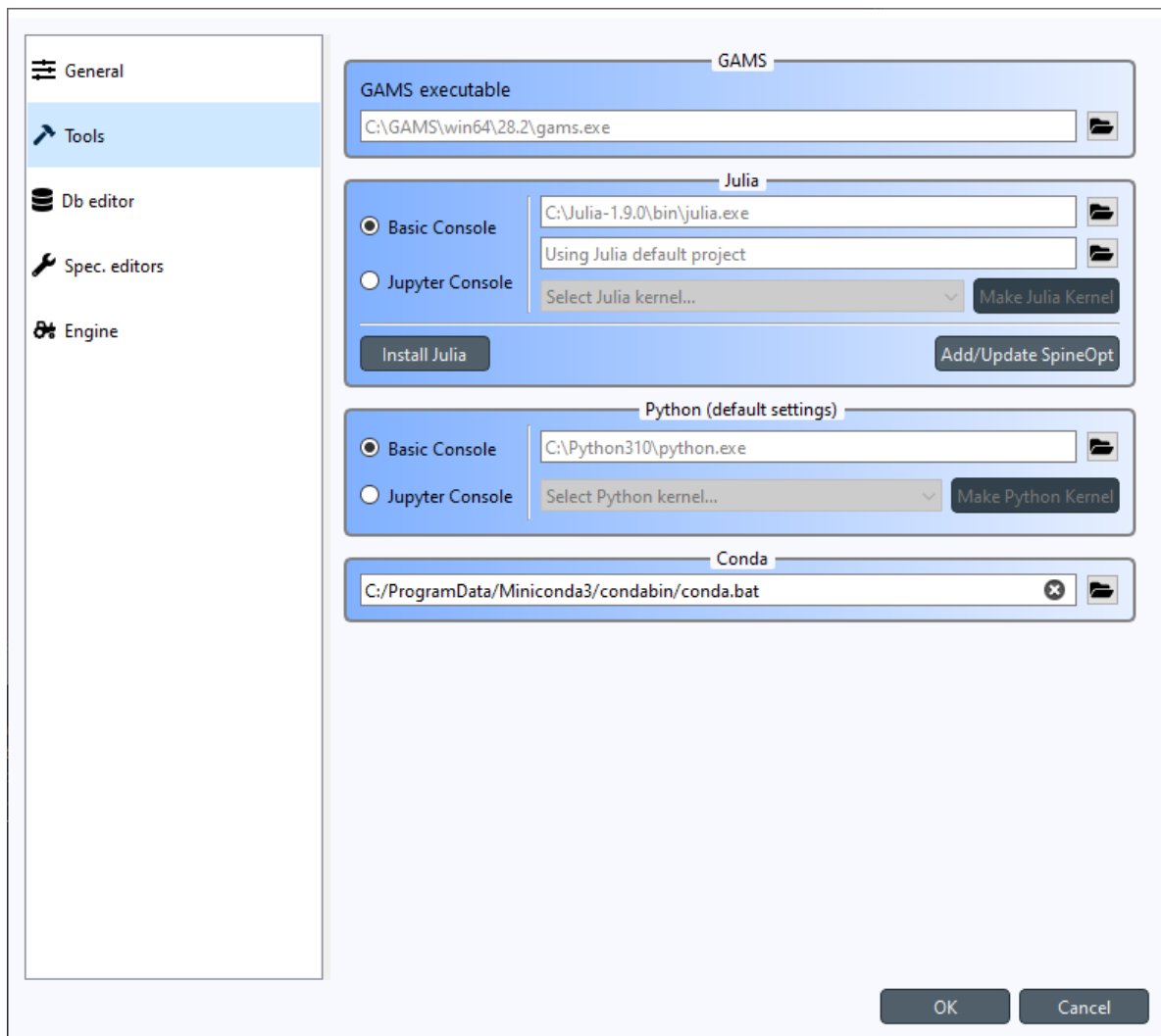


That's all for now. I hope you've enjoyed following this guide as much as I enjoyed writing it. See you next time.

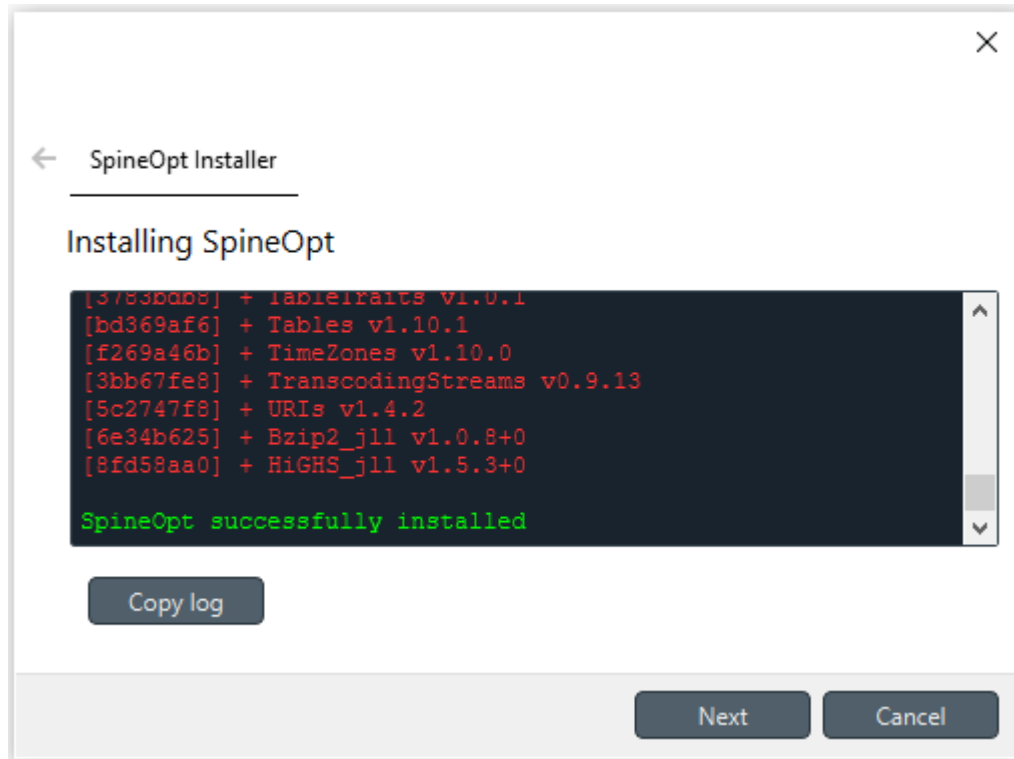
Where to next: If you need help on how to set up and run **SpineOpt.jl** using Spine Toolbox, see chapter [How to Set up SpineOpt.jl](#). After setting up SpineOpt, there are three tutorials over on **SpineOpt.jl**'s documentation that will help you get started on using SpineOpt in Spine Toolbox: [Simple system](#), [Two hydro plants](#), and [Case study A5](#).

HOW TO SET UP SPINEOPT.JL

1. Install Julia (v1.6 or later) from <https://julialang.org/downloads/> if you don't have one. See latest **SpineOpt.jl** Julia compatibility information [here](#).
2. Start Spine Toolbox
3. Create a new project (**File -> New project...**)
4. Select **File -> Settings** from the main menu and open the *Tools* page.
5. Set a path to a Julia executable to the appropriate line edit (e.g. *C:/Julia-1.6.0/bin/julia.exe*). Your selections should look similar to this now.



6. [Optional] If you want to install and run SpineOpt in a specific Julia project environment (the place for *Project.toml* and *Manifest.toml*), you can set the path to the environment folder to the line edit just below the Julia executable (the one that says *Using Julia default project*).
7. Next, you need to install **SpineOpt.jl** package for the Julia you just selected for Spine Toolbox. You can do this manually by [following the instructions](#) or you can install **SpineOpt.jl** by clicking the **Add/Update SpineOpt** button. After clicking the button, an install/upgrade SpineOpt wizard appears. Click **Next** twice and finally **Install SpineOpt**. Wait until the process has finished and you are greeted with this screen.



Close the wizard.

8. Click **Ok** to close the **Settings** window
9. Back in the main window, select **Plugins -> Install plugin...** from the menu
10. Select *SpineOpt* and click **Ok**. After a short while, a red **SpineOpt Plugin Toolbar** will appear in the main window.

Spine Toolbox and Julia are now correctly set up for running **SpineOpt.jl**. Next step is to [Create a project workflow using SpineOpt.jl](#) (takes you to SpineOpt documentation). See also [Tutorials](#) in SpineOpt documentation for more advanced use cases. For more information on how to select a specific Python or Julia version, see [Setting up Consoles and External Tools](#).

Note: The **SpineOpt Plugin Toolbar** contains an exporter specification as well as three predefined Tools that make use of SpineOpt.jl. **The SpineOpt Plugin is not a requirement to run SpineOpt.jl**, they are provided just for convenience and as examples to get you started quickly.

SETTING UP CONSOLES AND EXTERNAL TOOLS

This section describes the options for executing different Python, Julia, Gams, and Executable Tools and how to set them up. To get started with **SpineOpt.jl**, see [How to Set up SpineOpt.jl](#). See also [Executing Projects](#).

- *Basic Consoles*
 - *Python*
 - *Julia*
- *Jupyter Consoles*
 - *Python*
 - *Julia*
 - *Conda*
 - *Detached Consoles*
- *GAMS*
- *Executable*

Python and Julia Tools can be executed either in an embedded *Basic Console* or in a *Jupyter Console*. GAMS Tools are executed in a sub-process. Executable Tools (external programs) are executed in a shell or by running the executable file straight. You can also make a Tool that executes a shell command by creating an *Executable* Tool Spec in **Tool Specification Editor**, entering the shell command to the *Command* line edit and then selecting the Shell for this Tool.

4.1 Basic Consoles

Basic Console appears to the **Console** dock widget in the main window when you execute () a project containing either a Python or a Julia Tool with Basic Console selected.

4.1.1 Python

Executing a Python Tool in the Basic Console requires no set up. Simply create a *Python* Tool Spec in **Tool Specification Editor** and select the Basic Console radio button. The default Python interpreter used in launching the Console is the same Python that was used in launching Spine Toolbox. You can also select another Python by changing the Python interpreter line edit. Remember to save the new Tool Spec when closing the **Tool Spec. Editor**. Then drag the Python Tool Spec into the **Design View**, and press to execute it.

Note: The Python settings on the *Tools* page in **File -> Settings** are the *default* settings for new Python Tool Specs. You can select a different Python executable for each Python Tool Spec separately using the **Tool Specification Editor**.

4.1.2 Julia

To execute Julia Tools in the Basic Console, first install Julia (v1.6 or later) [from here](#) and add `<julia install path>/bin` to your PATH environment variable (if not done automatically by the installer). Then go to the *Tools* page in **File -> Settings** and make sure that the Basic Console radio button is selected in the Julia group. If Julia is in your PATH, the Julia executable line edit should show the path as (grey) placeholder text. If you want to use another Julia on your system, you can change the path in the line edit. You can also set a Julia Project below the Julia executable line edit.

Note: The Julia settings are *global* application settings. All Julia Tools are executed with the settings selected on the *Tools* page in **File -> Settings**. In upcoming versions, the Julia settings will be consistent with the Python settings, in a way that you can select a specific Julia executable and Julia project for each Julia Tool Spec separately.

4.2 Jupyter Consoles

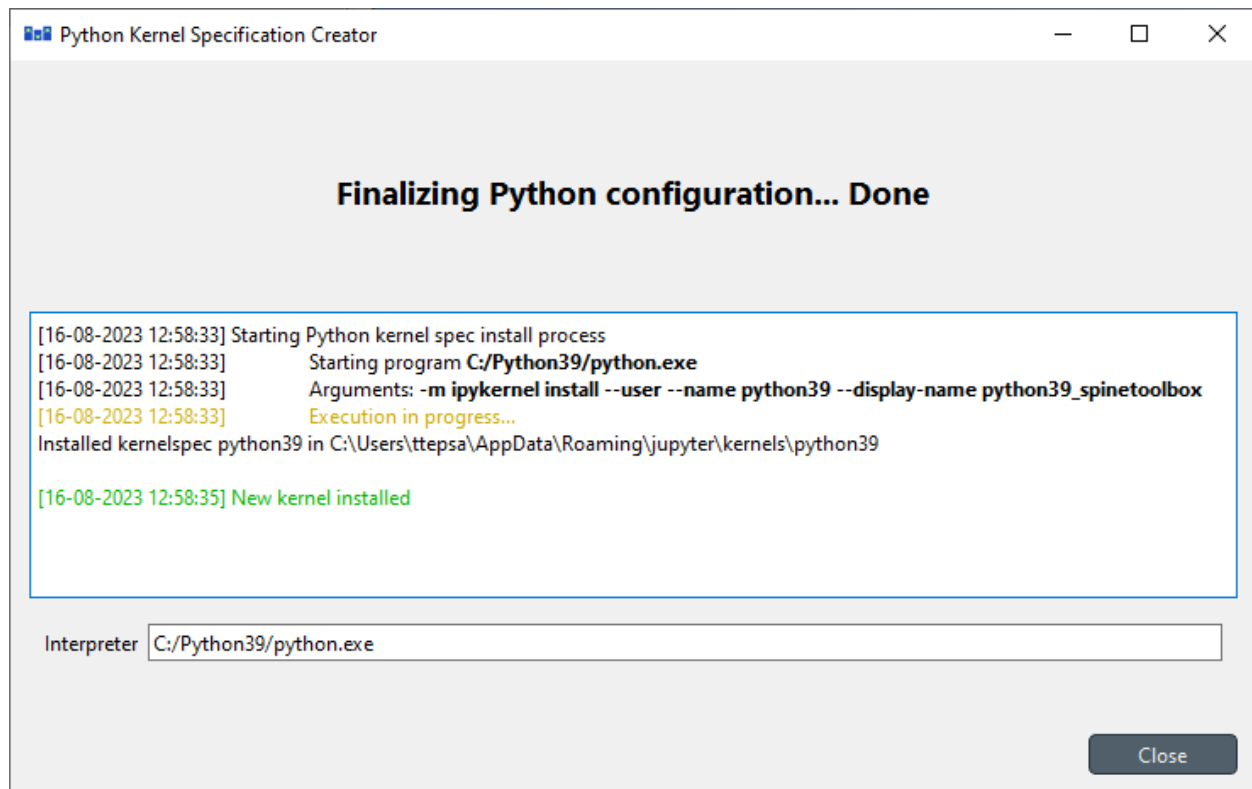
Jupyter Console appears to the **Console** dock widget in the main window when you execute () a project containing either a Python or a Julia Tool with the *Jupyter Console* selected. The Jupyter Console requires a Jupyter kernel to be installed on your system. Kernels are programming language specific processes that run independently and interact with the Jupyter Applications and their user interfaces.

4.2.1 Python

Select *Jupyter Console* radio button in **File -> Settings**. You also need to select the Python kernel you wish to use from the *Select Python kernel...* combo box. If this list is empty, you need to install the kernel specs on your system. You can either do this manually or click the **Make Python Kernel** button. Clicking the button opens a **Python Kernel Specification Creator** window, that first installs the **ipykernel** package (if missing) for the Python that is currently selected in the Python interpreter line edit (the kernel specs will be created for `C:/Python39/python.exe` in the picture below). You can make kernel specs for other Pythons and virtual environments (venv) by changing the Python interpreter line edit path to point to another Python. Please see specific instructions for creating kernel specs for Conda environments below.

The screenshot shows the 'Settings' dialog box in Spine Toolbox. On the left is a sidebar with icons and labels for 'General', 'Tools' (selected), 'Db editor', 'Spec. editors', and 'Engine'. The main area is divided into four tabs: 'GAMS', 'Julia', 'Python (default settings)', and 'Conda'.
 - The 'GAMS' tab has a 'GAMS executable' field with the path 'C:\GAMS\win64\28.2\gams.exe'.
 - The 'Julia' tab has radio buttons for 'Basic Console' and 'Jupyter Console' (selected). It includes fields for 'C:\Julia-1.9.0\bin\julia.exe', 'Using Julia default project', and a 'Select Julia kernel...' dropdown. Buttons for 'Install Julia', 'Make Julia Kernel', and 'Add/Update SpineOpt' are present.
 - The 'Python (default settings)' tab has radio buttons for 'Basic Console' and 'Jupyter Console' (selected). It includes fields for 'C:/Python39/python.exe' and a 'Select Python kernel...' dropdown, with a 'Make Python Kernel' button.
 - The 'Conda' tab has a field for 'C:/ProgramData/Miniconda3/condabin/conda.bat'.
 At the bottom right are 'OK' and 'Cancel' buttons.

Once the **ipykernel** package is installed, the wizard runs the **ipykernel** install command, which creates the kernel specs directory on your system. You can quickly open the kernel spec directory from the **Select Python Kernel...** combo box's context-menu (mouse right-click menu). Once the process finishes, click Close, and the newly created kernel spec (*python39* in this case) should be selected automatically. Click *Ok* to close the **Settings** widget and to save your selections.



If something went wrong, or if you want to remake the kernel specs, you can remove the kernel spec directory from your system, try the **Make Python Kernel** button again, or install the kernel specs manually.

If you want to install the Python kernel specs manually, these are the commands that you need to run.

To install **ipykernel** and its dependencies, run:

```
python -m pip install ipykernel
```

And to install the kernel specs run:

```
python -m ipykernel install --user --name python39 --display-name python39_spinetoolbox
```

Make sure to use the `--user` argument to make sure that the kernel specs are discoverable by Spine Toolbox.

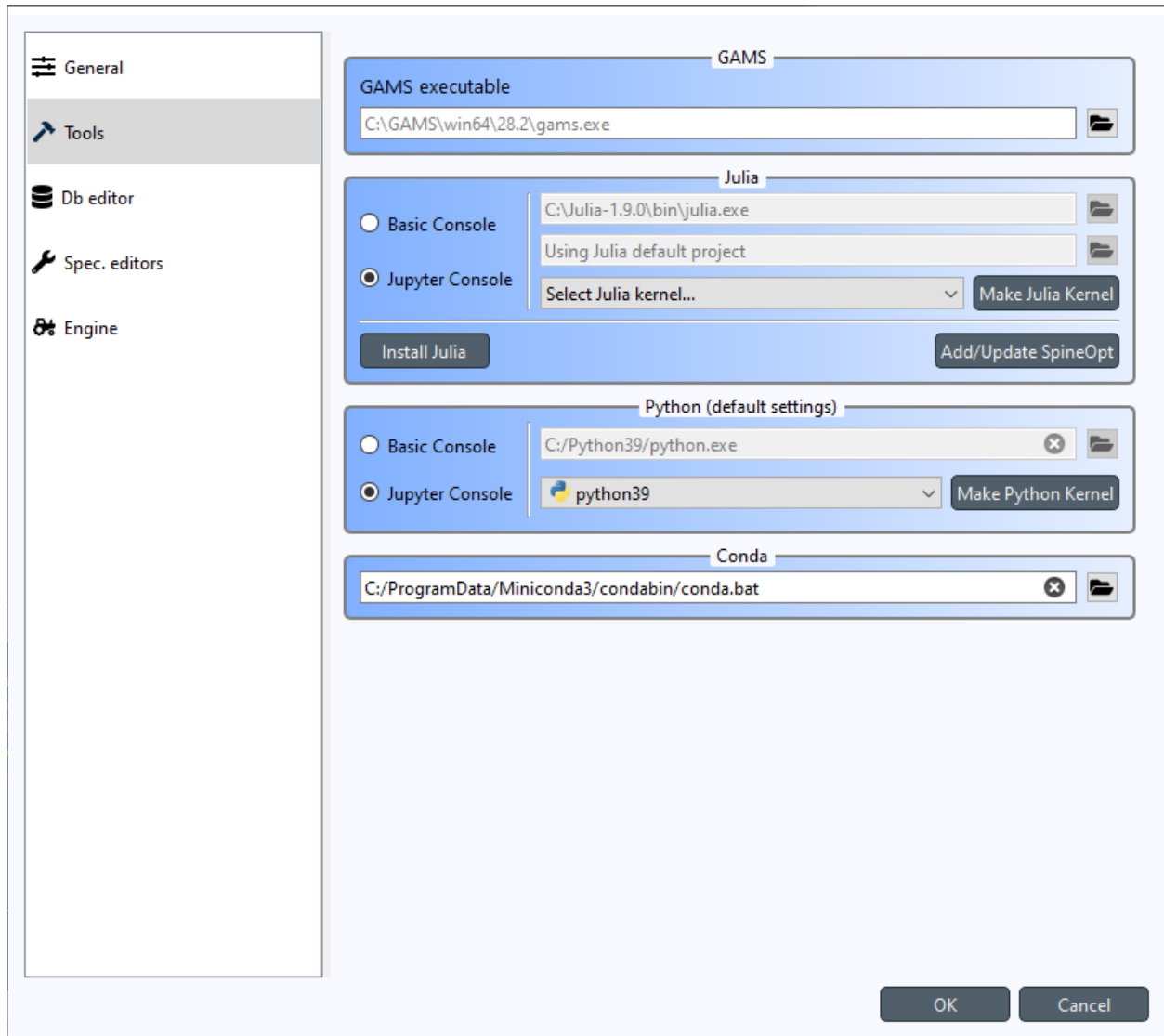
Note: Clicking **Make Python Kernel** button when the kernel specs have already been installed, does NOT open the **Python Kernel Specification Creator**, but simply selects the Python kernel automatically.

Note: Executing Python Tools using the Jupyter Console supports Python versions from 2.7 all the way to latest one. This means, that if you still have some old Python 2.7 scripts lying around, you can incorporate those into a Spine Toolbox project workflow and execute them without modifications.

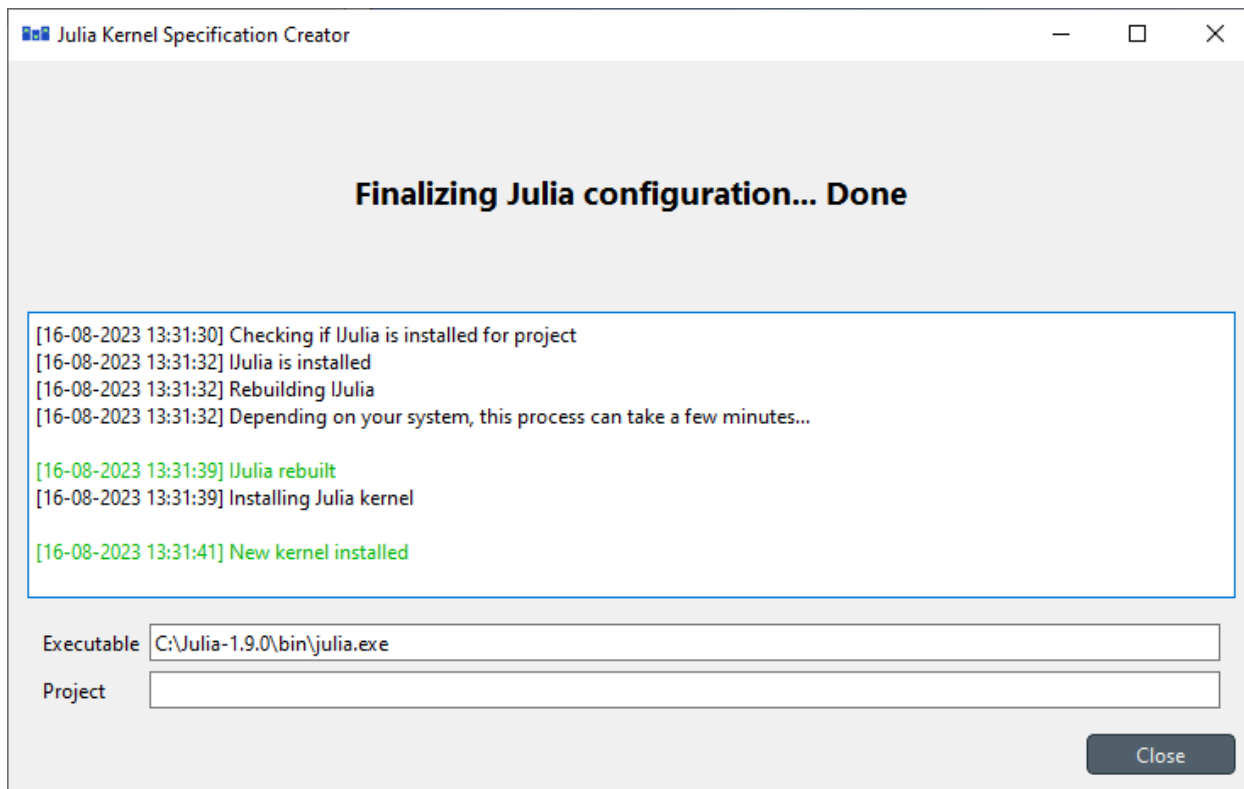
Important: If you want to have access to *spinedb_api*, you need to install it manually for the Python you select here.

4.2.2 Julia

To use the Jupyter Console with Julia Tools, go to the *Tools* page in **File -> Settings** and select the Jupyter Console radio button like in the picture below.



Like with Python, you need to select an existing Julia kernel for the Julia Jupyter Console, or create one either manually, or by clicking the **Make Julia Kernel** button.



Clicking the button opens **Julia Kernel Specification Creator** window, that first installs the **IJulia** package (if missing) for the Julia and Julia project that are currently selected in the Julia executable and Julia Project line edits (the kernel specs will be created for the default project of *C:/Julia-1.9.0/bin/julia.exe* in the picture above).

If something went wrong, or if you want to remake the kernel specs, you can remove the kernel spec directory from your system, try the **Make Julia Kernel** button again, or install the kernel specs manually.

If you want to install the Julia kernel specs manually, these are the commands that you need to run.

To install **IJulia** and its dependencies, open Julia REPL with the project you want and run:

```
using Pkg
Pkg.add("IJulia")
```

Rebuild IJulia:

```
Pkg.build("IJulia")
```

And to install the kernel specs run:

```
using IJulia
installkernel("julia", --project="my_project")
```

Note: Clicking **Make Julia Kernel** button when the kernel specs have already been installed, does NOT open the **Julia Kernel Specification Creator**, but simply selects a Julia kernel that matches the selected Julia executable and Julia Project. If a kernel spec matching the Julia executable is found but the Julia project is different, a warning window appears, saying that Julia kernel spec may be overwritten if you continue.

4.2.3 Conda

You also have the option of running Python Tools in a Conda environment. All you need to do is the following.

1. Open Anaconda Prompt and make a new Conda environment:

```
conda create -n test python=3.10
```

2. Activate the environment:

```
conda activate test
```

3. Install **ipykernel**:

```
pip install ipykernel
```

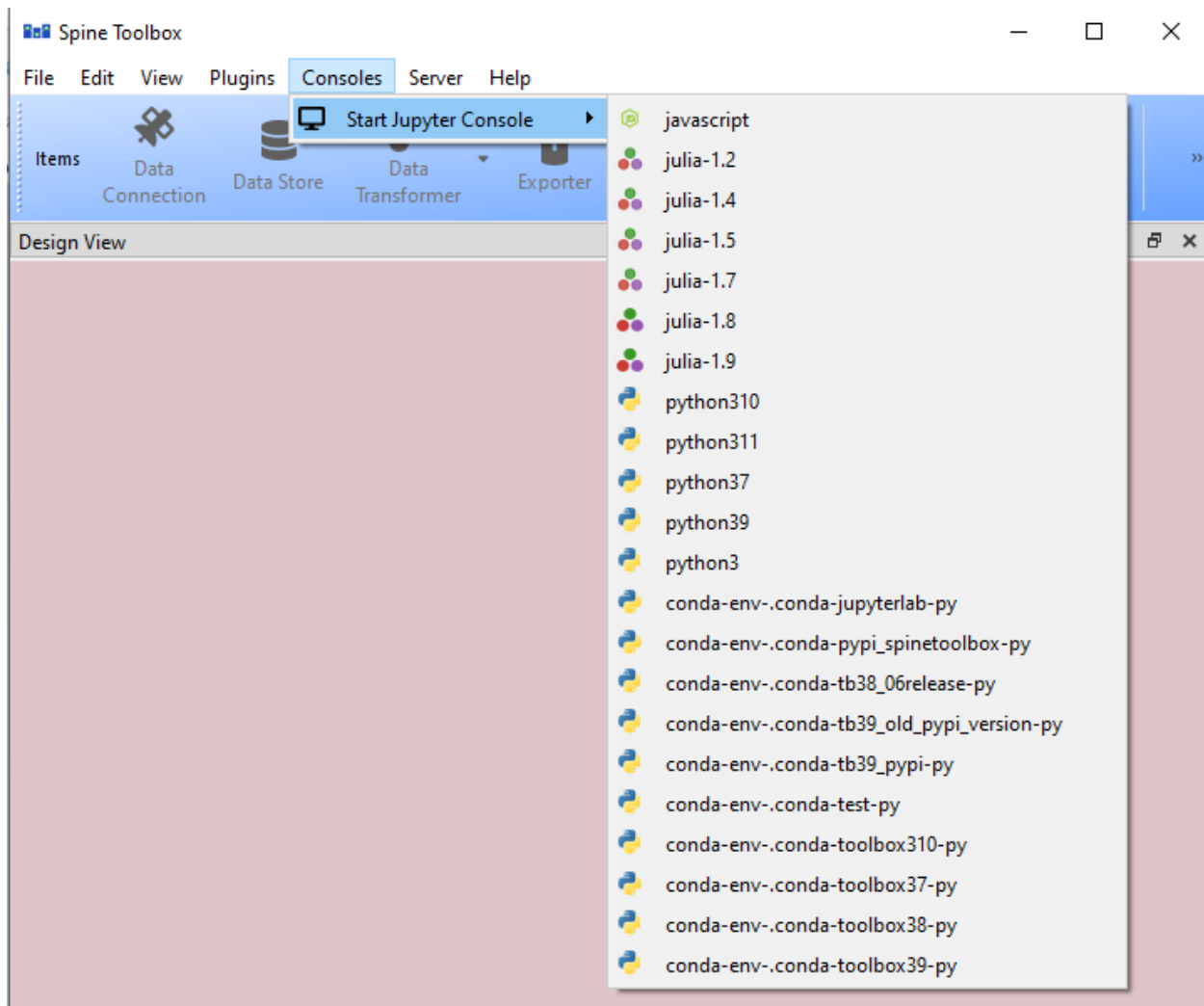
4. Back in Spine Toolbox, add path to Conda executable on the *Tools* page in **File -> Settings**.

That's it! Now, in Spine Toolbox main window, open the **Consoles -> Start Jupyter Console** menu, wait a second, and the new kernel should appear in the list. In this case, the new kernel name is *conda-env-.conda-test-py*. This autogenerated name will most likely change to something more readable in the future. You can use Conda Python kernels just like regular Python kernels, i.e. select one of them as the default kernel in the **File -> Settings** widget or select them for individual Python Tool Specs in **Tool Specification Editor** directly.

4.2.4 Detached Consoles

You can open 'detached' Jupyter Consoles from the main window menu **Consoles -> Start Jupyter Console**. The menu is populated dynamically with every Jupyter kernel that Spine Toolbox is able to find on your system. 'Detached' here means that the Consoles are not bound to any Tool. These Consoles are mostly useful e.g. for checking that the kernel has access to the correct packages, debugging, small coding, testing, etc. These may be especially useful for checking that everything works before running a full workflow that may take hours to finish.

Officially, Spine Toolbox only supports Python and Julia Jupyter kernels but it's possible that other kernels can be accessed in a Detached Console as well. For example, if you install a javascript kernel on your system, you can open a Detached Console for it, but this does not mean that Spine Toolbox projects should support Javascript. However, if there's interest and legitimate use cases for other kernels, we may build support for them in future releases.



If interested, you can [read more on Jupyter kernels](#) . There you can also find a [list of available kernels](#).

4.3 GAMS

Executing Gams Tools or needing to use the GDX file format requires an installation of Gams on your system. You can download Gams from <https://www.gams.com/download/>.

Note: You do not need to own a Gams license as the demo version works just as well.

Important: The bitness (32 or 64bit) of Gams has to match the bitness of the Python interpreter.

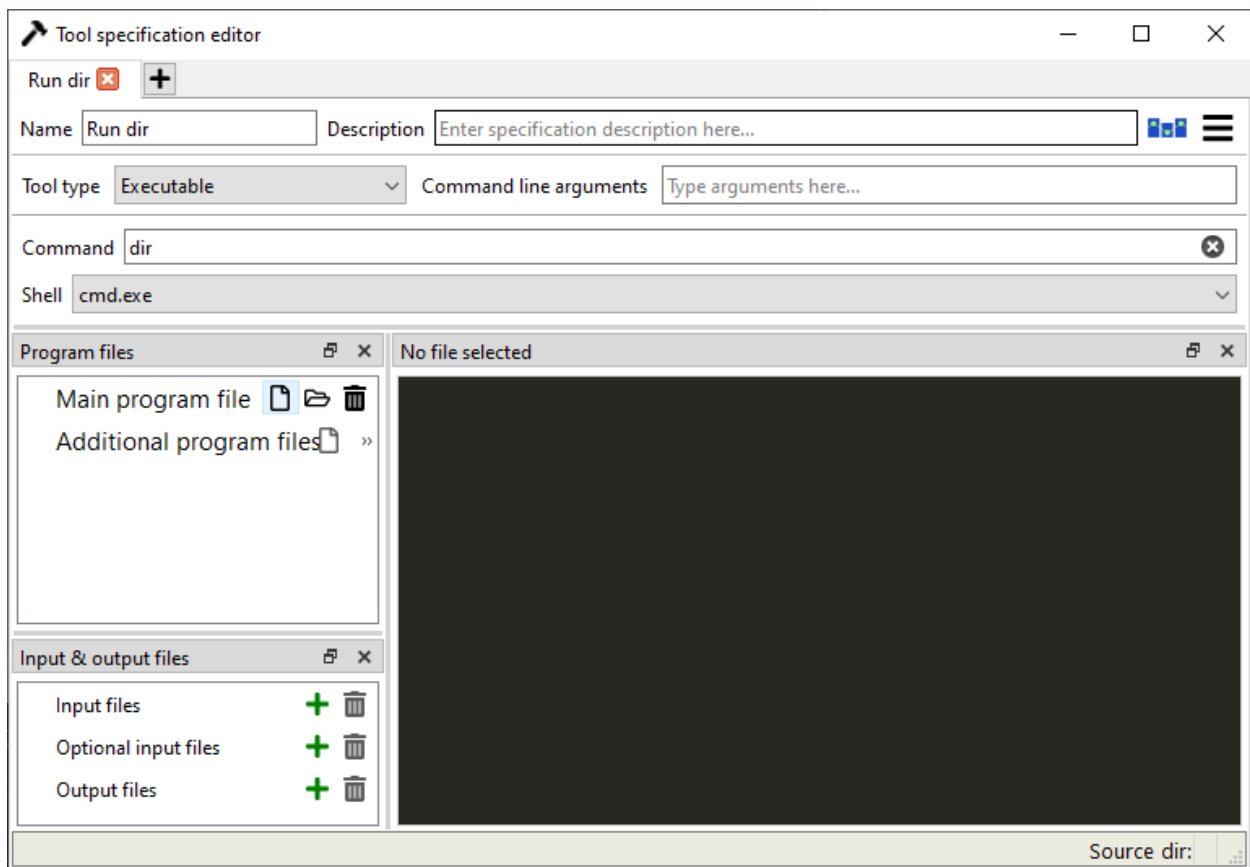
The default Gams is the Gams defined under `gams.location` in Windows registry or in your PATH environment variable. You can see the one that is currently in use from the *Tools* page in **File -> Settings**. The placeholder text shows the default Gams if found. You can also override the default Gams by setting some other gams executable path to the line edit.

4.4 Executable

Executable Tool Spec types can be used to execute virtually any program as part of a Spine Toolbox workflow. They also provide the possibility to run Shell commands as part the workflow. To run an executable with a shell you need to select a shell out of the three available options that is appropriate for your operating system. Then you can write a command that runs the executable with the arguments that it needs into the *Command* line edit just like you would on a normal shell.

To run an executable file without a shell you can either select the executable file as the main program file of the Tool and write the possible arguments into *Command line arguments* or select *no shell* and write the filepath of the executable file followed by it's arguments into the *Command* textbox. Either way the file is executed independent of a shell and with the provided arguments.

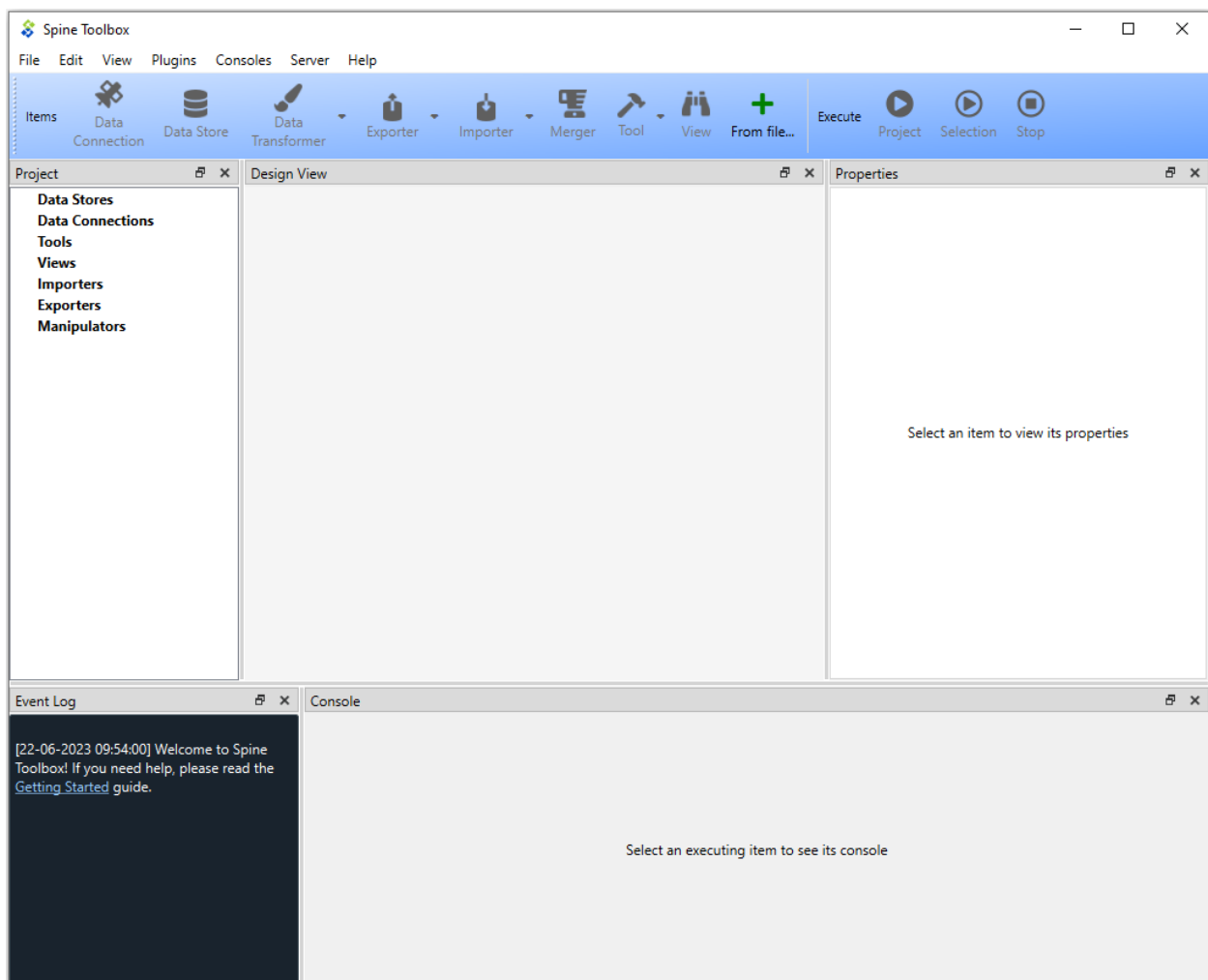
To run a Shell command, just type the command into the *command* line edit and select the appropriate Shell from the list. Picture below depicts an Executable Tool Spec that runs *dir* in in cmd.exe.



MAIN WINDOW

This section describes the different components in the application main window.

The first time you start the application you will see the main window like this.



The application main window contains four dock widgets (**Project**, **Properties**, **Event Log**, **Console**), a **Toolbar**, a **Design View**, and a menu bar with **File**, **Edit**, **View**, **Plugins**, **Consoles**, **Server** and **Help** menus. The **Project** dock widget contains a list of project items in the project. The **Properties** dock widget shows the properties of the selected project item. **Event Log** shows messages based on user actions and item executions. It shows messages from processes that are spawned by the application, i.e. it shows the stdout and stderr streams of GAMS and executable programs.

Also things like whether an item's execution was successful and when the project or an item specification is saved are shown.

Console provides Julia and Python consoles that can be interacted with. What kind of console is shown depends on the Tool type of the specific Tool. Only an item that is currently executing or has already executed shows something in this dock widget. To view an item's console, the item must be selected. When executing Python/Julia tools, the Tool's Python/Julia code will be included into the console and executed there.

Tip: You can configure the Julia and Python versions you want to use in **File -> Settings**.

The menu bar in the top of the application contains **File**, **Edit**, **View**, **Plugins**, **Consoles**, **Server** and **Help** menus. In the **File** menu you can create a new project, open an existing project, save the project or open the application Settings among other things. Spine Toolbox is project based, which means that you need to create a new project or open an existing one before you can do anything. You can create a new project by selecting **File -> New project...** from the menu bar. In the **Edit** menu you can for example copy, paste and duplicate items as well as undo and redo actions. In the **Plugins** menu you can install and manage plugins. **Consoles** menu provides a way to start detached consoles. In the **Server** menu you can retrieve projects from a server. **Help** contains a link to this documentation as well as various tidbits about Spine Toolbox.

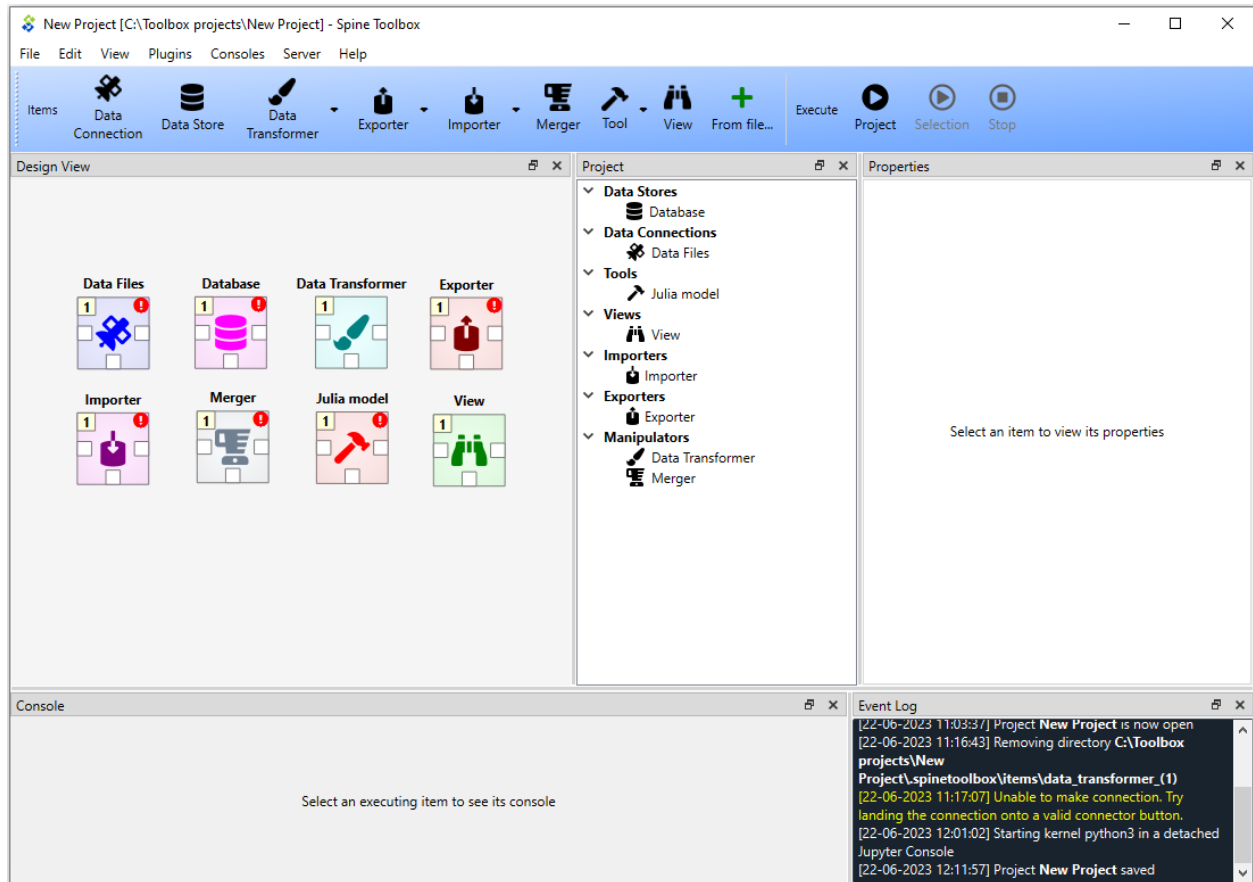
The **Items** section of the **Toolbar** contains the available *project item* types. The **Execute** section contains icons that control the execution of the items in the **Design view** where you build your project. The button executes all Directed Acyclic Graphs (DAG) in the project in a row. The button executes the selected project items only. The button terminates the execution (if running).

You can add a new project item to your project by pointing your mouse cursor on any of the draggable items in the **Toolbar**, then click-and-drag the item on to the **Design view**. After this you will be presented a dialog, which asks you to fill in basic information about the new project item (name, description, etc.).

The main window is very customizable so you can e.g. close the dock widgets that you do not need, rearrange the order of the dock widgets by dragging them around and/or resize the views to fit your needs and display size or resolution. You can find more ways to customize the visual elements of Spine Toolbox in the *settings*.



Note: If you want to restore all dock widgets to their default place use the menu item **View -> Dock Widgets -> Restore Dock Widgets**. This will show all hidden dock widgets and restore them to the main window.

Below is an example on how you can customize the main window. In the picture, a user has created a project *New Project* and created one project item from each of the eight categories. A Data Connection called *Data files*, a Data Store called *Database*, a Data Transformer called *Data Transformer*, an Exporter called *Exporter*, an Importer called *Importer*, a Merger called *Merger*, a Tool called *Julia model* and a View called *View*. The project items are also listed in the **Project** dock widget. Some of the dock widgets have also been moved from their original places.



PROJECT ITEMS


- *Project Item Properties*
- *Project Item Descriptions*
 - *Data Connection*
 - *Data Store*
 - *Data Transformer*
 - *Exporter*
 - *Importer*
 - *Merger*
 - *Tool*
 - *View*

Project items in the *Design view* and the connections between them make up the graph (Directed Acyclic Graph, DAG) that is executed when the  or  buttons are pressed.

See *Executing Projects* for more information on how a DAG is processed by Spine Toolbox. Those interested in looking under the hood can check the *Project Item Development* section.

6.1 Project Item Properties

Each project item has its own set of *properties*. You can view and edit them by selecting a project item in the **Design View**. The properties are displayed in the **Properties** dock widget on the main window. Project item properties are saved into the project save file (`project.json`), which can be found in `<proj_dir>/spinetoolbox/` directory, where `<proj_dir>` is your current project directory.

In addition, each project item has its own directory in the `<proj_dir>/spinetoolbox/items/` directory. You can quickly open the project item directory in a file explorer by clicking the  button located in the upper right corner of each **Properties** form.

6.2 Project Item Descriptions

The following items are currently available:

6.2.1 Data Connection

A Data connection item provides access to data files. The item has two categories of files: **references** connect to files anywhere on the file system or on remote (non-Spine) databases while **data** files reside in the item's own data directory.

6.2.2 Data Store

A Data store item represents a connection to a (Spine) database. Currently, the item supports sqlite and mysql dialects. The database can be accessed and modified in *Spine db editor* available by double-clicking a Data store on the Design view, from the item's properties, or from a right-click context menu.

6.2.3 Data Transformer

Data transformers set up database manipulators for successor items in a DAG. They do not transform data themselves; rather, Spine Database API does the transformations configured by Data transformers when the database is accessed. Currently supported transformations include entity class and parameter renaming as well as value transformations.

6.2.4 Exporter

Exporter outputs database data into tabulated file formats that can be consumed by Tool or be used by external software for analysis. See *Importing and Exporting Data* for more information.

6.2.5 Importer

This item provides the user a chance to define a mapping from tabulated data such as comma separated values or Excel to the Spine data model. See *Importing and Exporting Data* for more information.

6.2.6 Merger

A Merger item transfers data between Data Stores. When connected to a single source database, it simply copies data from the source to all output Data Stores. Data from more than one source gets merged to outputs.

6.2.7 Tool

Tool is the heart of a DAG. It is usually the actual model to be executed in Spine Toolbox but can be an arbitrary script, executable or system command as well. A Tool is specified by its *specification*.

6.2.8 View

A View item is meant for plotting preselected parameter values from multiple sources.

LINKS

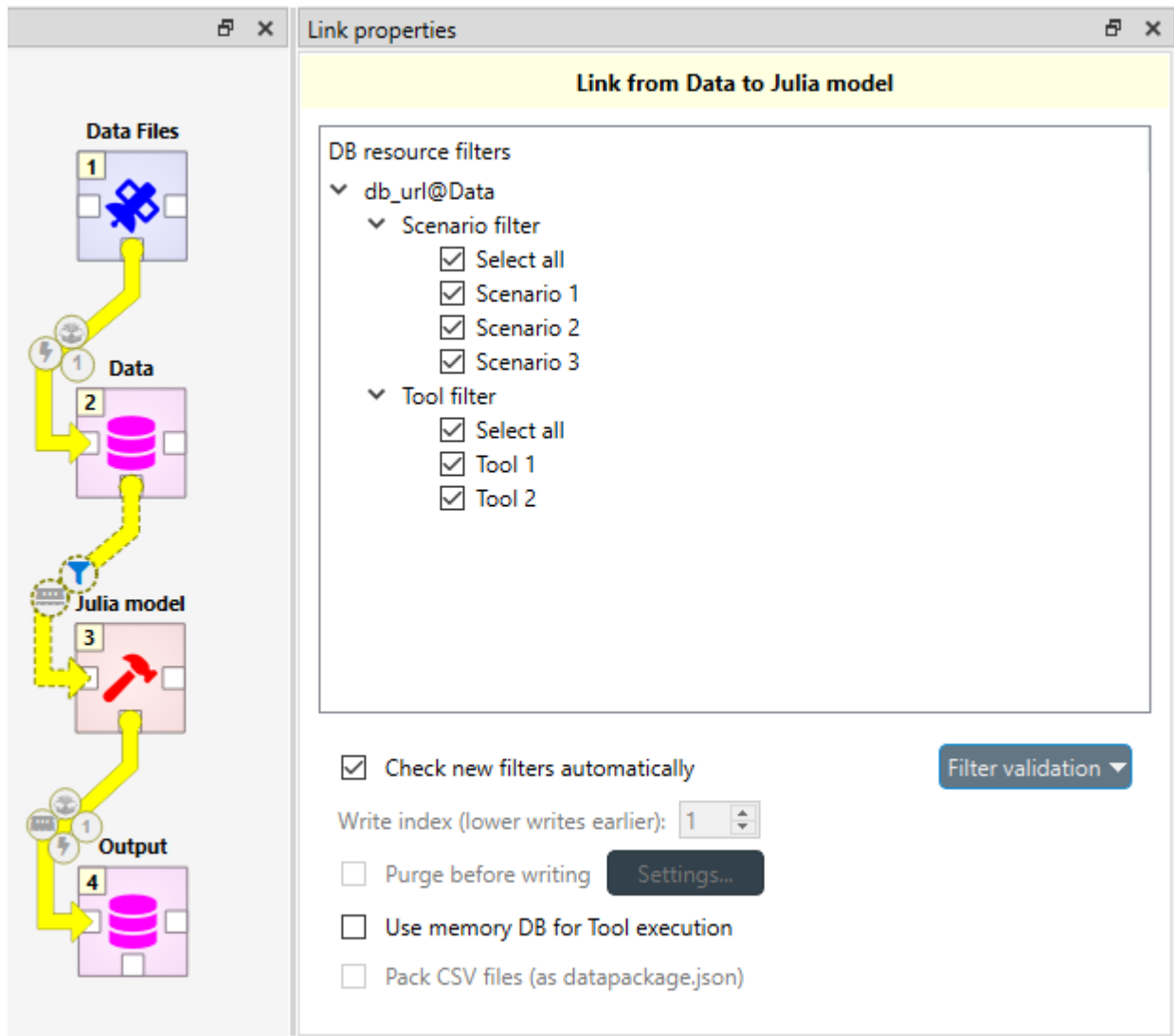
Links are the things that connect project items to each other. If Tool is the heart of a DAG, then links are the veins that connect the heart to other vital organs.

Creating a new link between items is simple. First you need to select any of the connector slots on the item where you want the link to originate from. Then select any connector slot on the item that you want to connect to. There are no limitations for how many links one connector slot can have. Like items, links can also have properties, depending on the types of items that they are connecting. When a link is selected, these properties can be modified in the **Properties** dock widget.

The small bubble icons in a link represent the state of the link's properties. When an icon is blue, the corresponding selection is active in the **Properties** dock widget.

7.1 Data Store as Source

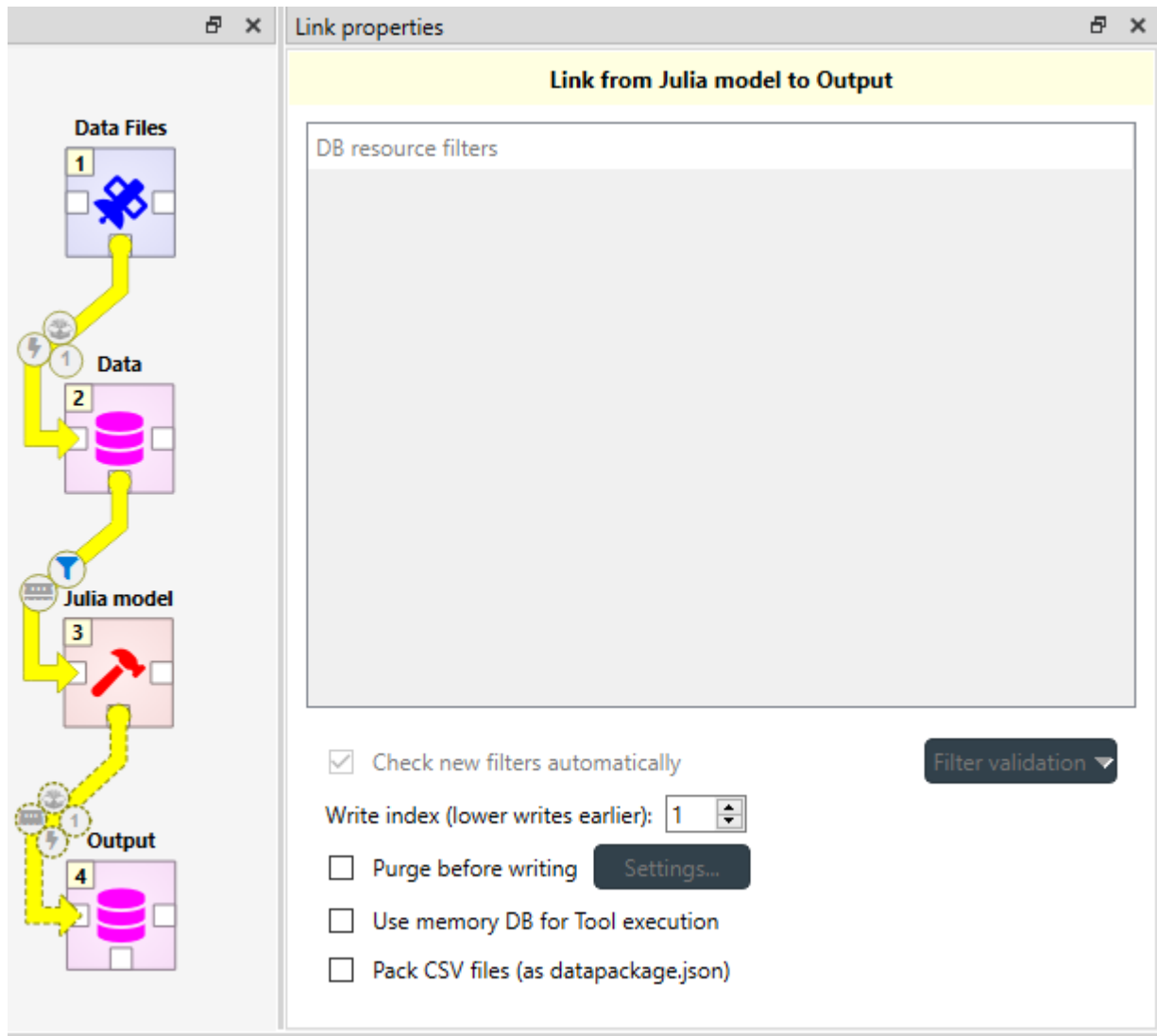
Below is an example of what the **Properties** dock widget can look like when a link originating from a Data Store is selected:



DB resource filters is divided into *Scenario* and *Tool filters*. With the scenario filters, you can select which scenarios to include in the execution. The *Tool filter* lets you choose which tools specified in the database are active. The *Check new filters automatically* option allows you to choose whether new *Scenario* and *Tool filters* added to the database should be automatically selected in this specific link. *Filter validation* allows you to force that at least one *Scenario* and/or *Tool filter* is selected at all times in that specific link.

7.2 Data Store as Destination

In the image below, the selected link ends in a Data Store. Because of this, the available selections in **Properties** differ from the previous image.



Now the link has no filters, but the write index and other various database related options become available.

Write index controls which items write to the database first. Smaller indices take precedence over larger ones while items with the same index write in an undefined order.

Purge before writing option purges the target Data Store before write operations. Click on **Settings...** to set up which items to purge.

Warning: This purge has no undo available.

7.3 Using Memory Databases

Use memory DB for Tool execution allows using a temporary in-memory database while executing a Tool which may speed up execution if the Tool accesses the database a lot.

7.4 Packing CSV files into datapackage

When the source item may provide output files, the **Pack CSV files (as datapackage.json)** option becomes enabled. This option may be handy when an item provides a lot of CSV files that e.g. need to be imported into a Data Store. Checking this option does two things:

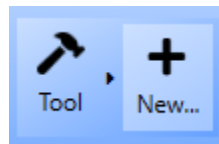
- A `datapackage.json` file is created in the common parent directory of all CSV files the source item provides. This file defines a datapackage that consists of the CSV files.
- The destination item receives only the `datapackage.json` file instead of any CSV files from the source item.

See [the datapackage specification](#) for more information on datapackages.

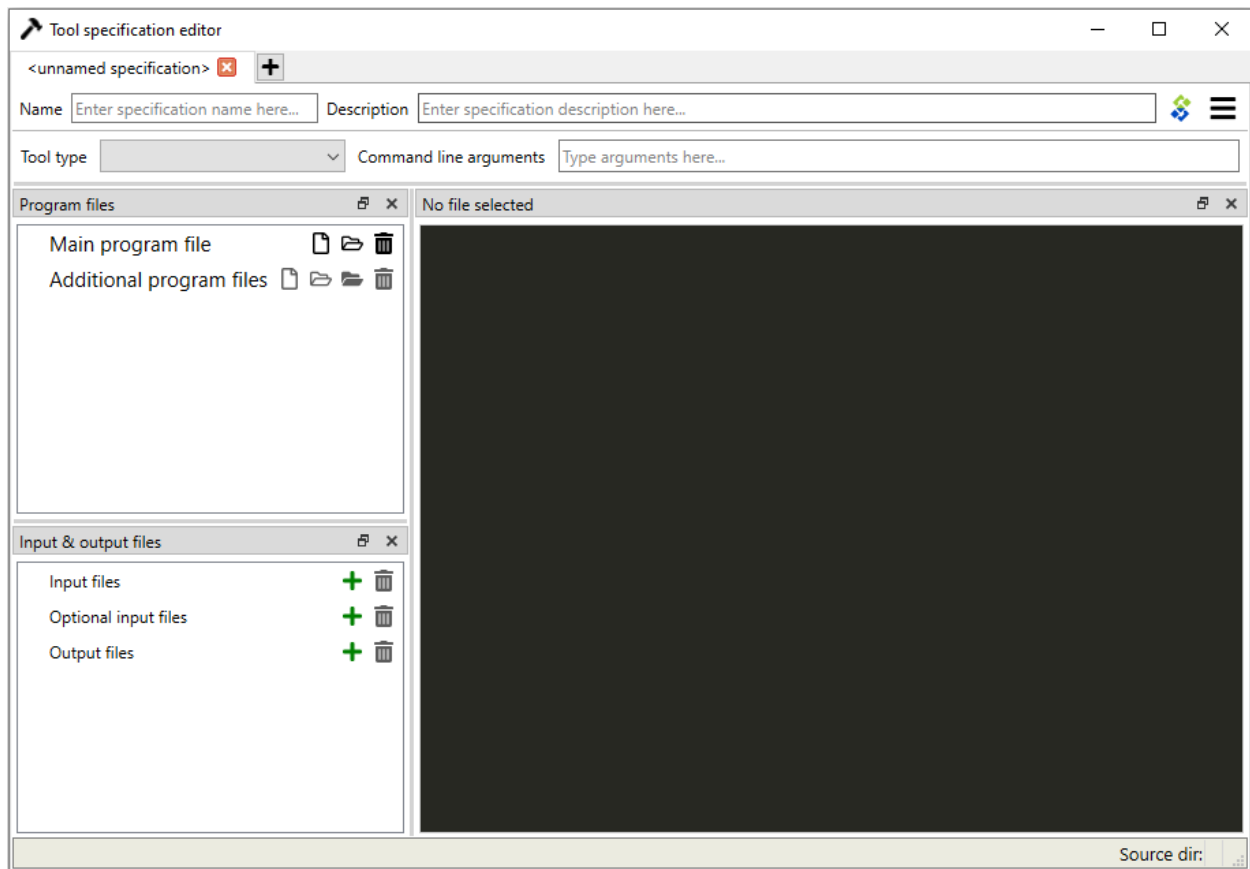
TOOL SPECIFICATION EDITOR

This section describes how to make a new Tool specification and how to edit existing Tool specifications.



To execute a Julia, Python, GAMS, or an executable script in Spine Toolbox, you must first create a Tool specification for your project. You can open the Tool specification editor in several ways. One way is to press the arrow next to the Tool icon in the toolbar to expand the Tool specifications, and then press the *New...* button.



When you press *New...* the following form pops up;




Start by giving the Tool specification a name. Then select the type of the Tool. You have four options (Julia, Python,

GAMS or Executable). You can give the Tool specification a description, describing what the Tool specification does. Main program file is the main file of your tool, i.e. a script that can be passed to Julia, Python, GAMS, or the system shell. You can create a blank file by pressing the  button, or you can browse to find an existing main program file by pressing the  button.

Command line arguments can be appended to the actual command that Spine Toolbox executes in the background. For example, you may have a Windows batch file called `do_things.bat`, which accepts command line arguments *a* and *b*. Writing *a b* on the command line arguments field in the tool specification editor is the equivalent of running the batch file in command prompt with the command `do_things.bat a b`.

Tip: Another way to pass arguments to a Tool is to write them into the *Tool arguments* drop-down list in the **Properties** dock widget. There it is possible to also rearrange existing arguments or to select available resources that are provided by other project items as arguments.

Unlike the arguments set in Tool Specification Editor, the arguments in **Properties** are *Tool specific*.

Additional source files is a list of files that the main program requires in order to run. You can add individual files the same way as with the main program file or whole directories at once by pressing the  button.

Input files is a list of input data files that the program **requires** in order to execute. You can also add directories and subdirectories. Wildcards are **not** supported (see Optional input files).

Examples:

- **data.csv** -> File is copied to the same work directory as the main program
- **input/data.csv** -> Creates directory `input/` to the work directory and copies file `data.csv` there
- **output/** -> Creates an empty directory `output/` into the work directory

Optional input files are files that may be utilized by your program if they are found. Unix-style wildcards `?` and `*` are supported.

Examples:

- **data.csv** -> If found, file is copied to the same work directory as the main program
- ***.csv** -> All found `.csv` files are copied to the same work directory as the main program
- **input/data_?.dat** -> All found files matching the pattern `data_?.dat` are copied into `input/` directory in the work directory.

Output files are files that will be archived into a timestamped result directory inside Tool's data directory after the Tool specification has finished execution. Unix-style wildcards `?` and `*` are supported.

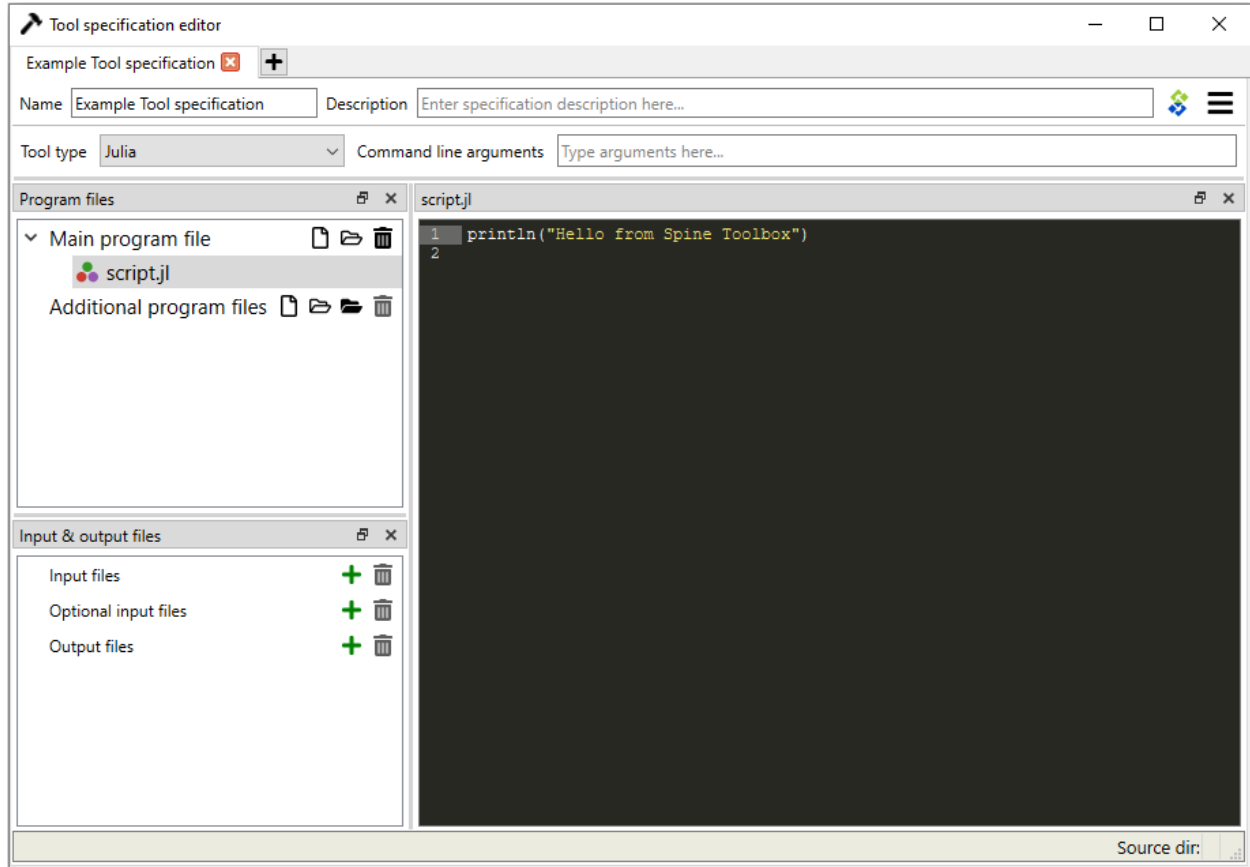
Examples:

- **results.csv** -> File is copied from work directory into results directory
- ***.csv** -> All `.csv` files from work directory are copied into results directory
- **output/*.gdx** -> All GDX files from the work directory's `output/` subdirectory will be copied into `output/` subdirectory in the results directory.

When you are happy with your Tool specification, press **Ctrl+S** to save it. You will see a message in the Event log (back in the main Spine Toolbox window), specifying the path of the saved specification file. The Tool specification file is a text file in JSON format and has an extension `.json`. You can change the location by pressing `[change]`. Also, you need to save your project for the specification to stick.

Tip: Only *name*, *type*, and either *main program file* or *command* fields are required to make a Tool specification. The other fields are optional.

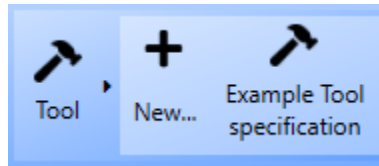
Here is a minimal Tool specification for a Julia script *script.jl*



Note: Under the hood, the contents of the Tool specification are saved to a *Tool specification file* in JSON format. Users do not need to worry about the contents of these files since reading and writing them is managed by the app. For the interested, here are the contents of the *Tool specification file* that we just created.:

```
{
  "name": "Example Tool specification",
  "tooltype": "julia",
  "includes": [
    "script.jl"
  ],
  "description": "",
  "inputfiles": [],
  "inputfiles_opt": [],
  "outputfiles": [],
  "cmdline_args": [],
  "includes_main_path": "../.."
}
```

After you have saved the specification, the new Tool specification has been added to the project.



To edit this Tool specification, just right-click on the Tool specification name and select *Edit specification* from the context-menu.

You are now ready to execute the Tool specification in Spine Toolbox. You just need to select a Tool item in the **Design view**, set the specification *Example Tool specification* for it, and click or button.

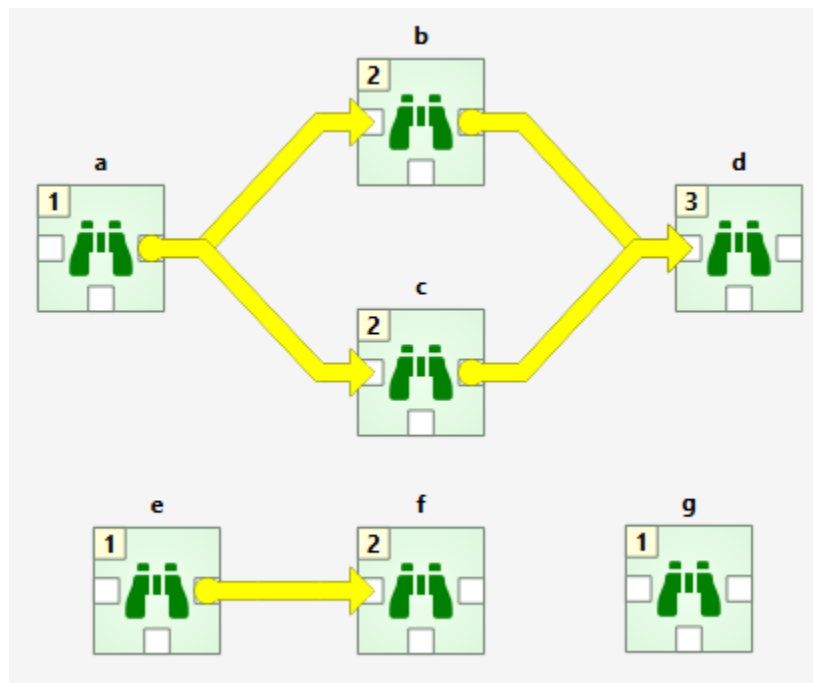
EXECUTING PROJECTS

This section describes how executing a project works and what resources are passed between project items at execution time. The buttons used to control executions are located in the **Toolbar**'s Execute -section. Execution happens by either pressing the **Project** button () to execute the whole project, or by pressing the **Selection** button () to only execute selected items. Next to these buttons is the **Stop** button (), which can be used to stop an ongoing execution. A project consists of project items and connections (yellow arrows) that are visualized on the **Design View**. You use the project items and the connections to build a **Directed Acyclic Graph (DAG)**, with the project items as *nodes* and the connections as *edges*. A DAG is traversed using the **breadth-first-search** algorithm.

Rules of DAGs:

1. A single project item with no connections is a DAG.
2. All project items that are connected, are considered as a single DAG (no matter, which direction the arrows go).
If there is a path between two items, they are considered as belonging to the same DAG.
3. Loops are not allowed (this is what acyclic means).

You can connect the nodes in the **Design View** how ever you want but you cannot execute the resulting DAGs if they break the rules above. Here is an example project with three DAGs.




- DAG 1: items: *a, b, c, d*. connections: *a-b, a-c, b-d, c-d*

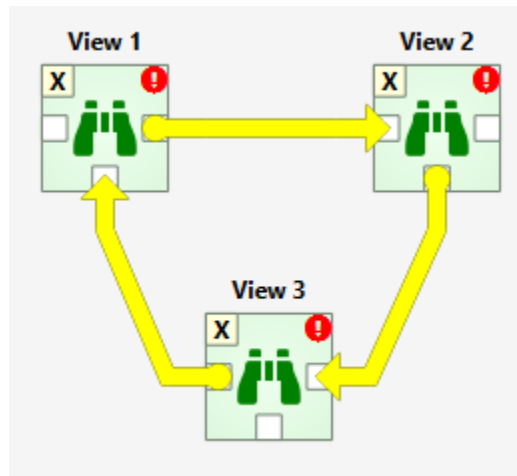
- DAG 2: items: *e, f*. connections: *e-f*
- DAG 3: items: *g*. connections: None



The numbers on the upper left corners of the icons show the item's **execution ranks** which roughly tell the order of execution within a DAG. Execution order of DAG 1 is *a->b->c->d* or *a->c->b->d* because *b* and *c* are **siblings** which is also indicated by their equal execution rank. DAG 2 execution order is *e->f* and DAG 3 is just *g*. All three DAGs are executed in a row though which DAG gets executed first is undefined. Therefore all DAGs have their execution ranks starting from 1.

We use the words **predecessor** and **successor** to refer to project items that are upstream or downstream from a project item. **Direct predecessor** is a project item that is the immediate predecessor while **Direct Successor** is a project item that is the immediate successor. For example, in the DAG 1 presented before, the successors of *a* are project items *b*, *c* and *d*. The direct successor of *b* is *d*. The predecessor of *b* is *a*, which is also its direct predecessor.

After you press the  button, you can follow the progress and the current executed item in the **Event Log**. **Design view** also animates the execution.

Items in a DAG that breaks the rules above are marked by X as their rank. Such DAGs are skipped during execution. The image below shows such a DAG where the items form a loop.

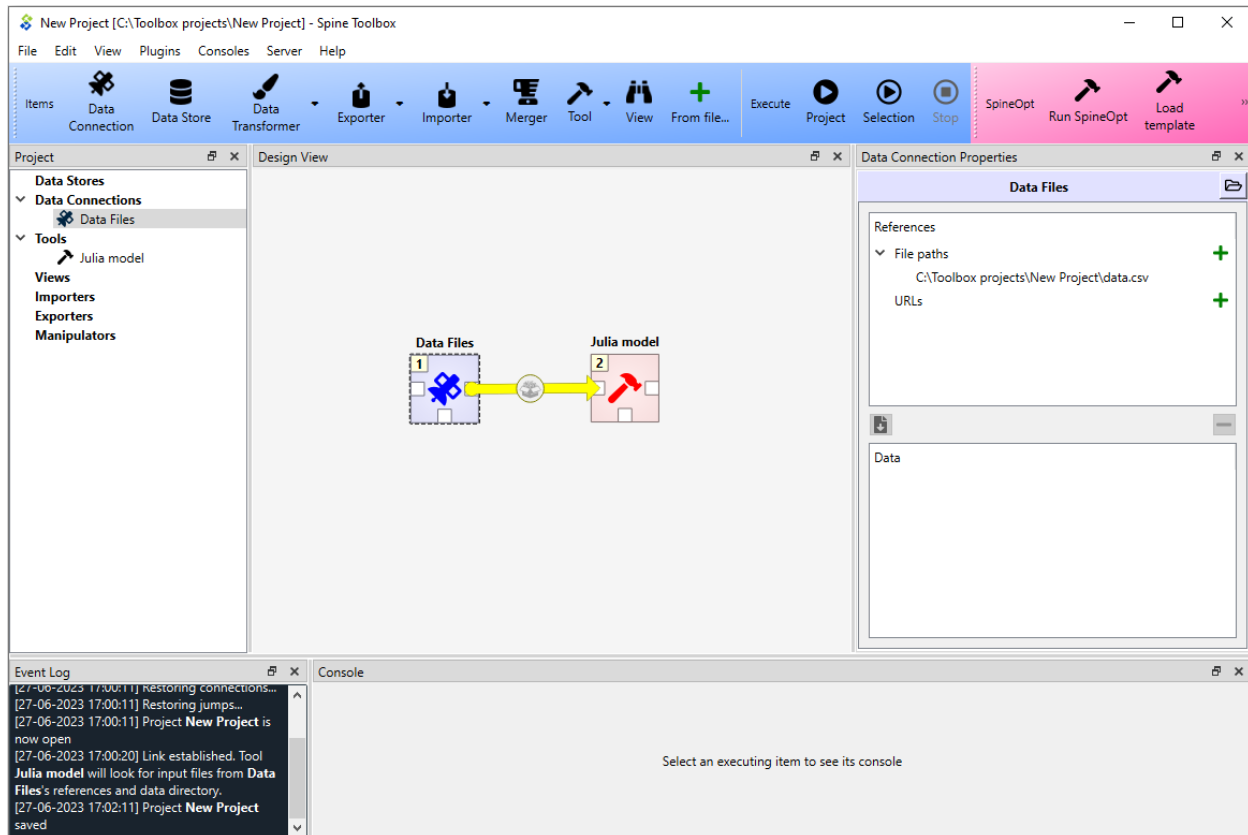


You can also execute only the selected parts of a project by multi-selecting the items you want to execute and pressing the  button in the tool bar. For example, to execute only items *b*, *d* and *f*, select the items in **Design View** or in the project item list in **Project** dock widget and then press the  button.

Tip: You can select multiple project items by holding the **Ctrl** key down and clicking on desired items or by drawing a rectangle on the **Design view**.

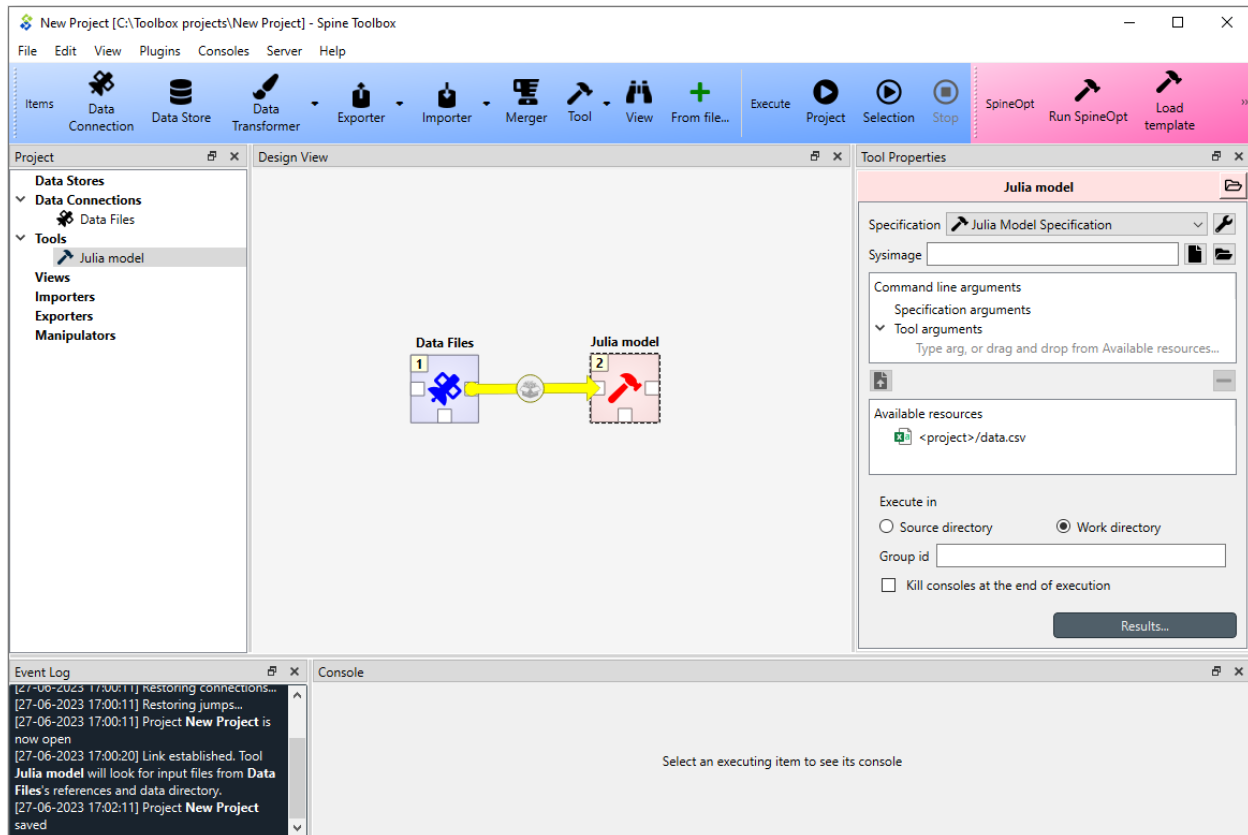
9.1 Example DAG

When you have created at least one Tool specification, you can execute a Tool as part of the DAG. The Tool specification defines the process that is executed by the Tool project item. As an example, below we have two project items; *Data Files* Data Connection and *Julia model* Tool connected to each other.




In this example, *Data Files* has a single file reference `data.csv`. Data Connections make their files visible to direct successors and thus the connection between *Data Files* and *Julia model* provides `data.csv` to the latter.

Selecting the *Julia model* shows its properties in the **Properties** dock widget.



In the top of the Tool Properties, there is **Specification** drop-down menu. From this drop-down menu, you can select the Tool specification for this particular Tool item. The *Julia Model Specification* tool specification has been selected for *Julia model*. Below the drop-down menu, you can choose a precompiled sysimage and edit Tool's command line arguments. Note that the command line argument editor already 'sees' the *data.csv* file provided by *Data Files*. The *Execute in* radio buttons control, whether this Tool is first copied to a work directory and executed there, or if the execution should happen in the source directory where the main program file is located. In *Group id*, an execution group identifier can be given. Below that, there is a checkbox with the choice to kill consoles after execution. *Results...* button opens the Tool's result archive directory in system's file browser (all Tools have their own result directory).

When you click on the  button, the execution starts from the *Data Files* Data Connection as indicated by the execution rank numbers. When executed, Data Connection items *advertise* their files and references to project items that are their direct successors. In this particular example, *data.csv* contained in *Data Files* is also a required input file in *Julia model Specification*. When it is the *Julia model* tool's turn to be executed, it checks if it finds the *data.csv* from its direct predecessor items that have already been executed. Once the input file has been found the Tool starts processing the main program file *script.jl*. Note that if the connection would be the other way around (from *Julia Model* to *Data Files*) execution would start from the *Julia model* and it would fail because it cannot find the required *data.csv*. The same thing happens if there is no connection between the two project items. In this case the project items would be in separate DAGs.

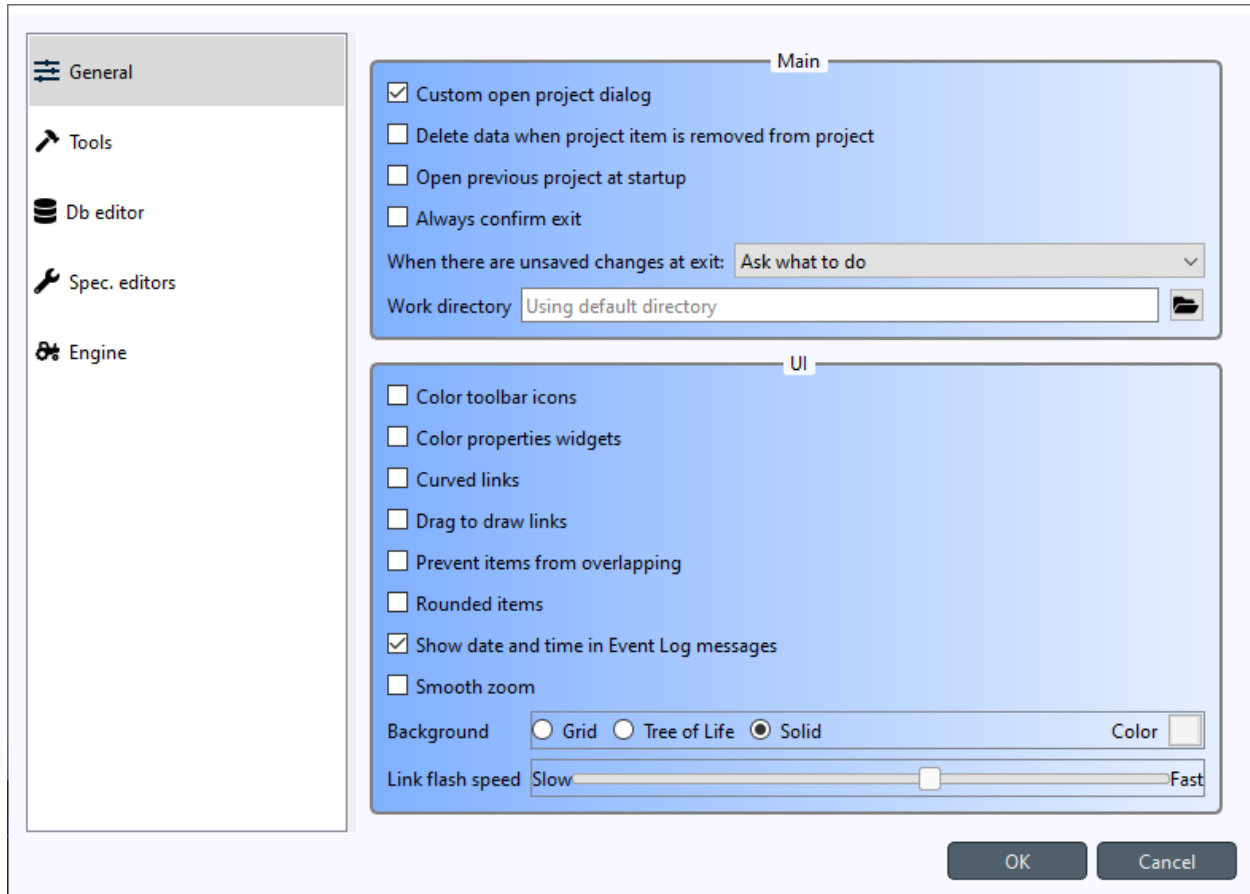
Since the Tool specification type was set as *Julia* and the main program is a Julia script, Spine Toolbox starts the execution in the Julia Console (if you have selected this in the application *Settings*, See [Settings](#) section).

SETTINGS

You can open Spine Toolbox settings from the main window menu **File -> Settings...**, or by pressing **Ctrl+,**. Settings are categorized into five tabs; *General*, *Tools*, *Db editor*, *Spec. editors* and *Engine*. In addition to application settings, each project item has user adjustable properties (See *Project Items*). See also *Setting up Consoles and External Tools* for more information on how to set up Consoles and external Tools.

- *General Settings*
- *Tools Settings*
- *Db editor Settings*
- *Spec. editor Settings*
- *Engine settings*
- *Application preferences*
- *Where are the application settings stored?*

10.1 General Settings



The General tab contains the general application settings.

Settings in the **Main** group:

- **Custom open project dialog** If checked, the application uses a special-purpose dialog when opening a project. If left unchecked, the operating system's default dialog is used.
- **Delete data when project item is removed from project** Check this box to delete project item's data when a project item is removed from project. This means that the *project item directory* and its contents will be deleted from your hard drive. You can find the project item directories from the `<proj_dir>/spinetoolbox/items/` directory, where `<proj_dir>` is your current project directory.
- **Open previous project at startup** If checked, application opens the project at startup that was open the last time the application was shut down. If left unchecked, application starts without a project open.
- **Always confirm exit** If checked, confirm exit prompt is shown always. If unchecked, application exits without prompt when there are no unsaved changes.
- **When there are unsaved changes at exit** The combo box chooses what to do with unsaved changes when the application exits.
- **Work directory** Directory where processing Tools takes place. Default place (if left empty) is shown as placeholder text. Make sure to clean up the directory every now and then.

Settings in the **UI** group:

- **Color toolbar icons** Check this box to give some color to the otherwise black toolbar icons.

- **Color properties widgets** Check this box to make the background of Project item properties more colorful.
- **Curved links** Controls the look of the arrows on **Design View**.
- **Drag to draw links** When checked, the mouse button needs to be pressed while drawing links between project items. If unchecked, single clicks at link source and destination connector slots suffices.
- **Prevent items from overlapping** When checked, other project items can be pushed away when moving an item around the **Design view**. If left unchecked, items can be piled on top of each other.
- **Rounded items** Check this box to round the corners of otherwise rectangular project items.
- **Show date and time in Event Log messages** If checked, every **Event Log** message is prepended with a date and time 'tag'.
- **Smooth zoom** Controls the way zooming (by using the mouse wheel) behaves in **Design View** and in **Spine DB Editor**. Controls if the zoom in/out is continuous or discrete. On older computers, smooth zoom is not recommended because it may be slower.
- **Background** Has some pattern options for the background of the **Design View**. Clicking on the square next to 'Color' lets you choose the pattern's color.
- **Link flash speed** This slider controls the speed of the link animation on **Design View** when execution is ongoing.

10.2 Tools Settings

The Tools tab contains settings for external tools.

Settings in the **GAMS** group:

- **GAMS executable** Set the path to GAMS executable you want to use when executing GAMS tools. If you have GAMS in your PATH environment variable, it will be automatically used. You can also choose another GAMS by clicking the button.

Settings in the **Julia** group:

Choose the settings on how Julia Tools are executed.

- **Basic Console** When selected, Julia Tools will be executed in a custom interactive Julia REPL.
- **Julia executable** Set the path to a Julia Executable used in launching the Basic Console. If Julia is in PATH this will be autofilled, but you can also choose another Julia executable.
- **Julia project** Set the Julia project you want to activate in the Basic Console.
- **Jupyter Console** Choosing this option runs Julia Tools in a custom Jupyter QtConsole embedded into Spine Toolbox.
- **Select Julia kernel... drop-down menu** Select the kernel you want to launch in Jupyter Console.

- **Make Julia Kernel** Clicking this button makes a new kernel based on the selected *Julia executable*, and *Julia project*. The progress of the operation is shown in another dialog. Installing a Julia kernel requires the **IJulia** package which will be installed to the selected *Julia project*. After **IJulia** has been installed, the kernel is installed. This process can take a couple of minutes to finish.
- **Install Julia** Installs the latest Julia on your system using the **jill** package.
- **Add/Update SpineOpt** Installs the latest compatible **SpineOpt** to the selected Julia project. If the selected *Julia project* already has SpineOpt, it is upgraded if there is a new version available.

Note: These Julia settings are *global* application settings. All Julia Tools are executed with the settings selected here.

Settings in the **Python** group:

Choose the settings on how Python Tools are executed.

- **Basic Console** When selected, Python Tools will be executed in a custom interactive Python REPL.
- **Python executable** Set the path to a Python Executable used in launching the Basic Console. The default option (if the line edit is blank) is the Python executable that was used in launching Spine Toolbox.
- **Jupyter Console** Choosing this option runs Python Tools in a custom Jupyter QtConsole embedded into Spine Toolbox.
- **Select Python kernel... drop-down menu** Select the kernel you want to launch in Jupyter Console.
- **Make Python Kernel** clicking this button makes a new kernel based on the selected *Python executable*. The progress of the operation is shown in another dialog. Installing a Python kernel (actually IPython kernel) requires the **ipykernel** package which will be installed to the selected *Python executables*. After **ipykernel** has been installed, the kernel is installed. This process can take a couple of minutes to finish.

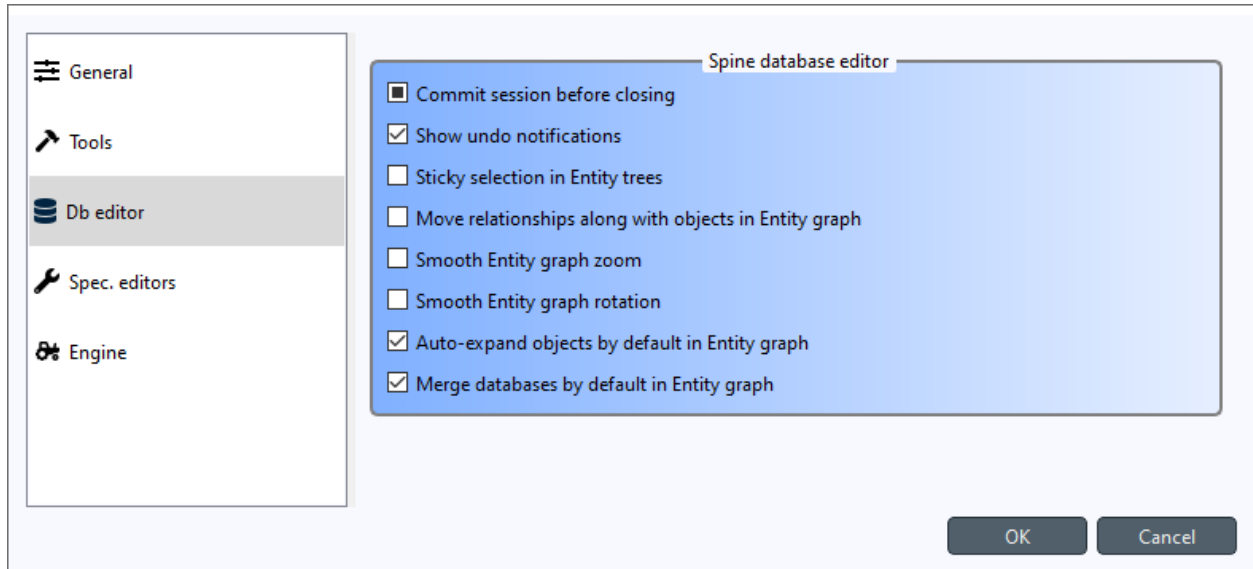
Note: These Python settings are just the default settings *for new Python Tool Specs*. You can select a specific Python kernel for each Python Tool Spec separately using the **Tool Specification Editor**.

Settings in the **Conda** group:

- **Miniconda executable** If you want to run Python Tools in a Conda environment, you can set the path to your Conda executable here.

See [Setting up Consoles and External Tools](#) for more information and examples.

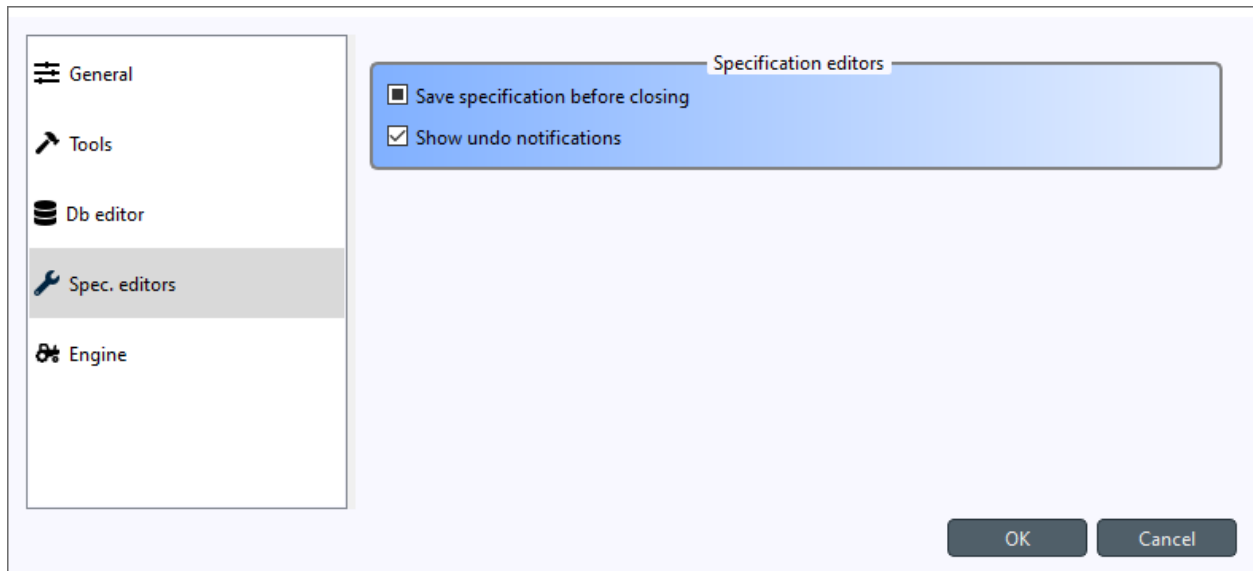
10.3 Db editor Settings



This tab contains settings for the Spine Database editor. The same settings can be accessed directly from the Database editor itself.

- **Commit session before closing** This checkbox controls what happens when you close a database editor which has uncommitted changes. When this is unchecked, all changes are discarded without notice. When this is partially checked (default), a message box warning you about uncommitted changes is shown. When this is checked, a commit message box is shown immediately without first showing the message box.
- **Show undo notifications** Checking this will show undo notification boxes in the editor every time something undoable happens. Unchecking hides the notifications.
- **Sticky selection in entity trees** Controls how selecting items in Spine database editor's Object and Relationships trees using the left mouse button works. If checked, multiple selection is enabled and pressing **Ctrl** enables single selection. If unchecked, single selection is enabled and pressing **Ctrl** enables multiple selection.
- **Move relationships along with objects in Entity graph** This controls how relationship nodes behave on the Graph view when object nodes are moved around. If checked, connected relationship nodes move along with the object node. If unchecked, connected relationship nodes remain where they are when objects nodes are moved.
- **Smooth Entity graph zoom** Checking this enables smooth zoom on the Graph view.
- **Smooth Entity graph rotation** Checking this enables smooth rotation on the Graph view.
- **Auto-expand objects by default in Entity graph** This checkbox controls which relationship nodes to show on the Graph view. If checked, all relationships that contain a visible object node are included. If unchecked, relationship nodes are included only if all their objects are show on the Graph view.
- **Merge databases by default in Entity graph** If checked, Graph view will combine all databases that are open on the same table into a single graph if they contains common object nodes. If unchecked, a separate graph will be drawn for each database.

10.4 Spec. editor Settings



The Spec. editor tab contains common settings for all specification editors.

- **Save specification before closing** If checked, specification editors will save the specification automatically at exit. If partially checked, the editors will prompt what to do explicitly. If unchecked, no prompts will be shown and all changes will be lost at exit.
- **Show undo notifications** Checking this will show undo notification boxes in the editor every time something undoable happens. Unchecking hides the notifications.

10.5 Engine settings

The screenshot shows the 'Engine' settings tab. On the left is a sidebar with icons for 'General', 'Tools', 'Db editor', 'Spec. editors', and 'Engine' (which is highlighted). The main panel contains three settings groups:

- Maximum number of concurrent processes:** Three radio buttons: 'Unlimited' (selected), 'Limit to available CPU cores', and 'User defined limit' (with a spin box set to 8).
- Maximum number of open consoles:** Three radio buttons: 'Unlimited' (selected), 'Limit to available CPU cores', and 'User defined limit' (with a spin box set to 8).
- Remote execution:** A checkbox labeled 'Enabled' is unchecked and greyed out. Below it are fields for 'Host' (text input 'Enter host name...'), 'Port' (spin box '49152'), 'Security' (dropdown menu 'None'), and 'Certs' (text input 'Select certificate directory...' with a folder icon).

At the bottom right are 'OK' and 'Cancel' buttons.

The Engine settings tab contains settings for Spine Engine.

- **Maximum number of concurrent processes** This sets a limit on how many concurrent processes the Engine can launch. *Unlimited* means that there is no upper limit. With no limits to concurrent processes the execution never stalls waiting for processes to finish. It may, however, consume all system's resources. *Limit to available CPU cores* sets the upper limit to the number of cores on the system. Finally, exact upper limit can be set by the *User defined limit* spin box.
- **Maximum number of open consoles** This sets a limit on how many concurrent Python or Julia consoles (Basic and Jupyter) there can be running at the same time. Note, that this is a separate limit from the number of concurrent processes above. *Unlimited* means that there is no upper limit. With no limits to open consoles the execution never stalls waiting for console to become free. It may, however, consume all system's resources. *Limit to available CPU cores* sets the upper limit to the number of cores on the system. Finally, exact upper limit can be set by the *User defined limit* spin box.
- **Remote execution** This group is for executing workflows on a remote Spine engine. You can find instructions on how to set it up in [Spine Engine Server](#)

10.6 Application preferences

Spine Toolbox remembers the size, location, and placement of most of the application windows from the previous session (i.e. when closing and restarting the app).

10.7 Where are the application settings stored?

Application settings and preferences (see above) are saved to a location that depends on your operating system. On Windows, they are stored into registry key `HKEY_CURRENT_USER\Software\SpineProject\Spine Toolbox`. It is safe to delete this key if you want to reset Spine Toolbox to factory defaults.

WELCOME TO SPINE DATABASE EDITOR'S USER GUIDE!

Spine database editor is a dedicated component of Spine Toolbox, that you can use to visualize and edit data in one or more Spine databases.

11.1 Spine data structure

- *Main features*
 - *Definitions*
 - *Diagram*

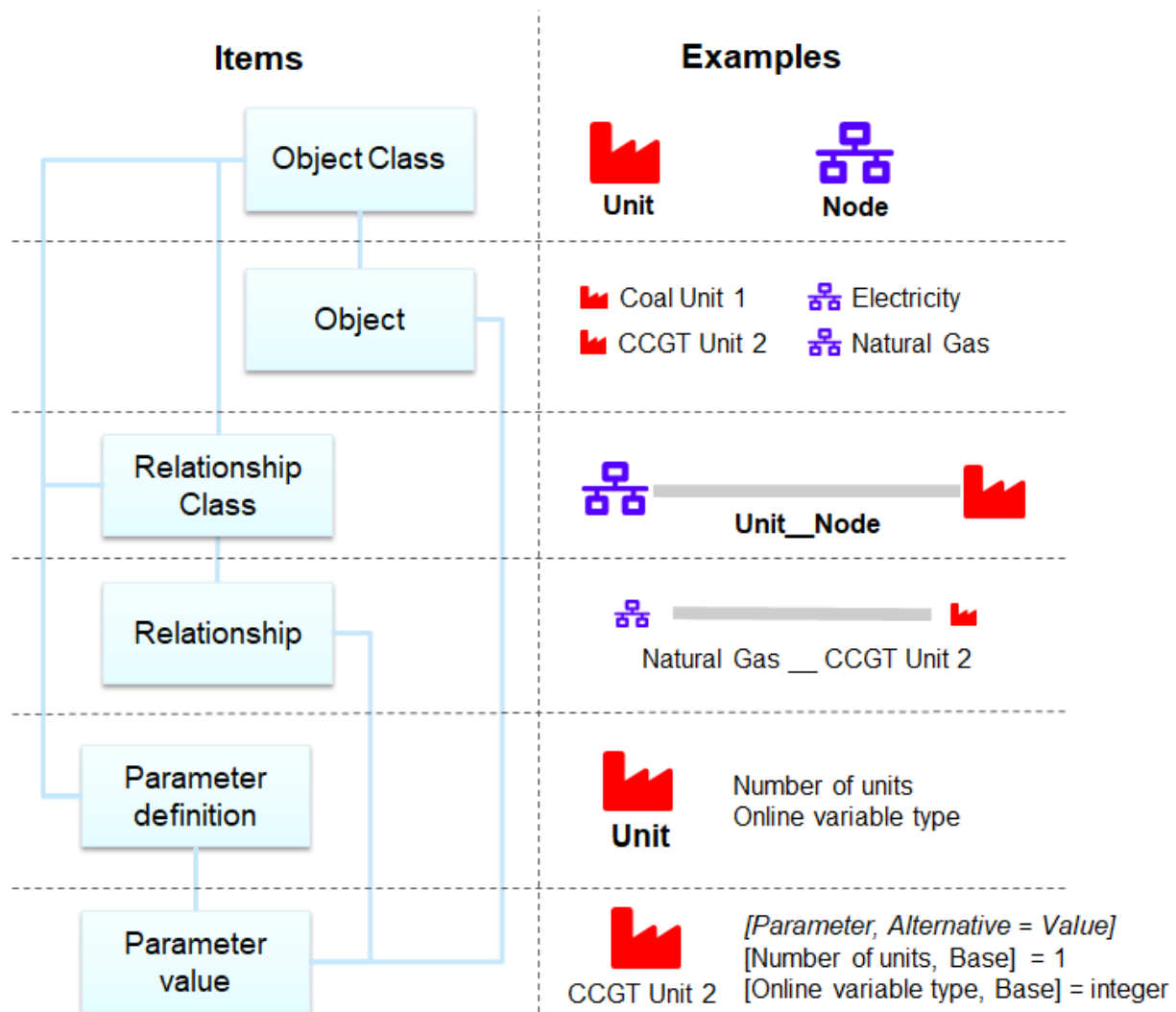
11.1.1 Main features

Spine data structure follows entity-attribute-value (EAV) with classes and relationships data model ([Wikipedia](#)). It is an open schema where the data structure is defined through data (and not through database structure). Spine Toolbox also adds an ability to hold alternative parameter values for the same parameter of a particular entity. This allows the creation of scenarios. A potential weakness of EAV is that each parameter value needs a separate row in the database which could make the parameter table large and slow. In Spine Toolbox this is circumvented by allowing different datatypes like time series and maps to be represented in the parameter field and thus greatly reducing the number of rows required to present large systems.

Definitions

1. Entity: an object (one dimension) or a relationship (n-dimensions)
2. Attribute: parameter name
3. Value: parameter value
4. Entity class: a category for entities (e.g. 'unit' is an object class while 'coal_power_plant' is an entity of 'unit' class)
5. Alternative: Each parameter value belongs to one alternative
6. Scenario: Combines alternatives into a single scenario

Diagram



11.2 Getting started

- *Launching the editor*
 - *From Spine Toolbox*
 - *From the command line*
- *Knowing the UI*

11.2.1 Launching the editor

From Spine Toolbox

To open a single database in Spine database editor:

1. Create a *Data Store* project item.
2. Select the *Data Store*.
3. Enter the url of the database in *Data Store Properties*.
4. Press the **Open editor...** button in *Data Store Properties* or double-click the *Data Store* project item.

To open multiple SQLite databases in Spine database editor:

1. Open a database in Database editor as explained above.
2. Select **Add...** from the menu.
3. Open the SQLite file.

From the command line

To open a single database in Spine database editor, use the `spine-db-editor` application which comes with Spine Toolbox:

```
spine-db-editor "...url of the database..."
```

Note that for e.g. an SQLite database, the url should start with 'sqlite:'.

11.2.2 Knowing the UI

The form has the following main UI components:

- *Entity trees (Object tree and Relationship tree)*: they present the structure of classes and entities in all databases in the shape of a tree.
- *Stacked tables (Object parameter value, Object parameter definition, Relationship parameter value, and Relationship parameter definition)*: they present object and relationship parameter data in the form of stacked tables.
- *Pivot table and Frozen table*: they present data in the form of a pivot table, optionally with frozen dimensions.
- *Entity graph*: it presents the structure of classes and entities in the shape of a graph.
- *Tool/Feature tree*: it presents tools, features, and methods defined in the databases.
- *Parameter value list*: it presents parameter value lists available in the databases.
- *Alternative tree*: it presents alternatives defined in the databases.
- *Scenario tree*: it presents scenarios defined in the databases.
- *Metadata*: presents metadata defined in the databases.
- *Item metadata*: shows metadata associated with the currently selected entities or parameter values.

Tip: You can customize the UI from the **View** and **Pivot** sections in the hamburger menu.

11.3 Viewing data

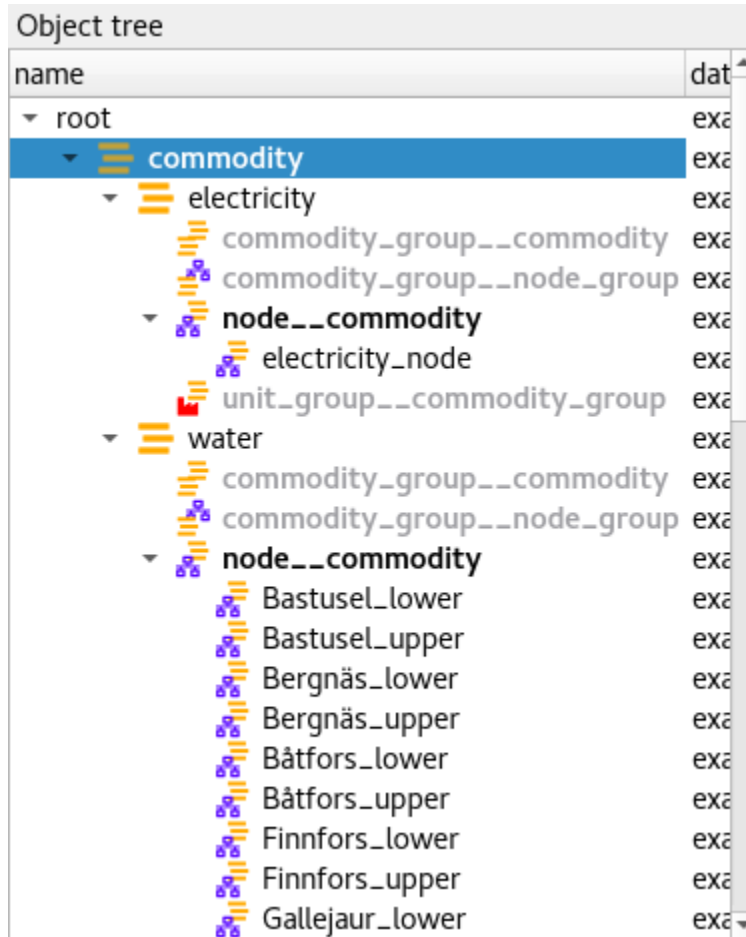
This section describes the available tools to view data.

- *Viewing entities and classes*
 - *Using Entity trees*
 - *Using Entity graph*
 - * *Building the graph*
 - * *Manipulating the graph*
- *Viewing parameter definitions and values*
 - *Using Stacked tables*
- *Viewing parameter values and relationships*
 - *Using Pivot table and Frozen table*
 - * *Selecting the input type*
 - * *Pivoting and freezing*
 - * *Filtering*
- *Viewing alternatives and scenarios*
- *Viewing scenarios*
- *Viewing tools and features*
- *Viewing parameter value lists*
- *Viewing metadata*
- *Viewing item metadata*

11.3.1 Viewing entities and classes

Using *Entity trees*

Entity trees present the structure of classes and entities in all databases in the shape of a tree:



In *Object tree*:

- To view all object classes from all databases, expand the root item (automatically expanded when loading the form).
- To view all objects of a class, expand the corresponding object class item.
- To view all relationship classes involving an object class, expand any objects of that class.
- To view all relationships of a class involving a given object, expand the corresponding relationship class item under the corresponding object item.

In *Relationship tree*:

- To view all relationship classes from all databases, expand the root item (automatically expanded when loading the form).
- To view all relationships of a class, expand the corresponding relationship class item.

Note: To expand an item in *Object tree* or *Relationship tree*, double-click on the item or press the right arrow while it's active. Items in gray don't have any children, thus they cannot be expanded. To collapse an expanded item, double-click on it again or press the left arrow while it's active.

Tip: To expand or collapse an item and all its descendants in *Object tree* or *Relationship tree*, right click on the item

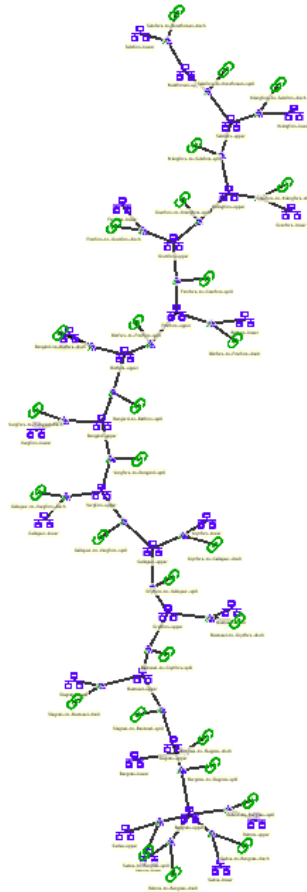
to display the context menu, and select **Fully expand** or **Fully collapse**.

Tip: In *Object tree*, the same relationship appears in many places (as many as it has dimensions). To jump to the next occurrence of a relationship item, either double-click on the item, or right-click on it to display the context menu, and select **Find next**.

Using *Entity graph*

Entity graph presents the structure of classes and entities from one database in the shape of a graph:

Entity graph



Tip: To see it in action, check out [this video](#).

Building the graph

To build the graph, select any number of items in either *Object tree* or *Relationship tree*. What is included in the graph depends on the specific selection you make:

- To include all objects and relationships from the database, select the root item in either *Object tree* or *Relationship tree*.
- To include all objects of a class, select the corresponding class item in *Object tree*.
- To include all relationships of a class, select the corresponding class item in *Relationship tree*.
- To include all relationships of a specific class involving a specific object, select the corresponding relationship class item under the corresponding object item in *Object tree*.
- To include specific objects or relationships, select the corresponding item in either *Object tree* or *Relationship tree*.

Note: In *Entity graph*, a small unnamed vertex represents a relationship, whereas a bigger named vertex represents an object. An arc between a relationship and an object indicates that the object is a member in that relationship.

The graph automatically includes relationships whenever *all* the member objects are included (even if these relationships are not selected in *Object tree* or *Relationship tree*). You can change this behavior to automatically include relationships whenever *any* of the member objects are included. To do this, enable **Auto-expand objects** via the **Graph** menu, or via *Entity graph*'s context menu.

Tip: To *extend* the selection in *Object tree* or *Relationship tree*, press and hold the **Ctrl** key while clicking on the items.

Tip: *Object tree* and *Relationship tree* also support **Sticky selection**, which allows one to extend the selection by clicking on items *without pressing Ctrl*. To enable **Sticky selection**, select **Settings** from the hamburger menu, and check the corresponding box.

Manipulating the graph

You can move items in the graph by dragging them with your mouse. By default, each items moves individually. To make relationship items move along with their member objects, select **Settings** from the hamburger menu and check the box next to *Move relationships along with objects in Entity graph*.

To display *Entity graph*'s context menu, just right-click on an empty space in the graph.

- To save the position of items into the database, select the items in the graph and choose **Save positions** from the context menu. To clear saved positions, select the items again and choose **Clear saved positions** from the context menu.
- To hide part of the graph, select the items you want to hide and choose **Hide** from context menu. To show the hidden items again, select **Show hidden** from the context menu.
- To prune the graph, select the items you want to prune and then choose **Prune entities** or **Prune classes** from the context menu. To restore specific pruned items, display the context menu, hover **Restore** and select the items you want to restore from the popup menu. To restore all pruned items at once, select **Restore all** from the context menu.
- To zoom in and out, scroll your mouse wheel over *Entity graph* or use **Zoom** buttons in the context menu.

- To rotate clockwise or anti-clockwise, press and hold the **Shift** key while scrolling your mouse wheel, or use the **Rotate** buttons in the context menu.
- To adjust the arcs' length, use the **Arc length** buttons in the context menu.
- To rebuild the graph after moving items around, select **Rebuild graph** from the context menu.
- To export the current graph as a PDF file, select **Export graph as PDF** from the context menu.

Note: *Entity graph* supports extended selection and rubber-band selection. To extend a selection, press and hold **Ctrl** while clicking on the items. To perform rubber-band selection, press and hold **Ctrl** while dragging your mouse around the items you want to select.

Note: Pruned items are remembered across graph builds.





















To display an object or relationship item's context menu, just right-click on it.

- To expand or collapse relationships for an object item, hover **Expand** or **Collapse** and select the relationship class from the popup menu.

11.3.2 Viewing parameter definitions and values

Using *Stacked tables*

Stacked tables present object and relationship parameter data from all databases in the form of stacked tables:

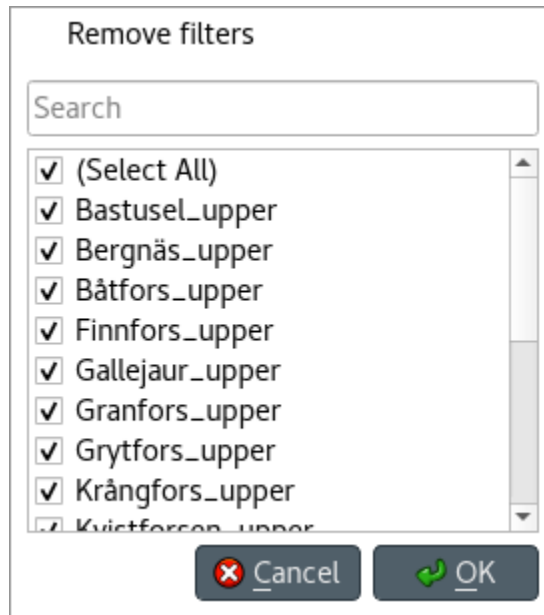
Object parameter value  				
object_class_name	object_name	parameter_name	value	database
 model	instance	duration_unit	hour	example
 model	instance	model_end	2019-01-08 00:00:00	example
 model	instance	model_start	2019-01-01 00:00:00	example
 node	Bastusel_upper	demand	-0.2579768519	example
 node	Bastusel_upper	fix_node_state	Time series	example
 node	Bastusel_upper	has_state	value_true	example
 node	Bastusel_upper	node_state_cap	8208.0	example
 node	Bergnäs_upper	demand	-22.29	example
 node	Bergnäs_upper	fix_node_state	Time series	example
 node	Bergnäs_upper	has_state	value_true	example
 node	Bergnäs_upper	node_state_cap	216120.0	example
 node	Båtfors_upper	demand	-2.0	example
 node	Båtfors_upper	fix_node_state	Time series	example
 node	Båtfors_upper	has_state	value_true	example
 node	Båtfors_upper	node_state_cap	1330.0	example
 node	Finnfors_upper	demand	0.0	example
 node	Finnfors_upper	fix_node_state	Time series	example
 node	Finnfors_upper	has_state	value_true	example

To filter *Stacked tables* by any entities and/or classes, select the corresponding items in either *Object tree*, *Relationship tree*, or *Entity graph*. To remove all these filters, select the root item in either *Object tree* or *Relationship tree*.

Stacked tables can also be filtered by selecting alternatives or scenarios from *Alternative tree* and *Scenario tree*. This filter is orthogonal to the entity/class filter and can be used together with it. To remove all these filters, select the root items or deselect all items from *Alternative tree* and *Scenario tree*.

All the filters described above can be cleared with the *Clear all filters* item available in the *Stacked tables* right-click context menu.

To apply a custom filter on a *Stacked table*, click on any horizontal header. A menu will pop up listing the items in the corresponding column:



Uncheck the items you don't want to see in the table and press **Ok**. Additionally, you can type in the search bar at the top of the menu to filter the list of items. To remove the current filter, select **Remove filters**.

To filter a *Stacked table* according to a selection of items in the table itself, right-click on the selection to show the context menu, and then select **Filter by** or **Filter excluding**. To remove these filters, select **Remove filters** from the header menus of the filtered columns.

Tip: You can rearrange columns in *Stacked tables* by dragging the headers with your mouse. The ordering will be remembered the next time you open Spine DB editor.

11.3.3 Viewing parameter values and relationships

Using *Pivot table* and *Frozen table*

Pivot table and *Frozen table* present data for an individual class from one database in the form of a pivot table, optionally with frozen dimensions:

			parameter ▸	connection_...	fix
	connection ▾	node1 ▾	node2 ▾		
	Bastusel_to_Grytfors_disch	Grytfors_upper	Bastusel_lower	1h	
	Bastusel_to_Grytfors_spill	Grytfors_upper	Bastusel_upper	150m	
	Bergnäs_to_Slagnäs_disch	Slagnäs_upper	Bergnäs_lower	1h	
	Bergnäs_to_Slagnäs_spill	Slagnäs_upper	Bergnäs_upper	1h	
	Båtfors_to_Finnfors_disch	Finnfors_upper	Båtfors_lower	3h	
	Båtfors_to_Finnfors_spill	Finnfors_upper	Båtfors_upper	3h	
	Finnfors_to_Granfors_disch	Granfors_upper	Finnfors_lower	3h	
	Finnfors_to_Granfors_spill	Granfors_upper	Finnfors_upper	3h	
	Gallejaur_to_Vargfors_disch	Vargfors_upper	Gallejaur_lower	30m	
	Gallejaur_to_Vargfors_spill	Vargfors_upper	Gallejaur_upper	150m	
	Granfors_to_Krångfors_disch	Krångfors_upper	Granfors_lower	3h	
	Granfors_to_Krångfors_spill	Krångfors_upper	Granfors_upper	3h	
	Grytfors_to_Gallejaur_disch	Gallejaur_upper	Grytfors_lower	15m	
	Grytfors_to_Gallejaur_spill	Gallejaur_upper	Grytfors_upper	15m	
	Krångfors_to_Selsfors_disch	Selsfors_upper	Krångfors_lower	3h	
	Krångfors_to_Selsfors_spill	Selsfors_upper	Krångfors_upper	3h	
	Rebnis_to_Bergnäs_disch	Bergnäs_upper	Rebnis_lower	2D	
	Rebnis_to_Bergnäs_spill	Bergnäs_upper	Rebnis_upper	2D	
	Rengård_to_Båtfors_disch	Båtfors_upper	Rengård_lower	3h	
	Rengård_to_Båtfors_spill	Båtfors_upper	Rengård_upper	3h	
	Sadva_to_Bergnäs_disch	Bergnäs_upper	Sadva_lower	2D	
	Sadva_to_Bergnäs_spill	Bergnäs_upper	Sadva_upper	2D	
	Selsfors_to_Kvistforsen_disch	Kvistforsen_upper	Selsfors_lower	3h	
	Selsfors_to_Kvistforsen_spill	Kvistforsen_upper	Selsfors_upper	3h	
	Slagnäs_to_Bastusel_disch	Bastusel_upper	Slagnäs_lower	4h	
	Slagnäs_to_Bastusel_spill	Bastusel_upper	Slagnäs_upper	4h	
	Vargfors_to_Rengård_disch	Rengård_upper	Vargfors_lower	3h	
	Vargfors_to_Rengård_spill	Rengård_upper	Vargfors_upper	2h	

To populate the tables with data for a certain class, just select the corresponding class item in either *Object tree* or *Relationship tree*.

Selecting the input type

Pivot table and *Frozen table* support four different input types:

- **Parameter value** (the default): it shows objects, parameter definitions, alternatives, and databases in the headers, and corresponding parameter values in the table body.
- **Index expansion**: Similar to the above, but it also shows parameter indexes in the headers. Indexes are extracted from special parameter values, such as time-series.
- **Relationship**: it shows objects, and databases in the headers, and corresponding relationships in the table body. It only works when selecting a relationship class in *Relationship tree*.
- **Scenario**: it shows scenarios, alternatives, and databases in the header, and corresponding *rank* in the table body.

You can select the input type from the **Pivot** section in the hamburger menu.

Note: In *Pivot table*, header blocks in the top-left area indicate what is shown in each horizontal and vertical header. For example, in **Parameter value** input type, by default, the horizontal header has two rows, listing alternative and parameter names, respectively; whereas the vertical header has one or more columns listing object names.

Pivoting and freezing

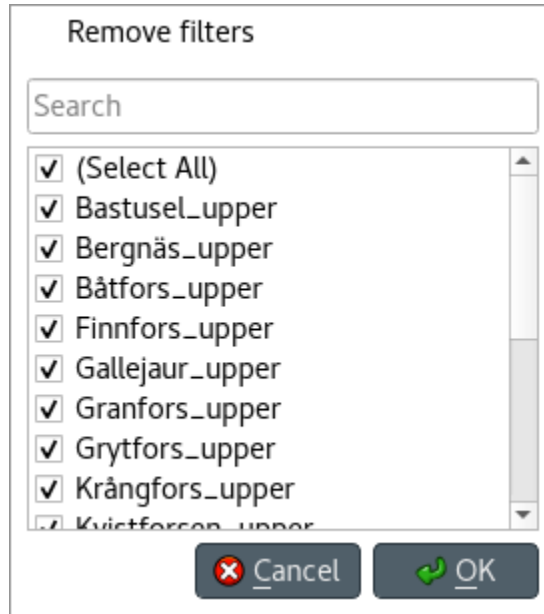
To pivot the data, drag a header block across the top-left area of the table. You can turn a horizontal header into a vertical header and vice versa, as well as rearrange headers vertically or horizontally.

To freeze a dimension, drag the corresponding header block from *Pivot table* into *Frozen table*. To unfreeze a frozen dimension, just do the opposite.

Note: Your pivoting and freezing selections for any class will be remembered when switching to another class.

Filtering

To apply a custom filter on *Pivot table*, click on the arrow next to the name of any header block. A menu will pop up listing the items in the corresponding row or column:

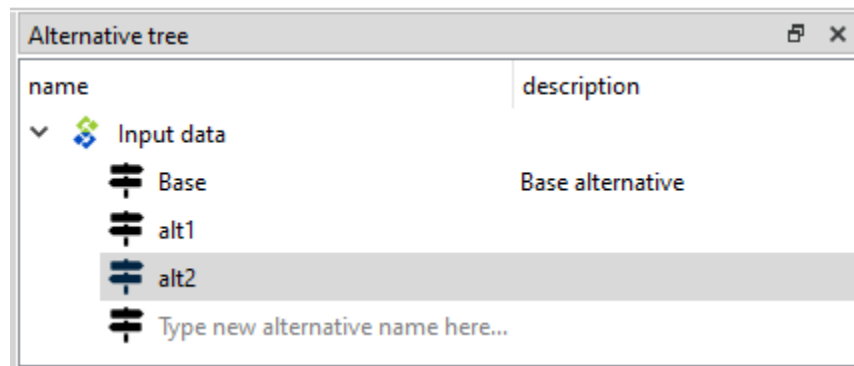


Uncheck the items you don't want to see in the table and press **Ok**. Additionally, you can type in the search bar at the top of the menu to filter the list of items. To remove the current filter, select **Remove filters**.

To filter the pivot table by an individual vector across the frozen dimensions, select the corresponding row in *Frozen table*.

11.3.4 Viewing alternatives and scenarios

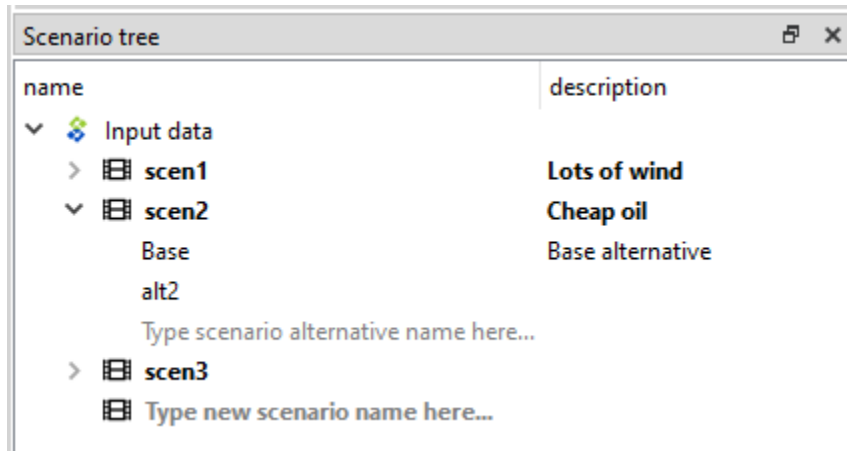
You can find alternatives from all databases under *Alternative tree*:



To view the alternatives from each database, expand the root item for that database.

11.3.5 Viewing scenarios

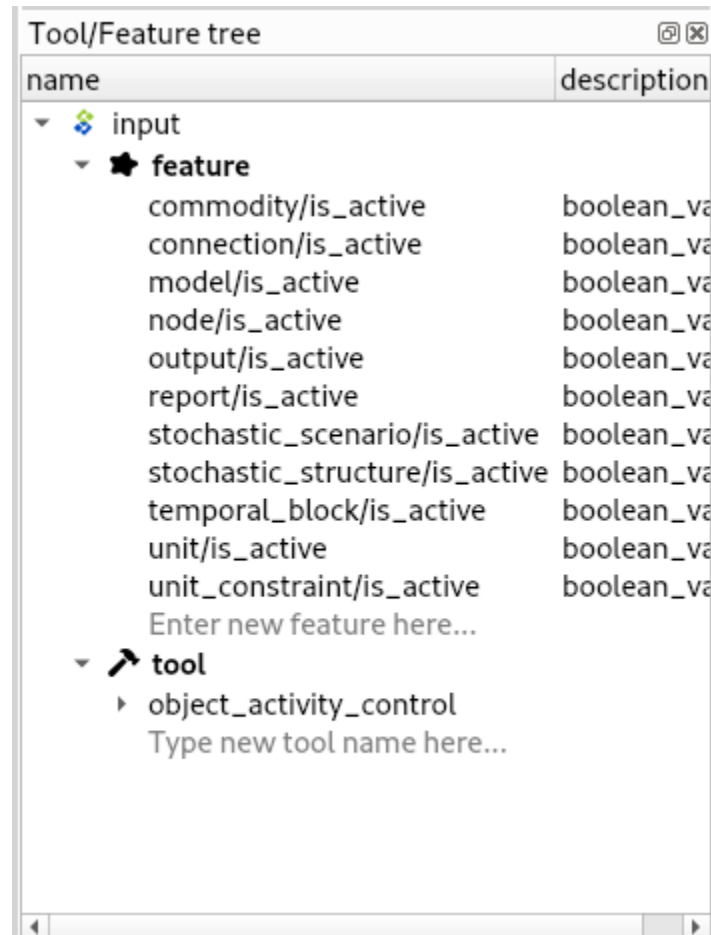
You can find scenarios from all databases under *Scenario tree*:



To view the scenarios from each database, expand the root item for that database. To view the alternatives for a particular scenario, expand the corresponding scenario item.

11.3.6 Viewing tools and features

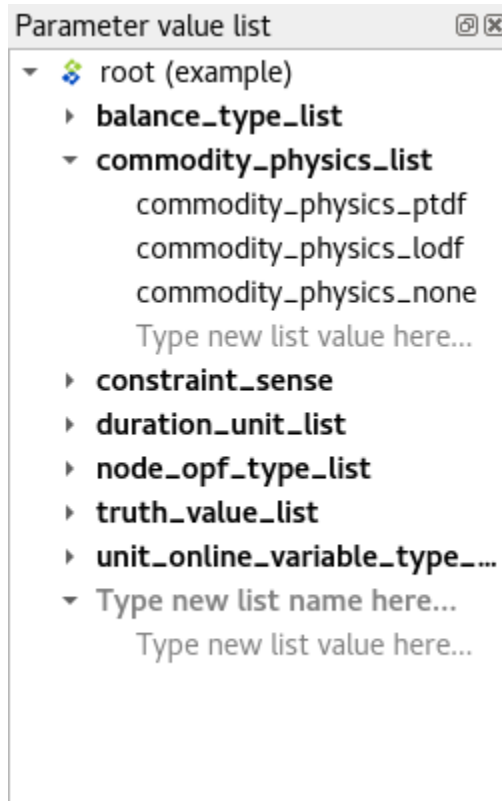
You can find tools, features, and methods from all databases under *Tool/Feature tree*:



To view the features and tools from each database, expand the root item for that database. To view all features, expand the **feature** item. To view all tools, expand the **tool** item. To view the features for a particular tool, expand the **tool_feature** item under the corresponding tool item. To view the methods for a particular tool-feature, expand the **tool_feature_method** item under the corresponding tool-feature item.

11.3.7 Viewing parameter value lists

You can find parameter value lists from all databases under *Parameter value list*:



To view the parameter value lists from each database, expand the root item for that database. To view the values for each list, expand the corresponding list item.

11.3.8 Viewing metadata

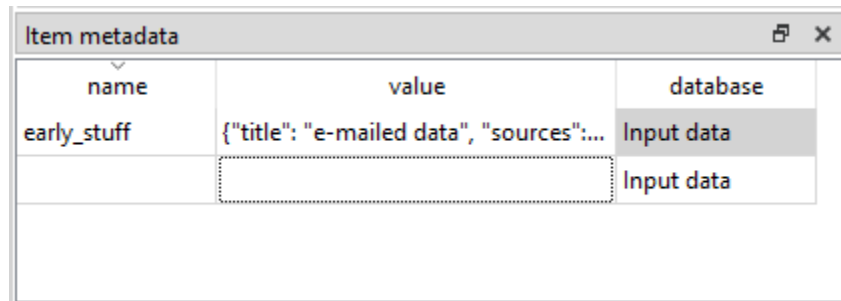
You can find metadata from all databases under *Metadata*:

Metadata		
name	value	database
early_stuff	{"title": "e-mailed data", ...	Input data
		Input data

See also *Spine Metadata Description*.

11.3.9 Viewing item metadata

You can find metadata for currently selected entities or parameter values under *Item metadata*:



name	value	database
early_stuff	{"title": "e-mailed data", "sources":...	Input data
		Input data

11.4 Adding data

This section describes the available tools to add new data.

- *Adding object classes*
 - *From Object tree*
- *Adding objects*
 - *From Object tree or Entity graph*
 - *From Pivot table*
 - *Duplicating objects*
- *Adding object groups*
- *Adding relationship classes*
 - *From Object tree or Relationship tree*
- *Adding relationships*
 - *From Object tree or Relationship tree*
 - *From Pivot table*
 - *From Entity graph*
- *Adding parameter definitions*
 - *From Stacked tables*
 - *From Pivot table*
- *Adding parameter values*
 - *From Stacked tables*
 - *From Pivot table*
- *Adding tools, features, and methods*
- *Adding alternatives*
 - *From Alternative tree*


- *From Pivot table*
- *Adding scenarios*
 - *From Scenario tree*
 - *From Pivot table*
 - *From Generate scenarios*
- *Adding parameter value lists*
- *Adding metadata and item metadata*


11.4.1 Adding object classes


From *Object tree*


Right-click on the root item in *Object tree* to display the context menu, and select **Add object classes**.

The *Add object classes* dialog will pop up:

	object class name	description	display icon	databases
1	<input type="text"/>	<input type="text"/>		example



 Cancel

 OK

Enter the names of the classes you want to add under the *object class name* column. Optionally, you can enter a description for each class under the *description* column. To select icons for your classes, double click on the corresponding cell under the *display icon* column. Finally, select the databases where you want to add the classes under *databases*. When you're ready, press **Ok**.



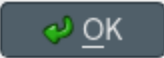
11.4.2 Adding objects

From *Object tree* or *Entity graph*

Right-click on an object class item in *Object tree*, or on an empty space in the *Entity graph*, and select **Add objects** from the context menu.

The *Add objects* dialog will pop up:

	object class name	object name	description	databases
1				example

Enter the names of the object classes under *object class name*, and the names of the objects under *object name*. To display a list of available classes, start typing or double click on any cell under the *object class name* column. Optionally, you can enter a description for each object under the *description* column. Finally, select the databases where you want to add the objects under *databases*. When you're ready, press **Ok**.

From *Pivot table*

To add an object to a specific class, bring the class to *Pivot table* using any input type (see [Using Pivot table and Frozen table](#)). Then, enter the object name in the last cell of the header corresponding to that class.

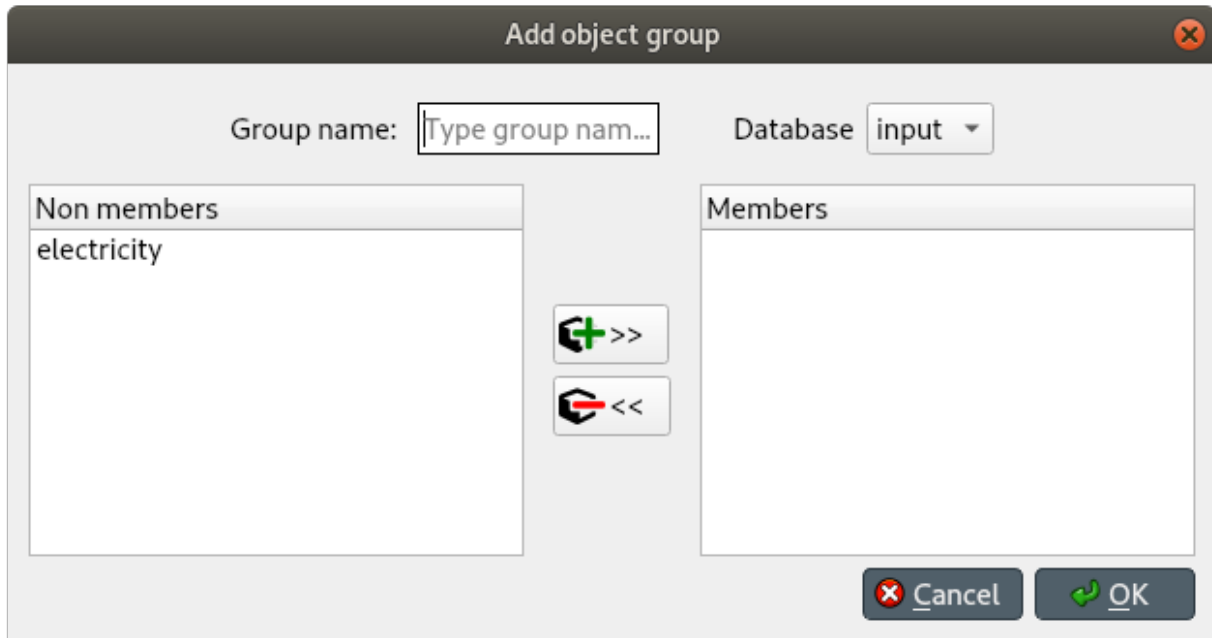
Duplicating objects

To duplicate an existing object with all its relationships and parameter values, right-click over the corresponding object item in *Object tree* to display the context menu, and select **Duplicate object**. Enter a name for the duplicate and press **Ok**.

11.4.3 Adding object groups

Right-click on an object class item in *Object tree*, and select **Add object group** from the context menu.

The *Add object group* dialog will pop up:



Enter the name of the group, and select the database where you want the group to be created. Select the member objects under *Non members*, and press the button in the middle that has a plus sign. Multiple selection works.

When you're happy with your selections, press **Ok** to add the group to the database.

11.4.4 Adding relationship classes


From *Object tree* or *Relationship tree*



Right-click on an object class item in *Object tree*, or on the root item in *Relationship tree*, and select **Add relationship classes** from the context menu.

The *Add relationship classes* dialog will pop up:

Number of dimensions

	object class name (1)	relationship class name	description	databases
1				example



Select the number of dimensions using the spinbox at the top; then, enter the names of the object classes for each dimension under each *object class name* column, and the names of the relationship classes under *relationship class name*. To display a list of available object classes, start typing or double click on any cell under the *object class name* columns. Optionally, you can enter a description for each relationship class under the *description* column. Finally, select the databases where you want to add the relationship classes under *databases*. When you're ready, press **Ok**.

11.4.5 Adding relationships


From *Object tree* or *Relationship tree*

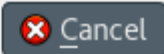

Right-click on a relationship class item either in *Object tree* or *Relationship tree*, and select **Add relationships** from the context menu.

The *Add relationships* dialog will pop up:

Relationship class

	connection	node	relationship name	databases
1				example



Select the relationship class from the combo box at the top; then, enter the names of the objects for each member object class under the corresponding column, and the name of the relationship under *relationship name*. To display a list of available objects for a member class, start typing or double click on any cell under that class's column. Finally, select the databases where you want to add the relationships under *databases*. When you're ready, press **Ok**.

From *Pivot table*

To add a relationship for a specific class, bring the class to *Pivot table* using the **Relationship** input type (see *Using Pivot table and Frozen table*). The *Pivot table* headers will be populated with all possible combinations of objects across the member classes. Locate the objects you want as members in the new relationship, and check the corresponding box in the table body.

From *Entity graph*

Make sure all the objects you want as members in the new relationship are in the graph. To start the relationship, either double click on one of the object items, or right click on it to display the context menu, and choose **Add relationships**. A menu will pop up showing the available relationship classes. Select the class you want; the mouse cursor will adopt a cross-hairs shape. Click on each of the remaining member objects, one by one and in the right order, to add them to the relationship. Once you've added enough objects for the relationship class, a dialog will pop up. Check the boxes next to the relationships you want to add, and press **Ok**.

Tip: All the *Add...* dialogs support pasting tabular (spreadsheet) data from the clipboard. Just select any cell in the table and press **Ctrl+V**. If needed, the table will grow to accommodate the exceeding data. To paste data on multiple cells, select all the cells you want to paste on and press **Ctrl+V**.

11.4.6 Adding parameter definitions

From *Stacked tables*

To add new parameter definitions for an object class, just fill the last empty row of *Object parameter definition*. Enter the name of the class under *object_class_name*, and the name of the parameter under *parameter_name*. To display a list of available object classes, start typing or double click under the *object_class_name* column. Optionally, you can also specify a default value, a parameter value list, or any number of parameter tags under the appropriate columns. The parameter is added when the background of the cells under *object_class_name* and *parameter_name* become gray.

To add new parameter definitions for a relationship class, just fill the last empty row of *Relationship parameter definition*, following the same guidelines as above.

From *Pivot table*

To add a new parameter definition for a class, bring the corresponding class to *Pivot table* using the **Parameter value** input type (see *Using Pivot table and Frozen table*). The **parameter** header of *Pivot table* will be populated with existing parameter definitions for the class. Enter a name for the new parameter in the last cell of that header.

11.4.7 Adding parameter values

From *Stacked tables*

To add new parameter values for an object, just fill the last empty row of *Object parameter value*. Enter the name of the class under *object_class_name*, the name of the object under *object_name*, the name of the parameter under *parameter_name*, and the name of the alternative under *alternative_name*. Optionally, you can also specify the parameter value right away under the *value* column. To display a list of available object classes, objects, parameters, or alternatives, just start typing or double click under the appropriate column. The parameter value is added when the background of the cells under *object_class_name*, *object_name*, and *parameter_name* become gray.

To add new parameter values for a relationship class, just fill the last empty row of *Relationship parameter value*, following the same guidelines as above.

Note: To add parameter values for an object, the object has to exist beforehand. However, when adding parameter values for a relationship, you can specify any valid combination of objects under *object_name_list*, and a relationship will be created among those objects if one doesn't yet exist.

From *Pivot table*

To add parameter value for any object or relationship, bring the corresponding class to *Pivot table* using the **Parameter value** input type (see *Using Pivot table and Frozen table*). Then, enter the parameter value in the corresponding cell in the table body.

Tip: All *Stacked tables* and *Pivot table* support pasting tabular (e.g., spreadsheet) data from the clipboard. Just select any cell in the table and press **Ctrl+V**. If needed, *Stacked tables* will grow to accommodate the exceeding data. To paste data on multiple cells, select all the cells you want to paste on and press **Ctrl+V**.

11.4.8 Adding tools, features, and methods

To add a new feature, go to *Tool/Feature tree* and select the last item under **feature** in the appropriate database, start typing or press **F2** to display available parameter definitions, and select the one you want to become a feature.

Note: Only parameter definitions that have associated a parameter value list can become features.

To add a new tool, just select the last item under **tool** in the appropriate database, and enter the name of the tool.

To add a feature for a particular tool, drag the feature item and drop it over the **tool_feature** list under the corresponding tool.

To add a new method for a tool-feature, select the last item under *tool_feature_method* (in the appropriate database), start typing or press **F2** to display available methods, and select the one you want to add.

11.4.9 Adding alternatives

From *Alternative tree*

To add a new alternative, just select the last item appropriate database, and enter the name of the alternative.

You can also copy and paste alternatives between different databases.

From *Pivot table*

Select the **Scenario** input type (see *Using Pivot table and Frozen table*). To add a new alternative, enter a name in the last cell of the **alternative** header.

11.4.10 Adding scenarios

From *Scenario tree*

To add a new scenario, just select the last item under the appropriate database, and enter the name of the scenario.

To add an alternative for a particular scenario, drag the alternative item from *Alternative tree* and drop it under the corresponding scenario. The position where you drop it determines the alternative's *rank* within the scenario. Alternatives can also be copied from *Alternative tree* and pasted at the appropriate position in *Scenario tree*.

Note: Alternatives with higher rank have priority when determining the parameter value for a certain scenario. If the parameter value is specified for two alternatives, and both of them happen to coexist in a same scenario, the value from the alternative with the higher rank takes precedence.

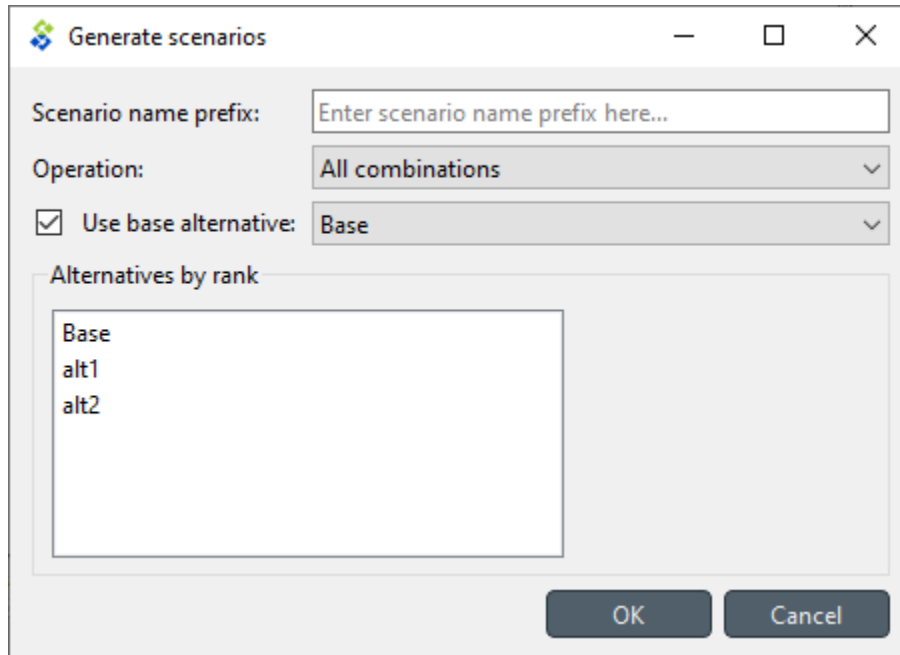
If it is desirable to base a scenario on an existing one, scenarios can be duplicated using the **Duplicate** item in the right-click context menu, or by pressing **Ctrl+D**. It is also possible to copy and paste scenarios between databases.

From *Pivot table*

Select the **Scenario** input type (see *Using Pivot table and Frozen table*). To add a new scenario, enter a name in the last cell of the **scenario** header.

From *Generate scenarios*

Scenarios can be added also by automatically generating them from existing alternatives. Select the alternatives in *Alternative tree* (using **Ctrl** and **Shift** while clicking the items), then right click to open a context menu. Select **Generate scenarios...**



Give the scenario names a prefix. An index will be appended to the prefix automatically: **prefix01**, **prefix02**,... Select appropriate operation from the combo box. Checking the **Use base alternative** check box will add the selected alternative to all generated scenarios as the lowest rank alternative. The **Alternative by rank** list allows reordering the ranks of the alternatives.

11.4.11 Adding parameter value lists

To add a new parameter value list, go to *Parameter value list* and select the last item under the appropriate database, and enter the name of the list.

To add new values for the list, select the last empty item under the corresponding list item, and enter the value. To enter a complex value, right-click on the empty item and select **Open editor** from the context menu.

Note: To be actually added to the database, a parameter value list must have at least one value.

11.4.12 Adding metadata and item metadata

To add new metadata go to *Metadata* and add a new name and value to the last row.

To add a new link metadata for an item, select an entity from one of the entity trees or a parameter value from one of the parameter value tables. Then go to *Item metadata* and select the appropriate metadata name and value on the last row.

11.5 Updating data

This section describes the available tools to update existing data.








- *Updating entities and classes*
 - *From Object tree, Relationship tree, or Entity graph*
 - *From Pivot table*
- *Updating parameter definitions and values*
 - *From Stacked tables*
 - *From Pivot table*
- *Updating alternatives*
 - *From Pivot table*
 - *From Alternative tree*
- *Updating scenarios*
 - *From Pivot table*
 - *From Scenario tree*
- *Updating tools and features*
- *Updating parameter value lists*



11.5.1 Updating entities and classes

From *Object tree*, *Relationship tree*, or *Entity graph*

Select any number of entity and/or class items in *Object tree* or *Relationship tree*, or any number of object and/or relationship items in *Entity graph*. Then, right-click on the selection and choose **Edit...** from the context menu.

One separate *Edit...* dialog will pop up for each selected entity or class type, and the tables will be filled with the current data of selected items. E.g.:

	object class name	description	display icon	databases
1	commodity	A commodity		example
2	connection	An entity where an energy transfer takes place		example
3	model			example
4	node	An entity where an energy balance takes place		example
5	output			example
6	report			example
7	temporal_block	A temporal block		example

Modify the field(s) you want under the corresponding column(s). Specify the databases where you want to update each item under the *databases* column. When you're ready, press **Ok**.

From *Pivot table*

To rename an object of a specific class, bring the class to *Pivot table* using any input type (see *Using Pivot table and Frozen table*). Then, just edit the appropriate cell in the corresponding class header.

11.5.2 Updating parameter definitions and values

From *Stacked tables*

To update parameter data, just go to the appropriate *Stacked table* and edit the corresponding row.

From *Pivot table*

To rename parameter definitions for a class, bring the corresponding class to *Pivot table* using the **Parameter value** input type (see *Using Pivot table and Frozen table*). Then, just edit the appropriate cell in the **parameter** header.

To modify parameter values for an object or relationship, bring the corresponding class to *Pivot table* using the **Parameter value** input type (see *Using Pivot table and Frozen table*). Then, just edit the appropriate cell in the table body.

11.5.3 Updating alternatives

From *Pivot table*

Select the **Scenario** input type (see *Using Pivot table and Frozen table*). To rename an alternative, just edit the proper cell in the **alternative** header.

From *Alternative tree*

To rename an alternative, just edit the appropriate item in *Alternative tree*.

11.5.4 Updating scenarios

From *Pivot table*

Select the **Scenario** input type (see *Using Pivot table and Frozen table*). To rename a scenario, just edit the proper cell in the **scenario** header.

To change the alternatives of a scenario as well as their ranks, check or uncheck the boxes on the pivot table. The number in the checkbox signifies the alternative's rank.

From *Scenario tree*

To rename a scenario, just edit the appropriate item in *Scenario tree*.

To change scenario alternative ranks, just drag and drop the items under the corresponding scenario.

11.5.5 Updating tools and features

To change a feature or method, or rename a tool, just edit the appropriate item in *Tool/Feature tree*.

11.5.6 Updating parameter value lists

To rename a parameter value list or change any of its values, just edit the appropriate item in *Parameter value list*.

11.6 Removing data

This section describes the available tools to remove data.

- *Removing entities and classes*
 - *From Object tree, Relationship tree, or Entity graph*
 - *From Pivot table*
- *Removing parameter definitions and values*
 - *From Stacked tables*
 - *From Pivot table*
- *Purging items*
- *Removing alternatives*
 - *From Pivot table*
 - *From Alternative tree*
- *Removing scenarios*

- *From Pivot table*
- *From Scenario tree*
- *Removing tools and features*
- *Removing parameter value lists*
- *Removing metadata*
- *Removing item metadata*

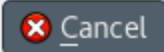

11.6.1 Removing entities and classes

From *Object tree*, *Relationship tree*, or *Entity graph*

Select the items in *Object tree*, *Relationship tree*, or *Entity graph*, corresponding to the entities and classes you want to remove. Then, right-click on the selection and choose **Remove** from the context menu.

The *Remove items* dialog will popup:

	type	name	databases
1	object	electricity	example
2	object	water	example
3	relationship class	commodity_group__commodity	example
4	relationship class	commodity_group__node_group	example
5	relationship class	node__commodity	example
6	relationship class	unit_group__commodity_group	example

Specify the databases from where you want to remove each item under the *databases* column, and press **Ok**.

From *Pivot table*

To remove objects or relationships from a specific class, bring the class to *Pivot table* using the **Parameter value** input type (see [Using Pivot table and Frozen table](#)), and select the cells in the table headers corresponding to the objects and/or relationships you want to remove. Then, right-click on the selection and choose the corresponding **Remove** option from the context menu.

Alternatively, to remove relationships for a specific class, bring the class to *Pivot table* using the **Relationship** input type (see [Using Pivot table and Frozen table](#)). The *Pivot table* headers will be populated with all possible combinations of objects across the member classes. Locate the member objects of the relationship you want to remove, and uncheck the corresponding box in the table body.

11.6.2 Removing parameter definitions and values

From *Stacked tables*

To remove parameter definitions or values, go to the relevant *Stacked table* and select any cell in the row corresponding to the items you want to remove. Then, right-click on the selection and choose the appropriate **Remove** option from the context menu.

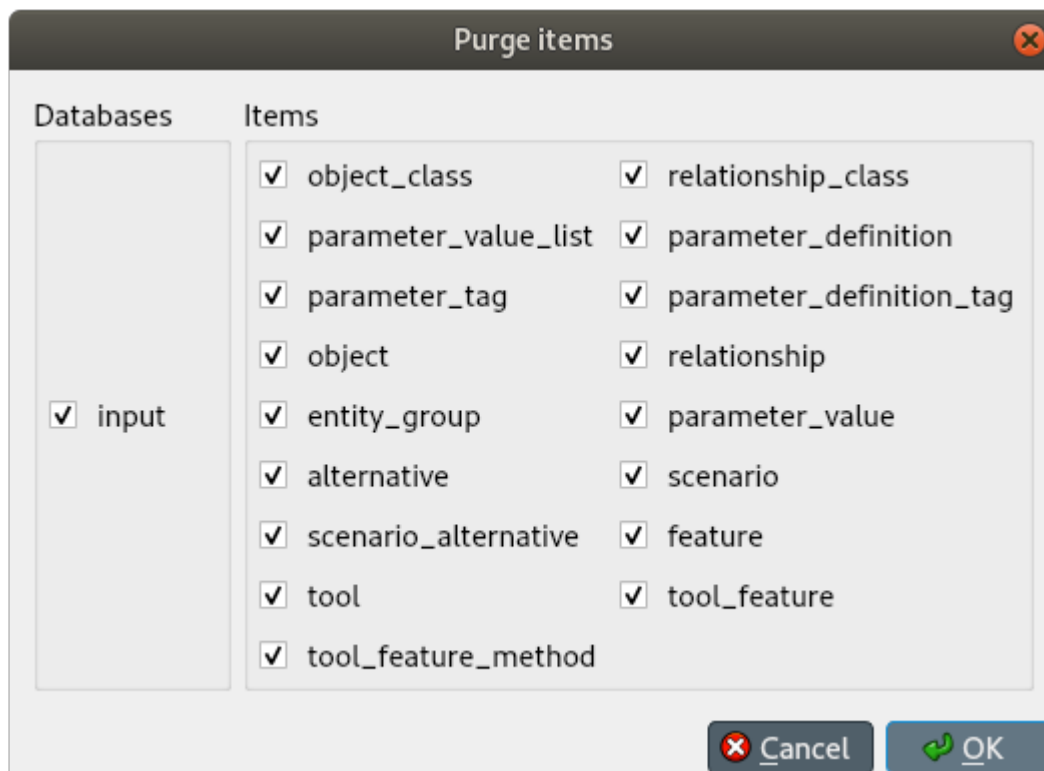
From *Pivot table*

To remove parameter definitions and/or values for a certain class, bring the corresponding class to *Pivot table* using the **Parameter value** input type (see [Using Pivot table and Frozen table](#)). Then:

1. Select the cells in the *parameter* header corresponding to the parameter definitions you want to remove, right-click on the selection and choose **Remove parameter definitions** from the context menu
2. Select the cells in the table body corresponding to the parameter values you want to remove, right-click on the selection and choose **Remove parameter values** from the context menu.

11.6.3 Purging items

To remove all items of specific types, select **Edit -> Purge** from the hamburger menu. The *Purge items* dialog will pop up:



Select the databases from where you want to remove the items under *Databases*, and the type of items you want to remove under *Items*. Then, press **Ok**.

11.6.4 Removing alternatives

From *Pivot table*

Select the **Scenario** input type (see *Using Pivot table and Frozen table*). To remove alternatives, just edit the proper cells in the **alternative** header, right-click on the selection and choose **Remove** from the context menu.

From *Alternative tree*

To remove an alternative, just select the corresponding items in *Alternative tree*, right-click on the selection and choose **Remove** from the context menu.

11.6.5 Removing scenarios

From *Pivot table*

Select the **Scenario** input type (see *Using Pivot table and Frozen table*). To remove scenarios, just select the proper cells in the **scenario** header, right-click on the selection and choose **Remove** from the context menu.

From *Scenario tree*

To remove a scenario, just select the corresponding items in *Scenario tree*, right-click on the selection and choose **Remove** from the context menu.

To remove a scenario alternative, select the corresponding alternative items in *Scenario tree*, right-click on the selection and choose **Remove** from the context menu.

11.6.6 Removing tools and features

To remove a feature, tool, or method, just select the corresponding items in *Tool/Feature tree*, right-click on the selection and choose **Remove** from the context menu.

11.6.7 Removing parameter value lists

To remove a parameter value list or any of its values, just select the corresponding items in *Parameter value list*, right-click on the selection and choose **Remove** from the context menu.

11.6.8 Removing metadata

Select the corresponding items in *Metadata*, right-click on the selection and choose **Remove row(s)** from the context menu.

11.6.9 Removing item metadata

Select the corresponding items in *Item metadata*, right-click on the selection and choose **Remove row(s)** from the context menu.

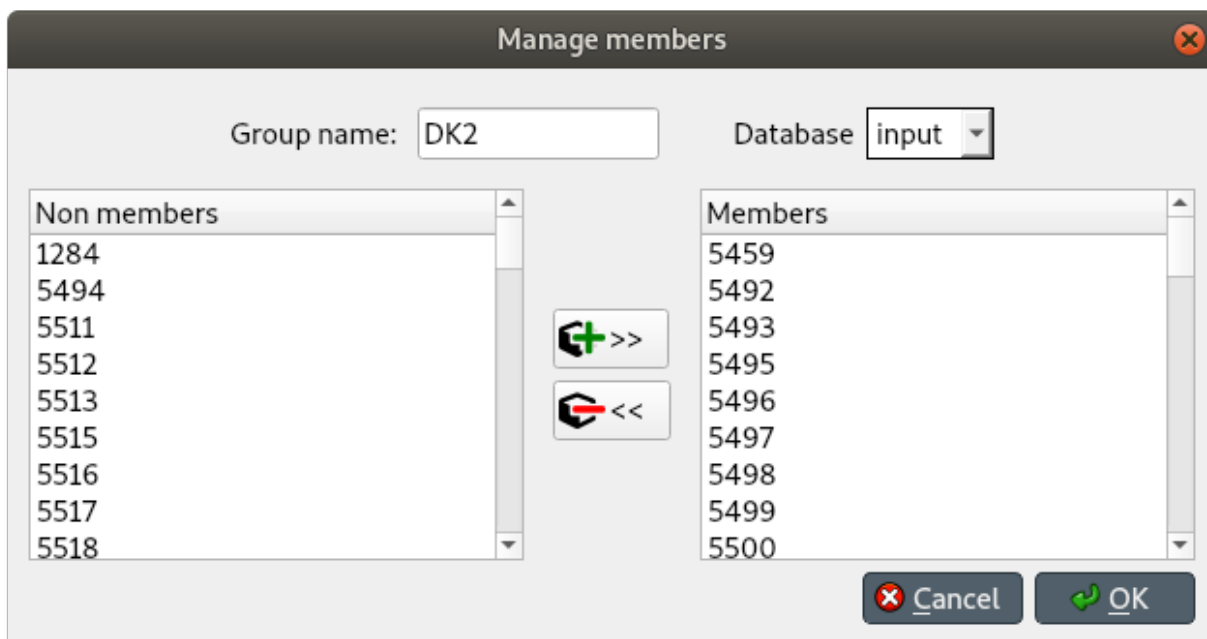
11.7 Managing data

This section describes the available tools to manage data, i.e., adding, updating or removing data at the same time.

- *Managing object groups*
- *Managing relationships*

11.7.1 Managing object groups

To modify object groups, expand the corresponding item in *Object tree* to display the **members** item, right-click on the latter and select **Manage members** from the context menu. The *Manage parameter tags* dialog will pop up:



To add new member objects, select them under *Non members*, and press the button in the middle that has a plus sign. To remove current member objects, select them under *Members*, and press the button in the middle that has a minus sign. Multiple selection works in both lists.

When you're happy, press **Ok**.

Note: Changes made using the *Manage members* dialog are not applied to the database until you press **Ok**.

11.7.2 Managing relationships

Select **Edit -> Manage relationships** from the menu bar. The *Manage relationships* dialog will pop up:

Relationship class: `connection__from_node` Database: `example`

Available objects

connection	node
Bastusel_to_Grytfors_disch	Bastusel_lower
Bastusel_to_Grytfors_spill	Bastusel_upper
Bergnäs_to_Slagnäs_disch	Bergnäs_lower
Bergnäs_to_Slagnäs_spill	Bergnäs_upper
Båtfors_to_Finnfors_disch	Båtfors_lower
Båtfors_to_Finnfors_spill	Båtfors_upper
Finnfors_to_Granfors_disch	Finnfors_lower
Finnfors_to_Granfors_spill	Finnfors_upper
Gallejaur_to_Vargfors_disch	Gallejaur_lower
Gallejaur_to_Vargfors_spill	Gallejaur_upper
Granfors_to_Krångfors_disch	Granfors_lower
Granfors_to_Krångfors_spill	Granfors_upper
Grytfors_to_Gallejaur_disch	Grytfors_lower
Grytfors_to_Gallejaur_spill	Grytfors_upper
Krångfors_to_Selsfors_disch	Krångfors_lower
Krångfors_to_Selsfors_spill	Krångfors_upper
Kvistforsen_to_downstream_disch	Kvistforsen_lower
	Kvistforsen_upper

Existing relationships

	connection	node
1	Bastusel_to_Grytfors_disch	Bastusel_lower
2	Bastusel_to_Grytfors_spill	Bastusel_upper
3	Bergnäs_to_Slagnäs_disch	Bergnäs_lower
4	Bergnäs_to_Slagnäs_spill	Bergnäs_upper
5	Båtfors_to_Finnfors_disch	Båtfors_lower
6	Båtfors_to_Finnfors_spill	Båtfors_upper
7	Finnfors_to_Granfors_disch	Finnfors_lower
8	Finnfors_to_Granfors_spill	Finnfors_upper
9	Gallejaur_to_Vargfors_disch	Gallejaur_lower
10	Gallejaur_to_Vargfors_spill	Gallejaur_upper
11	Granfors_to_Krångfors_disch	Granfors_lower
12	Granfors_to_Krångfors_spill	Granfors_upper
13	Grytfors_to_Gallejaur_disch	Grytfors_lower
14	Grytfors_to_Gallejaur_spill	Grytfors_upper
15	Krångfors_to_Selsfors_disch	Krångfors_lo...

>>

To get started, select a relationship class and a database from the combo boxes at the top.

To add relationships, select the member objects for each class under *Available objects* and press the **Add relationships** button at the middle of the form. The relationships will appear at the top of the table under *Existing relationships*.

To add multiple relationships at the same time, select multiple objects for one or more of the classes.

Tip: To *extend* the selection of objects for a class, press and hold the **Ctrl** key while clicking on more items.

Note: The set of relationships to add is determined by applying the *product* operation over the objects selected for each class.

To remove relationships, select the appropriate rows under *Existing relationships* and press the **Remove relationships** button on the right.

When you're happy with your changes, press **Ok**.

Note: Changes made using the *Manage relationships* dialog are not applied to the database until you press **Ok**.

11.8 Importing and exporting data

This section describes the available tools to import and export data.

- *Overview*
 - *Excel format*
 - *JSON format*
- *Importing*
- *Exporting*
 - *Mass export*
 - *Selective export*
 - *Session export*
- *Accessing/using exported files*

11.8.1 Overview

Spine database editor supports importing and exporting data in three different formats: SQLite, JSON, and Excel. The SQLite import/export uses the Spine database format. The JSON and Excel import/export use a specific format described below.

Tip: To create a template file with the JSON or Excel format you can simply export an existing Spine database into one of those formats.

Excel format

The Excel format consists of one sheet per object and relationship class. Each sheet can have one of four different formats:

1. Object class with scalar parameter data:

	A	B	C	D	E	F	G
1	sheet_type	entity					
2	entity_type	object					
3	class_name	node					
4	entity_dim_count	1					
5	value_type	single_value					
6	index_dim_count	0					
7							
8	node	alternative	demand	has_state	node_state_cap	state_coeff	
9	Bastusel_upper	Base	-0.2579768519	1	8208	1	
10	Bergnäs_upper	Base	-22.29	1	216120	1	
11	Båtfors_upper	Base	-2	1	1330	1	
12	Finnfors_upper	Base	0	1	300	1	
13	Gallejaure_upper	Base	15.356962963	1	3600	1	
14	Granfors_upper	Base	0	1	280	1	
15	Grytfors_upper	Base	-3.78	1	1248	1	
16	Krångfors_upper	Base	0	1	330	1	
17	Kvistforsen_upper	Base	-1.3273809524	1	1120	1	
18	Rebnis_upper	Base	-3.68	1	205560	1	
19	Rengård_upper	Base	-10.37	1	1400	1	
20	Sadva_upper	Base	-5.43	1	168000	1	
21	Selsfors_upper	Base	0	1	500	1	
22	Slagnäs_upper	Base	0	1	768	1	
23	Vargfors_upper	Base	-3.5584953704	1	4008	1	
24							
25							
26							

2. Object class with indexed parameter data:

	A	B	C	D	E	F
1	sheet_type	entity				
2	entity_type	object				
3	class_name	node				
4	entity_dim_count	1				
5	value_type	time_series				
6	index_dim_count	1				
7						
8	node	Bastusel_upper	Bergnäs_upper	Båtfors_upper	Finnfors_upper	Gallejaure_upper
9	alternative	Base	Base	Base	Base	Base
10	index	fix_node_state	fix_node_state	fix_node_state	fix_node_state	fix_node_state
11	2019-01-01T00:00:00	5581.44	114543.6	1117.2	234	1224
12	2019-01-01T01:00:00	nan	nan	nan	nan	nan
13	2019-01-07T23:00:00	5417.28	105898.8	891.1	234	2808
14						
15						

3. Relationship class with scalar parameter data:

	A	B	C	D	E
1	sheet_type	entity			
2	entity_type	relationship			
3	class_name	connection__node__node			
4	entity_dim_count	3			
5	value_type	single_value			
6	index_dim_count	0			
7					
8	connection	node	node	alternative	connection_flow_delay
9	<u>Bastusel_to_Grytfors_disch</u>	<u>Grytfors_upper</u>	<u>Bastusel_lower</u>	Base	1h
10	<u>Bastusel_to_Grytfors_spill</u>	<u>Grytfors_upper</u>	<u>Bastusel_upper</u>	Base	150m
11	<u>Bergnäs_to_Slagnäs_disch</u>	<u>Slagnäs_upper</u>	<u>Bergnäs_lower</u>	Base	1h
12	<u>Bergnäs_to_Slagnäs_spill</u>	<u>Slagnäs_upper</u>	<u>Bergnäs_upper</u>	Base	1h
13	<u>Båtfors_to_Finnfors_disch</u>	<u>Finnfors_upper</u>	<u>Båtfors_lower</u>	Base	3h
14	<u>Båtfors_to_Finnfors_spill</u>	<u>Finnfors_upper</u>	<u>Båtfors_upper</u>	Base	3h
15	<u>Finnfors_to_Granfors_disch</u>	<u>Granfors_upper</u>	<u>Finnfors_lower</u>	Base	3h
16	<u>Finnfors_to_Granfors_spill</u>	<u>Granfors_upper</u>	<u>Finnfors_upper</u>	Base	3h
17	<u>Gallejaure_to_Vargfors_disch</u>	<u>Vargfors_upper</u>	<u>Gallejaure_lower</u>	Base	30m
18	<u>Gallejaure_to_Vargfors_spill</u>	<u>Vargfors_upper</u>	<u>Gallejaure_upper</u>	Base	150m
19	<u>Granfors_to_Krångfors_disch</u>	<u>Krångfors_upper</u>	<u>Granfors_lower</u>	Base	3h

4. Relationship class with indexed parameter data:

	A	B	C	D	E	F
1	sheet_type	entity				
2	entity_type	relationship				
3	class_name	unit__from_node				
4	entity_dim_count	2				
5	value_type	time_series				
6	index_dim_count	1				
7						
8	unit	unit1	unit2	unit3	unit4	unit5
9	node	electricity_node	electricity_node	gas_node	gas_node	gas_node
10	alternative	Base	Base	Base	Base	Base
11	index	<u>vom_cost</u>	<u>vom_cost</u>	<u>vom_cost</u>	<u>vom_cost</u>	<u>vom_cost</u>
12	2019-01-01T00:00:00	-162.03	-175.56	-207.34	-178.87	-175.56
13	2019-01-01T01:00:00	-156.36	-283.11	-194.95	-174.71	-175.56
14	2019-01-01T02:00:00	-151.06	-278.76	-190.41	-168.75	-175.56
15	2019-01-01T03:00:00	-153.52	-299.57	-185.4	-172.89	-175.56
16	2019-01-01T04:00:00	-158.91	-285.28	-183.41	-172.13	-220.0
17	2019-01-01T05:00:00	-164.02	-207.34	-191.54	-171.66	-220.0
18	2019-01-01T06:00:00	-175.56	-194.95	-202.9	-173.27	-220.0
19	2019-01-01T07:00:00	-283.11	-190.41	-197.69	-176.97	-400.0
20						
21						

JSON format

The JSON format consists of a single JSON object with the following OPTIONAL keys:

- **object_classes**: the value of this key MUST be a JSON array, representing a list of object classes. Each element in this array MUST be itself a JSON array and MUST have three elements:
 - The first element MUST be a JSON string, indicating the object class name.
 - The second element MUST be either a JSON string, indicating the object class description, or null.
 - The third element MUST be either a JSON integer, indicating the object class icon code, or null.
- **relationship_classes**: the value of this key MUST be a JSON array, representing a list of relationships classes. Each element in this array MUST be itself a JSON array and MUST have three elements:
 - The first element MUST be a JSON string, indicating the relationship class name.
 - The second element MUST be a JSON array, indicating the member object classes. Each element in this array MUST be a JSON string, indicating the object class name.
 - The third element MUST be either a JSON string, indicating the relationship class description, or null.
- **parameter_value_lists**: the value of this key MUST be a JSON array, representing a list of parameter value lists. Each element in this array MUST be itself a JSON array and MUST have two elements:
 - The first element MUST be a JSON string, indicating the parameter value list name.
 - The second element MUST be a JSON array, indicating the values in the list. Each element in this array MUST be either a JSON object, string, number, or null, indicating the value.
- **object_parameters**: the value of this key MUST be a JSON array, representing a list of object parameter definitions. Each element in this array MUST be itself a JSON array and MUST have five elements:
 - The first element MUST be a JSON string, indicating the object class name.
 - The second element MUST be a JSON string, indicating the parameter name.
 - The third element MUST be either a JSON object, string, number, or null, indicating the parameter default value.
 - The fourth element MUST be a JSON string, indicating the associated parameter value list, or null.
 - The last element MUST be either a JSON string, indicating the parameter description, or null.
- **relationship_parameters**: the value of this key MUST be a JSON array, representing a list of relationship parameter definitions. Each element in this array MUST be itself a JSON array and MUST have five elements:
 - The first element MUST be a JSON string, indicating the relationship class name.
 - The second element MUST be a JSON string, indicating the parameter name.
 - The third element MUST be either a JSON object, string, number, or null, indicating the parameter default value.
 - The fourth element MUST be a JSON string, indicating the associated parameter value list, or null.
 - The last element MUST be either a JSON string, indicating the parameter description, or null.
- **objects**: the value of this key MUST be a JSON array, representing a list of objects. Each element in this array MUST be itself a JSON array and MUST have three elements:
 - The first element MUST be a JSON string, indicating the object class name.
 - The second element MUST be a JSON string, indicating the object name.
 - The third element MUST be either a JSON string, indicating the object description, or null.

- **relationships**: the value of this key **MUST** be a JSON array, representing a list of relationships. Each element in this array **MUST** be itself a JSON array and **MUST** have two elements:
 - The first element **MUST** be a JSON string, indicating the relationship class name.
 - The second element **MUST** be a JSON array, indicating the member objects. Each element in this array **MUST** be a JSON string, indicating the object name.
- **object_parameter_values**: the value of this key **MUST** be a JSON array, representing a list of object parameter values. Each element in this array **MUST** be itself a JSON array and **MUST** have four elements:
 - The first element **MUST** be a JSON string, indicating the object class name.
 - The second element **MUST** be a JSON string, indicating the object name.
 - The third element **MUST** be a JSON string, indicating the parameter name.
 - The fourth element **MUST** be either a JSON object, string, number, or null, indicating the parameter value.
- **relationship_parameter_values**: the value of this key **MUST** be a JSON array, representing a list of relationship parameter values. Each element in this array **MUST** be itself a JSON array and **MUST** have four elements:
 - The first element **MUST** be a JSON string, indicating the relationship class name.
 - The second element **MUST** be a JSON array, indicating the relationship's member objects. Each element in this array **MUST** be a JSON string, indicating the object name.
 - The third element **MUST** be a JSON string, indicating the parameter name.
 - The fourth element **MUST** be either a JSON object, string, number, or null, indicating the parameter value.

Example:

```
{
  "object_classes": [
    ["connection", "An entity where an energy transfer takes place", ↵
↵280378317271233],
    ["node", "An entity where an energy balance takes place", 280740554077951],
    ["unit", "An entity where an energy conversion process takes place", ↵
↵281470681805429],
  ],
  "relationship_classes": [
    ["connection__node__node", ["connection", "node", "node"] , null],
    ["unit__from_node", ["unit", "node"], null],
    ["unit__to_node", ["unit", "node"], null],
  ],
  "parameter_value_lists": [
    ["balance_type_list", ["\"balance_type_node\"", "\"balance_type_group\"", "\"
↵\"balance_type_none\""]],
    ["truth_value_list", ["\"value_false\"", "\"value_true\""]],
  ],
  "object_parameters": [
    ["connection", "connection_availability_factor", 1.0, null, null],
    ["node", "balance_type", "balance_type_node", "balance_type_list", null],
  ],
  "relationship_parameters": [
    ["connection__node__node", "connection_flow_delay", {"type": "duration", "data":
↵"0h"}, null, null],
    ["unit__from_node", "unit_capacity", null, null, null],
    ["unit__to_node", "unit_capacity", null, null, null],
  ],
}
```

(continues on next page)

(continued from previous page)

```

],
"objects": [
  ["connection", "Bastusel_to_Grytfors_disch", null],
  ["node", "Bastusel_lower", null],
  ["node", "Bastusel_upper", null],
  ["node", "Grytfors_upper", null],
  ["unit", "Bastusel_pwr_plant", null],
],
"relationships": [
  ["connection__node__node", ["Bastusel_to_Grytfors_disch", "Grytfors_upper",
↪ "Bastusel_lower"]],
  ["unit__from_node", ["Bastusel_pwr_plant", "Bastusel_upper"]],
  ["unit__to_node", ["Bastusel_pwr_plant", "Bastusel_lower"]],
],
"object_parameter_values": [
  ["node", "Bastusel_upper", "demand", -0.2579768519],
  ["node", "Bastusel_upper", "fix_node_state", {"type": "time_series", "data": {
↪ "2018-12-31T23:00:00": 5581.44, "2019-01-07T23:00:00": 5417.28}}],
  ["node", "Bastusel_upper", "has_state", "value_true"],
],
"relationship_parameter_values": [
  ["connection__node__node", ["Bastusel_to_Grytfors_disch", "Grytfors_upper",
↪ "Bastusel_lower"], "connection_flow_delay", {"type": "duration", "data": "1h"}],
  ["unit__from_node", ["Bastusel_pwr_plant", "Bastusel_upper"], "unit_capacity", ↪
↪ 127.5],
]
}

```

11.8.2 Importing

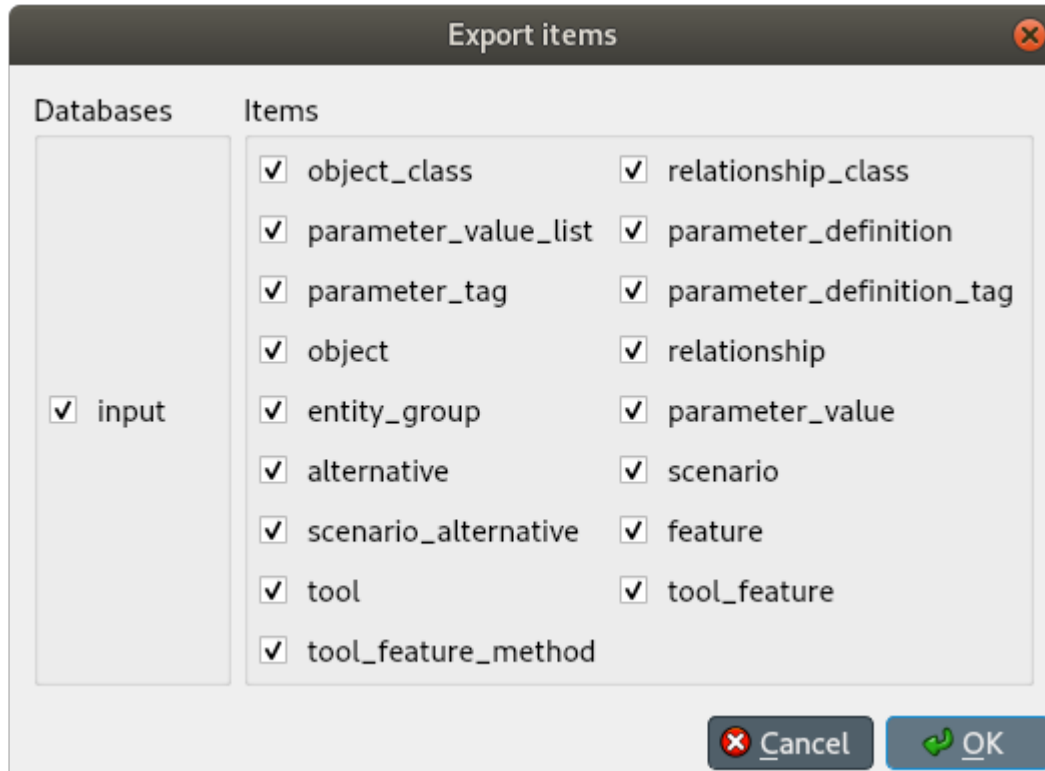
To import a file, select **File → Import** from the hamburger menu. The *Import file* dialog will pop up. Select the file type (SQLite, JSON, or Excel), enter the path of the file to import, and accept the dialog.

Tip: You can undo import operations using **Edit → Undo**.

11.8.3 Exporting

Mass export

To export items in mass, select **File → Export** from the hamburger menu. The *Export items* dialog will pop up:



Select the databases you want to export under *Databases*, and the type of items under *Items*, then press **Ok**. The *Export file* dialog will pop up now. Select the file type (SQLite, JSON, or Excel), enter the path of the file to export, and accept the dialog.

Selective export

To export a specific subset of items, select the corresponding items in either *Object tree* and *Relationship tree*, right click on the selection to bring the context menu, and select **Export**.

The *Export file* dialog will pop up. Select the file type (SQLite, JSON, or Excel), enter the path of the file to export, and accept the dialog.

Session export

To export only uncommitted changes made in the current session, select **File -> Export session** from the hamburger menu.

The *Export file* dialog will pop up. Select the file type (SQLite, JSON, or Excel), enter the path of the file to export, and accept the dialog.

Note: Export operations include all uncommitted changes.

11.8.4 Accessing/using exported files

Whenever you successfully export a file, a button with the file name is created in the *Exports* bar at the bottom of the form. To open the file in your registered program, press that button. To open the containing folder, click on the arrow next to the file name and select **Open containing folder** from the popup menu.

11.9 Committing and rolling back

Note: Changes are not immediately saved to the database(s). They need to be committed separately.

To commit your changes, select **Session -> Commit** from the hamburger menu, enter a commit message and press **Commit**. Any changes made in the current session will be saved into the database.

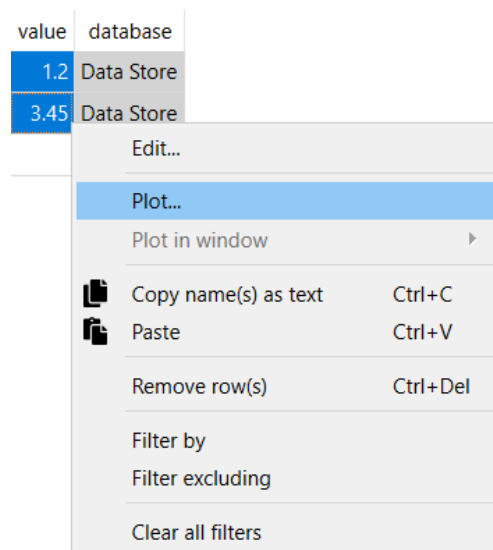
To undo *all* changes since the last commit, select **Session -> Rollback** from the hamburger menu.

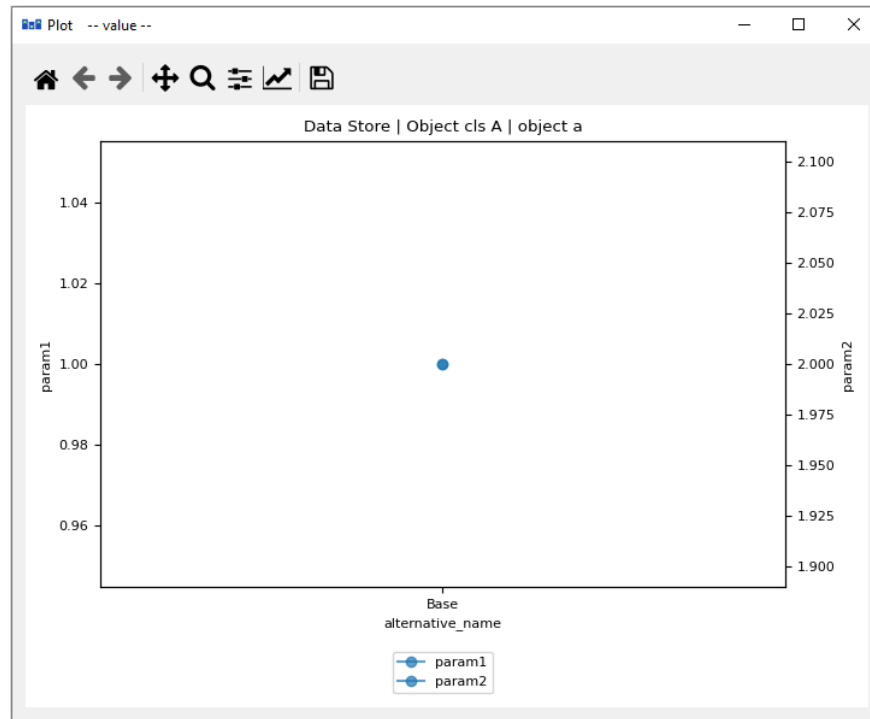
Tip: To undo/redo individual changes, use the **Edit -> Undo** and **Edit -> Redo** actions from the hamburger menu.

PLOTTING

Basic data visualization is available in the Spine database editors. Currently, it is possible to plot scalar values as well as time series, arrays and one dimensional maps with some limitations.

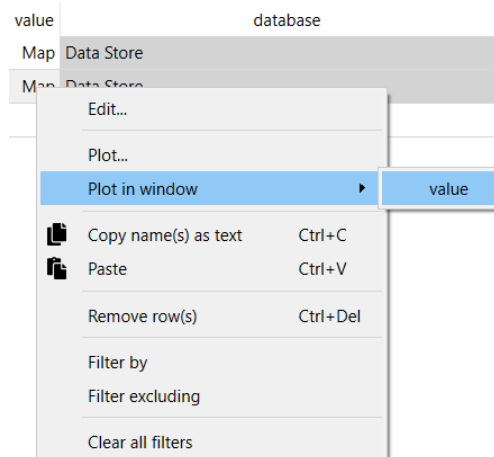
To plot a column, select the values from a table and then *Plot* from the **right click** popup menu.

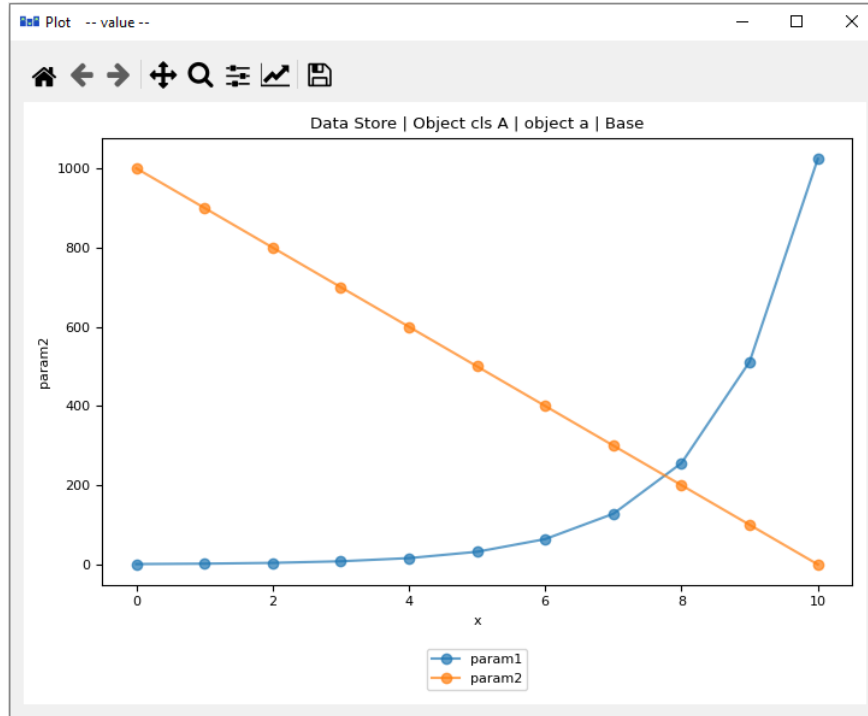




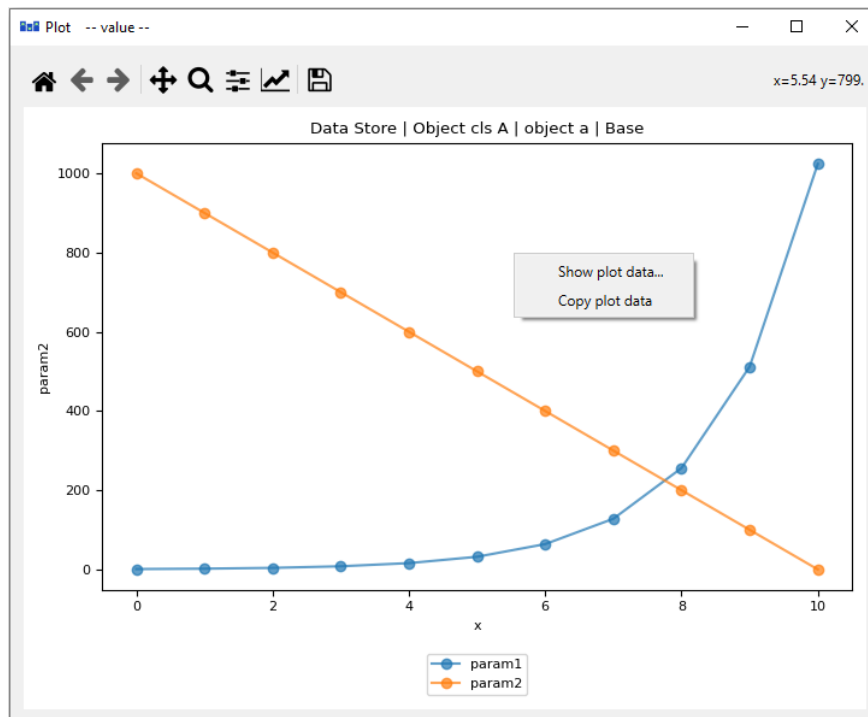
Selecting data in multiple columns plots the selection in a single window.

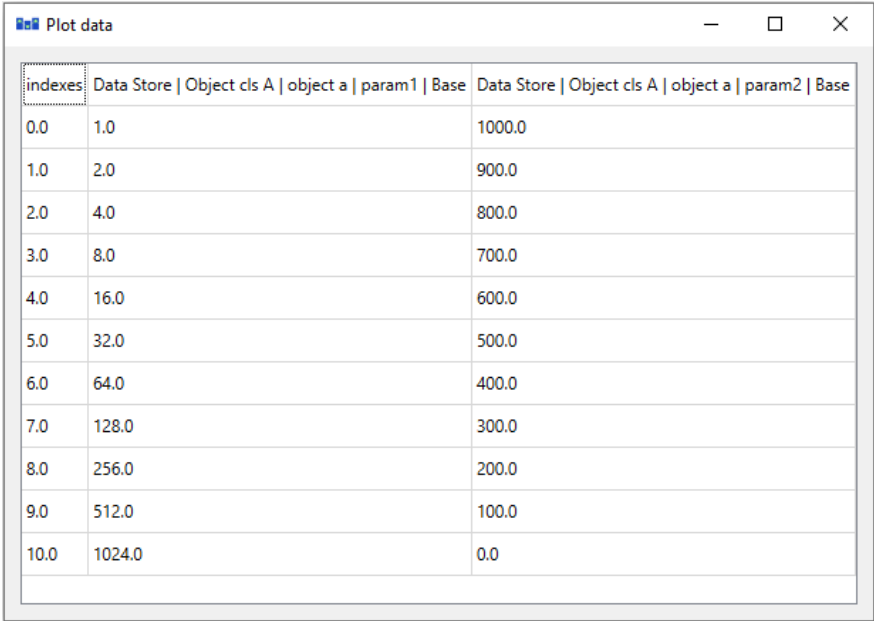
To add a plot to an existing window select the target plot window from the *Plot in window* submenu. There are some restrictions for what kinds of plots can be shown on the same window. In the example below two different maps have been plotted on the same graph.





If a plot is clicked with the right mouse button, options to copy or show the plot data are presented. When the data is copied it is saved to the clipboard in csv format with tab as the delimiter. If **Show plot data...** is clicked a new window opens that contains a table of the data used in the plot.



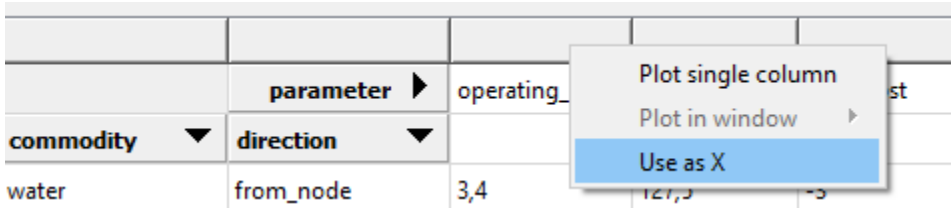


indexes	Data Store Object cls A object a param1 Base	Data Store Object cls A object a param2 Base
0.0	1.0	1000.0
1.0	2.0	900.0
2.0	4.0	800.0
3.0	8.0	700.0
4.0	16.0	600.0
5.0	32.0	500.0
6.0	64.0	400.0
7.0	128.0	300.0
8.0	256.0	200.0
9.0	512.0	100.0
10.0	1024.0	0.0

12.1 X column in pivot table

It is possible to plot a column of scalar values against a designated X column in the pivot table.

To set a column as the X column **right click** the top empty area above the column header and select *Use as X* from the popup menu. (X) in the topmost cell indicates that the column is designated as the X axis.



	parameter	operating_	
commodity	direction		
water	from_node	3,4	127,5

When selecting and plotting other columns in the same table the data will be plotted against the values in the X column instead of row numbers.

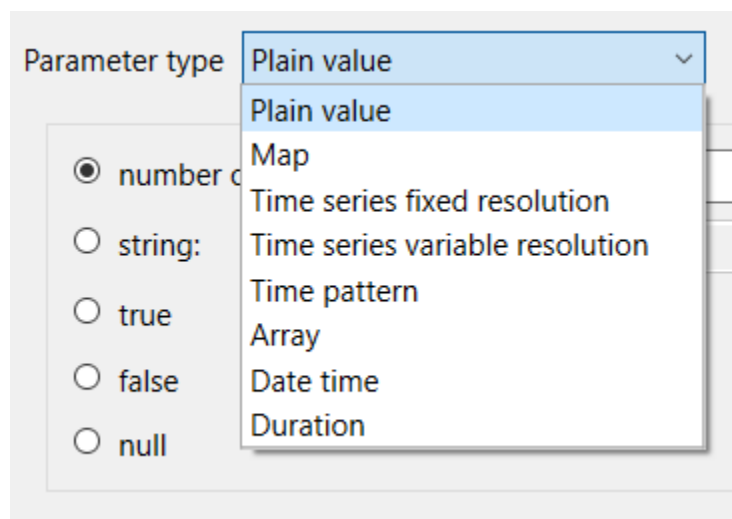
PARAMETER VALUE EDITOR

Parameter value editor is used to edit object and relationship parameter values such as time series, time patterns or durations. It can also convert between different value types, e.g. from a time series to a time pattern.

The editor can be opened by right clicking on the value field that you want to edit in a Spine database editor and selecting **Edit...** from the popup menu. In most cases the editor can be opened again by double clicking an existing value. With plain values like strings and floats this will not work since double clicking starts the text editor in that field.

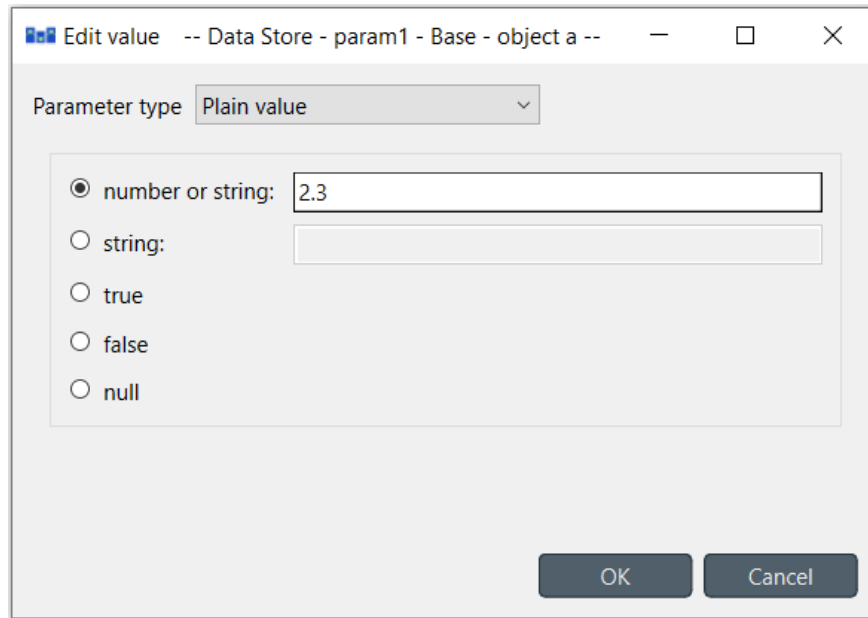
13.1 Choosing value type

The combo box at the top of the editor window allows changing the type of the current value.



13.2 Plain values

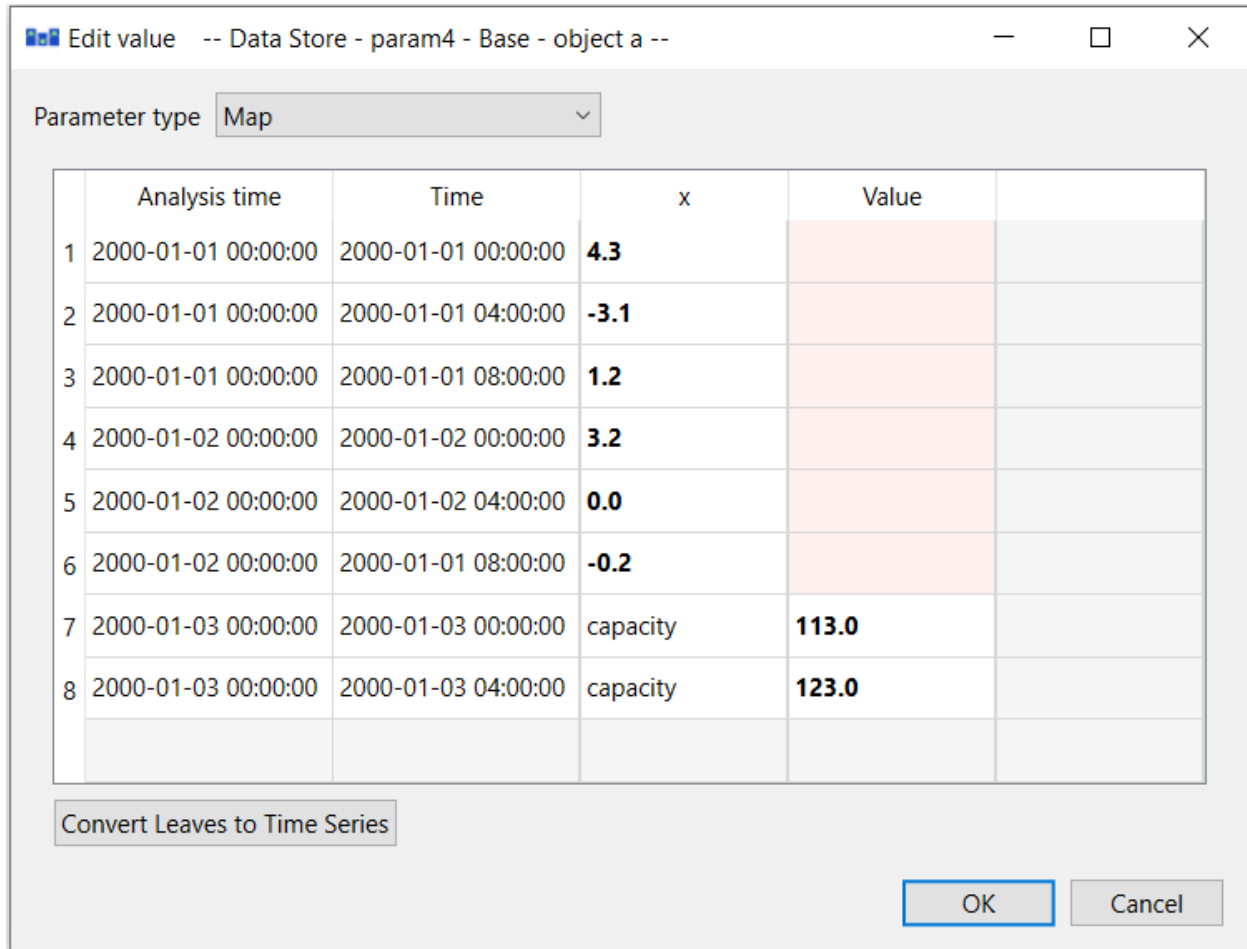
The simplest parameter values are of the *Plain value* type. The editor window lets you to write a number or string directly to the input field or set it to true, false or null as needed.



Numbers and strings can also be inserted without going through the value editor by double clicking on a value field. If the users input is a number like 3.14, the value type will be interpreted as *number or string*. If the input is a string like “ok”, the value type will automatically be set to *string*.

13.3 Maps

Maps are versatile nested data structures designed to contain complex data including one and multi dimensional indexed arrays. In Parameter value editor a map is shown as a table where the last non-empty cell on each row contains the value while the preceding cells contain the value’s indexes.

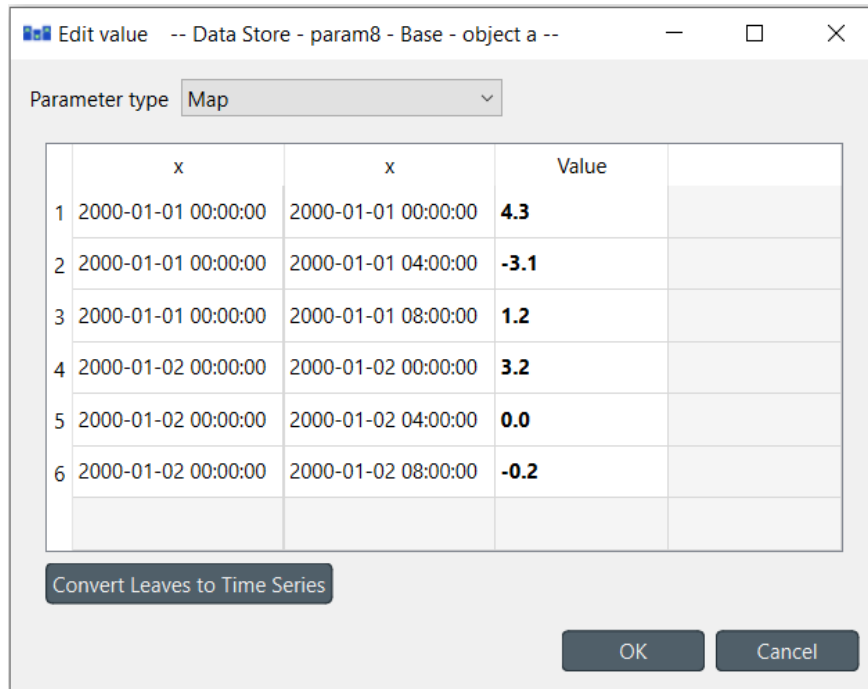


The extra gray column on the right allows expanding the map with a new dimension. You can append a value to the map by editing the bottom gray row. The reddish cells are merely a guide for the eye to indicate that the map has different nesting depths.

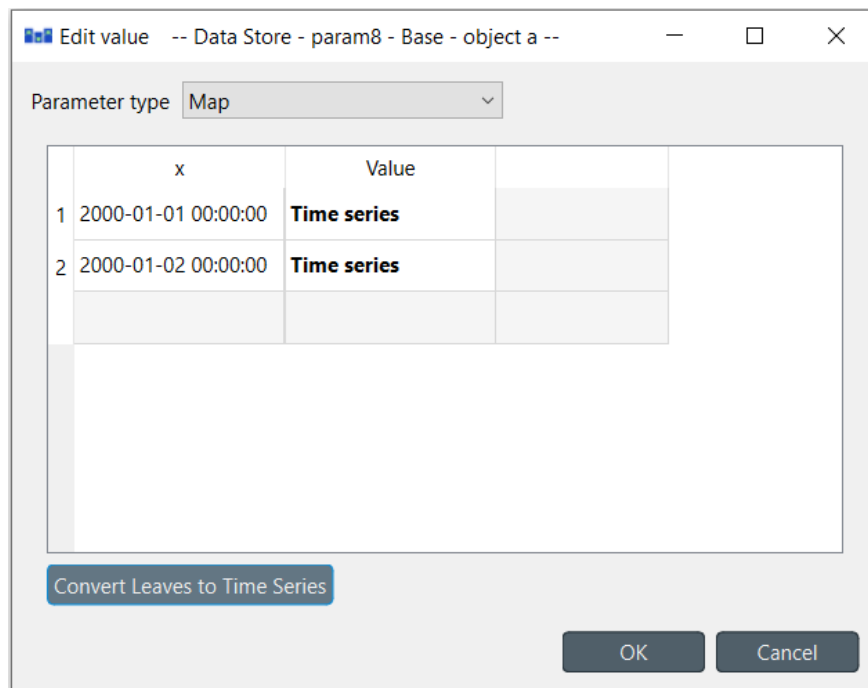
A **Right click** popup menu gives options to open a value editor for individual cells, plot individual and multiple cells, insert/remove rows or columns (effectively changing map's dimensions), trim empty columns from the right hand side and to copy and paste data. Copying and pasting data works between cells and external programs and can be also done using the usual **Ctrl+C** and **Ctrl+V** keyboard shortcuts.

The default name for new columns is *x*. Index names can however be modified. If a column holds both indices and data, the column header can also be modified. The last column of a map has to always contain values and therefore the header can't be modified from the default name *Value*.

Convert leaves to time series 'compacts' the map by converting the last dimension into time series. This works only if the last dimension's type is datetime. For example the following map contains two time dimensions. Since the indexes are datetimes, the 'inner' dimension can be converted to time series.

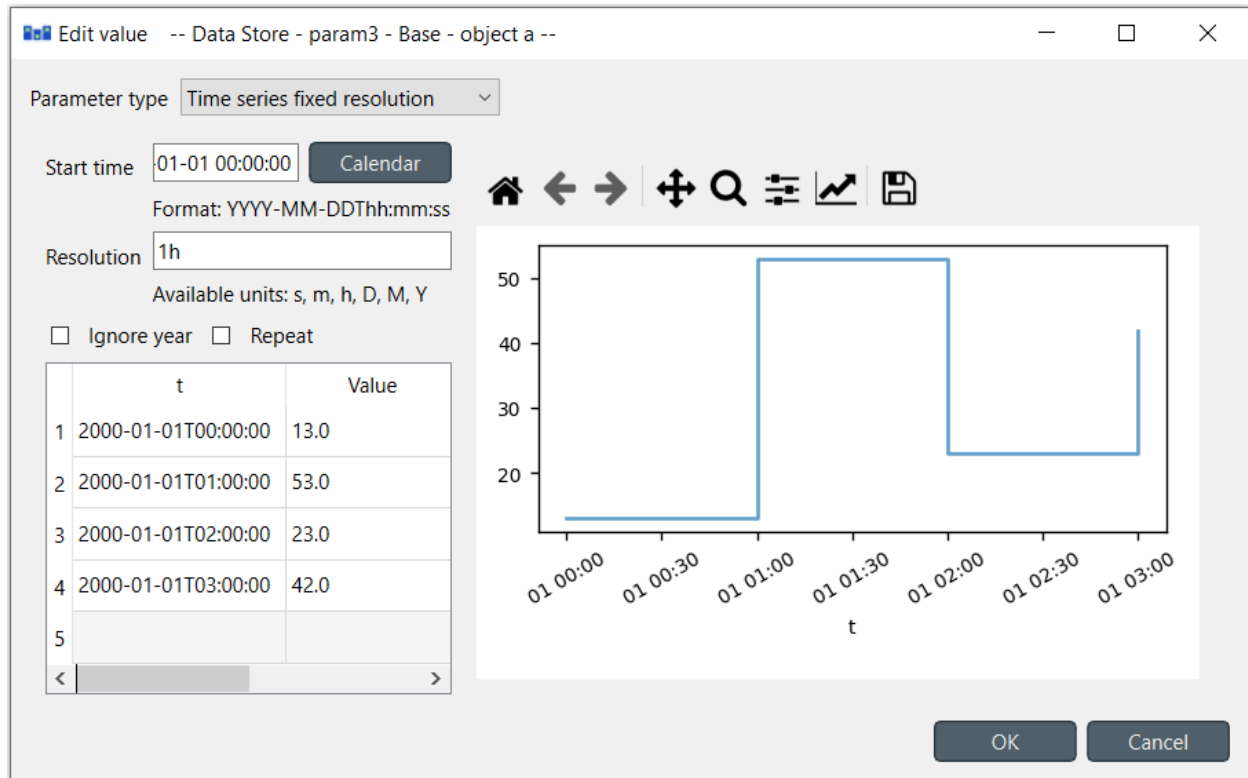


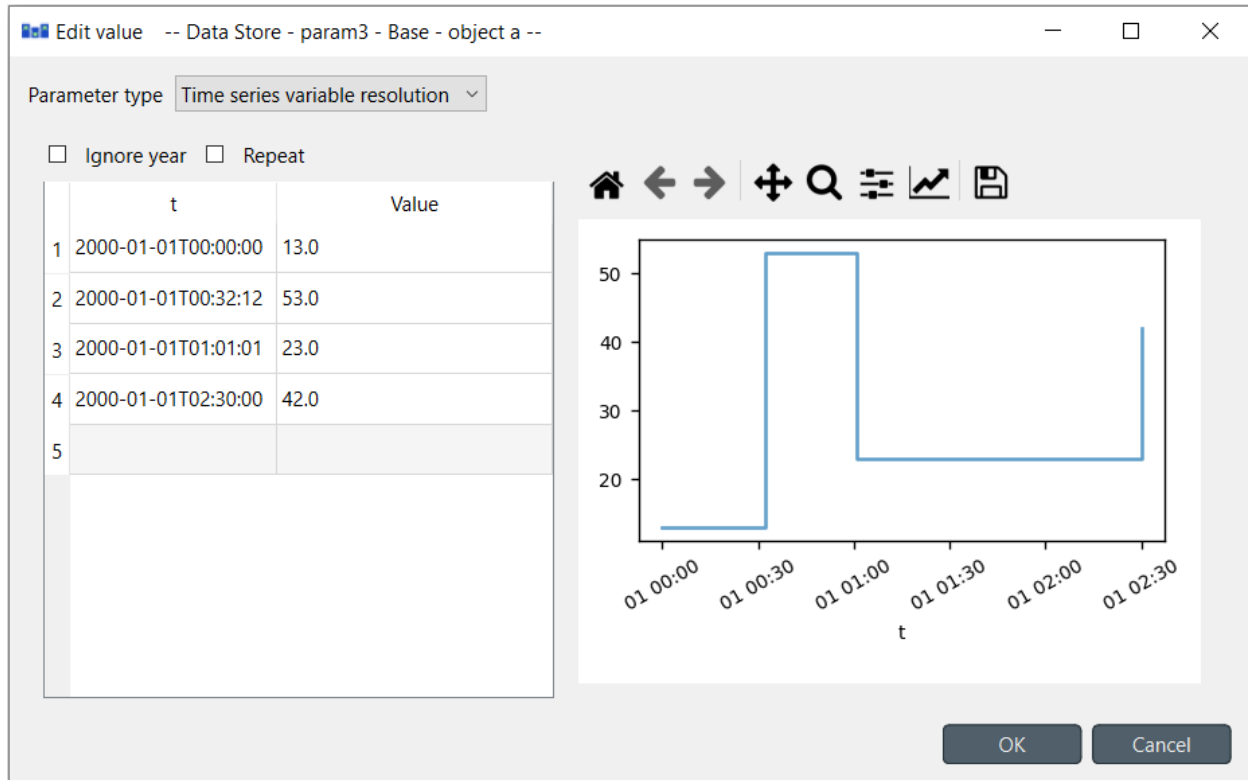
After clicking **Convert leaves to time series** the map looks like this:



13.4 Time series

There are two types of time series: *variable* and *fixed resolution*. Variable resolution means that the time stamps can be arbitrary while in fixed resolution series the time steps between consecutive stamps are fixed.





The editor window is split into two in both cases. The left side holds all the options and a table with all the data while the right side shows a plot of the series. The plot is not editable and is for visualization purposes only.

In the table rows can be added or removed from a popup menu available by a **right click**. Editing the last gray row appends a new value to the series. Data can be copied and pasted by **Ctrl+C** and **Ctrl+V**. Copying from/to an external spreadsheet program is supported.

The time steps of a fixed resolution series are edited by the *Start time* and *Resolution* fields. The format for the start time is [ISO8601](#). The *Resolution* field takes a single time step or a comma separated list of steps. If a list of resolution steps is provided then the steps are repeated so as to fit the data in the table.

The *Ignore year* option available for both variable and fixed resolution time series allows the time series to be used independent of the year. Only the month, day and time information is used by the model.

The *Repeat* option means that the time series is cycled, i.e. it starts from the beginning once the time steps run out.

13.5 Time patterns

The time pattern editor holds a single table which shows the *time period* on the right column and the corresponding values on the left. Inserting/removing rows and copy-pasting works as in the time series editor.

Parameter type: Time pattern

[Link to time period syntax.](#)

	p	Value
1	h1-2,h4-5	100.0
2	h3-4	200.0
3	h5-10	300.0
4		

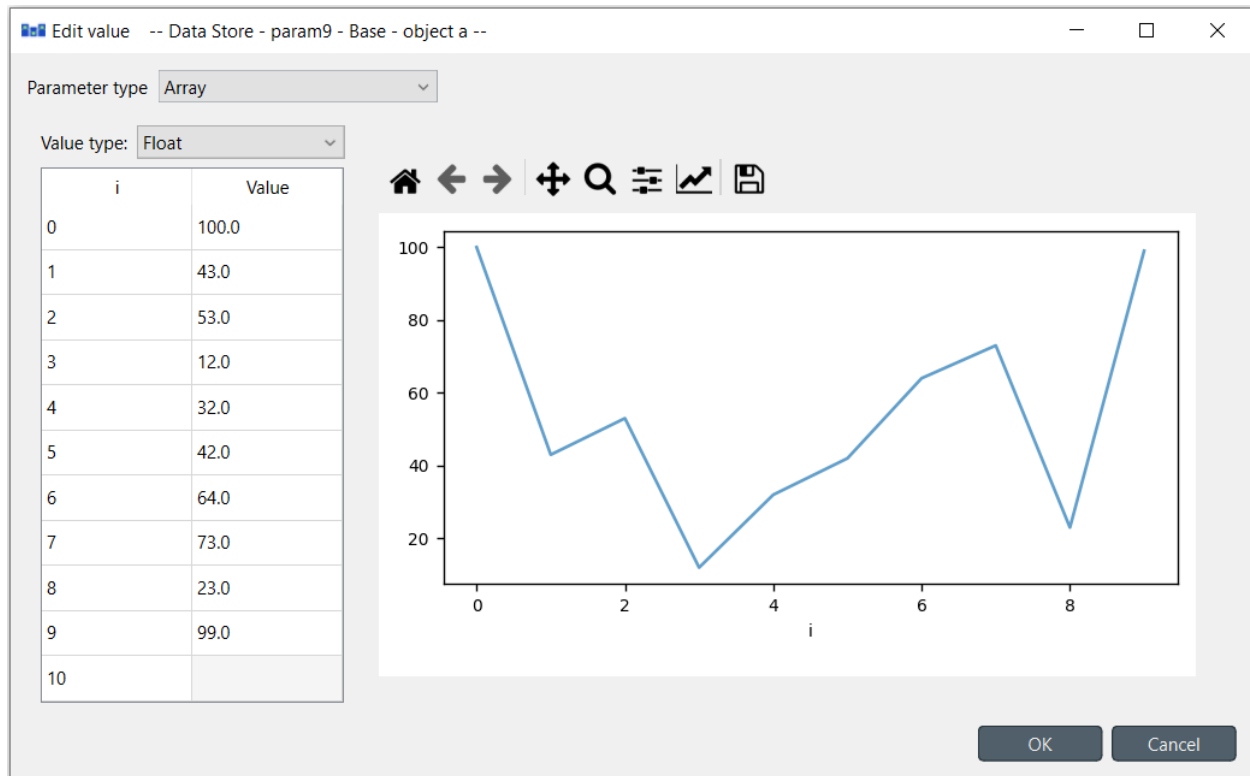
OK Cancel

Time periods consist of the following elements:

- An *interval* of time in a given *time-unit*. The format is $Ua-b$, where U is either Y (for year), M (for month), D (for day), WD (for weekday), h (for hour), m (for minute), or s (for second); and a and b are two integers corresponding to the lower and upper bound, respectively.
- An *intersection* of intervals. The format is $s1;s2;\dots$, where $s1, s2, \dots$, are intervals as described above.
- A *union of ranges*. The format is $r1,r2,\dots$, where $r1, r2, \dots$, are either intervals or intersections of intervals as described above.

13.6 Arrays

Arrays are lists of values of a single type. Their editor is split into two: the left side holds the actual array while the right side contains a plot of the array values versus the values' positions within the array. Note that not all value types can be plotted. The type can be selected from the *Value type* combobox. Inserting/removing rows and copy-pasting works as in the time series editor.



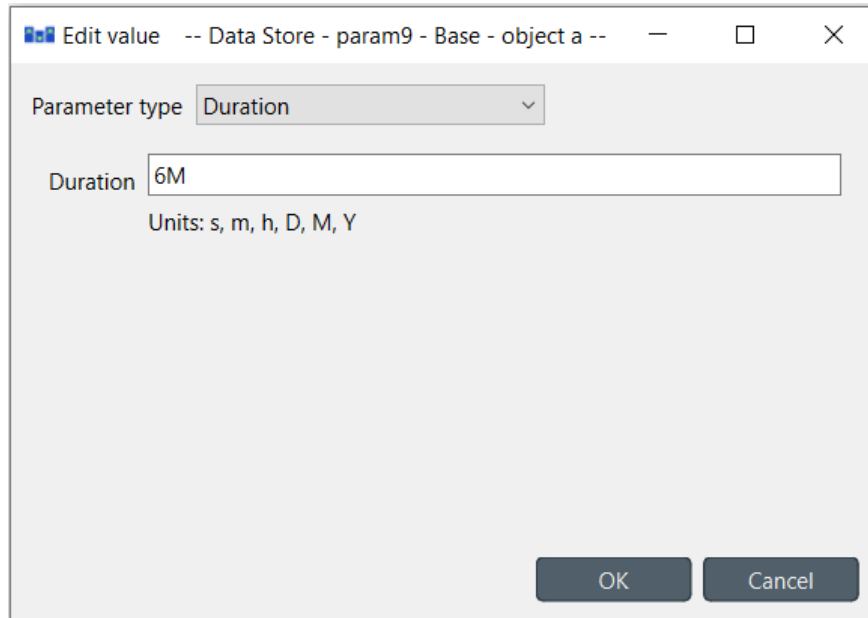
13.7 Datetimes

The datetime value should be entered in [ISO8601](#) format. Clicking small arrow on right end of the input field opens up a calendar that can be used to select a date.

The screenshot shows the 'Edit value' dialog for a Date time parameter. The 'Parameter type' is set to 'Date time'. The 'Datetime' input field contains the value '2023-07-28T10:20:15'. Below the input field, the format 'Format: YYYY--MM-DDThh:mm:ss' is displayed. The dialog has 'OK' and 'Cancel' buttons at the bottom.

13.8 Durations

A single value or a comma separated list of time durations can be entered to the *Duration* field.



The screenshot shows a dialog box titled "Edit value -- Data Store - param9 - Base - object a --". Inside the dialog, there is a "Parameter type" dropdown menu set to "Duration". Below this, there is a text input field labeled "Duration" containing the value "6M". Underneath the input field, it says "Units: s, m, h, D, M, Y". At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

SPINE METADATA DESCRIPTION

This is the metadata description for Spine, edited from <https://frictionlessdata.io/specs/data-package/>.

14.1 Required properties

title

One sentence description for the data sources.

sources

The raw sources of the data. Each source must have a **title** property and optionally a **path** property. Example:

```
"sources": [{  
  "title": "World Bank and OECD",  
  "path": "http://data.worldbank.org/indicator/NY.GDP.MKTP.CD"  
}]
```

contributors

The people or organisations who contributed to the data. Must include **title** and may include **path**, **email**, **role** and **organization**. Example:

```
"contributors": [{  
  "title": "Joe Bloggs",  
  "email": "joe@bloggs.com",  
  "path": "http://www.bloggs.com",  
  "role": "author"  
}]
```

Role is one of author, publisher, maintainer, wrangler, or contributor.

created

The date this data was created or put together, in ISO8601 format (YYYY-MM-DDTHH:MM)

14.2 Optional properties

description

A description of the data. Describe here how the data was collected, how it was processed etc.

The description **MUST** be markdown formatted – this also allows for simple plain text as plain text is itself valid markdown. The first paragraph (up to the first double line break) should be usable as summary information for the package.

spine_results_metadata

Key contains results metadata (described in a separate document).

keywords

An array of keywords

homepage

A URL for the home on the web that is related to this data package.

name

Name of the data package, url-usable, all-lowercase string.

id

Globally unique id, such as UUID or DOI

licenses

Licences that apply to the data. Each item must have a `name` property ([Open Definition license ID](#)) or a `path` property and may contain `title`. Example:

```
"licenses": [{
  "name": "ODC-PDDL-1.0",
  "path": "http://opendatacommons.org/licenses/pddl/",
  "title": "Open Data Commons Public Domain Dedication and License v1.0"
}]
```

temporal

Temporal properties of the data (if applicable). Example using [DCMI Period Encoding Scheme](#):

```
"temporal": {
  "start": "2000-01-01",
  "end": "2000-12-31",
  "name": "The first year of the 21st century"
}
```

spatial

Spatial properties of the data (if applicable). Example using [DCMI Point Encoding Scheme](#):

```
"spatial": {
  "east": 23.766667,
  "north": 61.5,
  "projection": "geographic coordinates (WGS 84)",
  "name": "Tampere, Finland"
}
```

unitOfMeasurement

Unit of measurement. Can also be embedded in description.

IMPORTING AND EXPORTING DATA

This section explains the different ways of importing and exporting data to and from a Spine database.

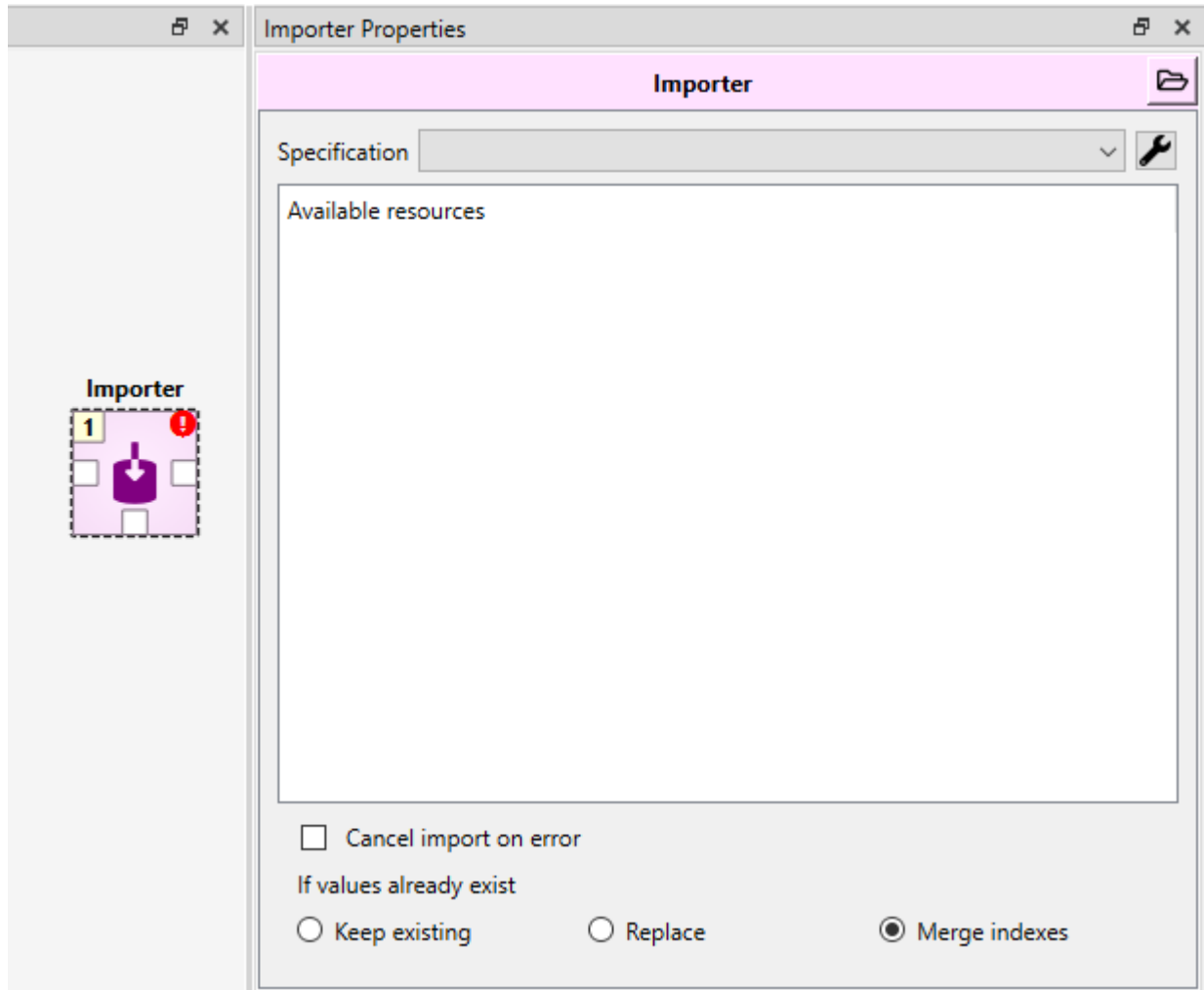
- *Importing data with Importer*
 - *Importer specification editor*
 - *Anonymous mode*
- *Exporting data with Exporter*
 - *Properties dock*
 - *Exporter specification editor*
 - * *Mapping options*
 - * *Mapping specification*
 - *CSV and multiple tables*
 - *SQL export*
 - *GAMS.gdx export*
- *Basic regular expressions for filtering*

15.1 Importing data with Importer

Data importing is handled by the Importer project item which can import tabulated and to some degree tree-structured data into a Spine database from various formats. The same functionality is also available in **Spine database editor** from **File->Import** but using an Importer item is preferred because then the process is documented and repeatable.

Tip: A Tool item can also be connected to Importer to import tool's output files to a database.

The heart of Importer is the **Importer Specification Editor** window in which the mappings from source data to Spine database entities are set up. The editor window can be accessed by the button in Importer's Properties dock where existing specifications can also be selected or by double-clicking the item on the **Design View**.



In the **Properties** tab there is also the option to cancel the import if a non-fatal error occurs during execution and also three options for how to handle if some of the values that you are trying to import already exist in the target Data Store. The three choices are:

Keep existing

When this is selected, data that is already in the target Data Store will be kept and the new data the Importer tried to import is forgotten.

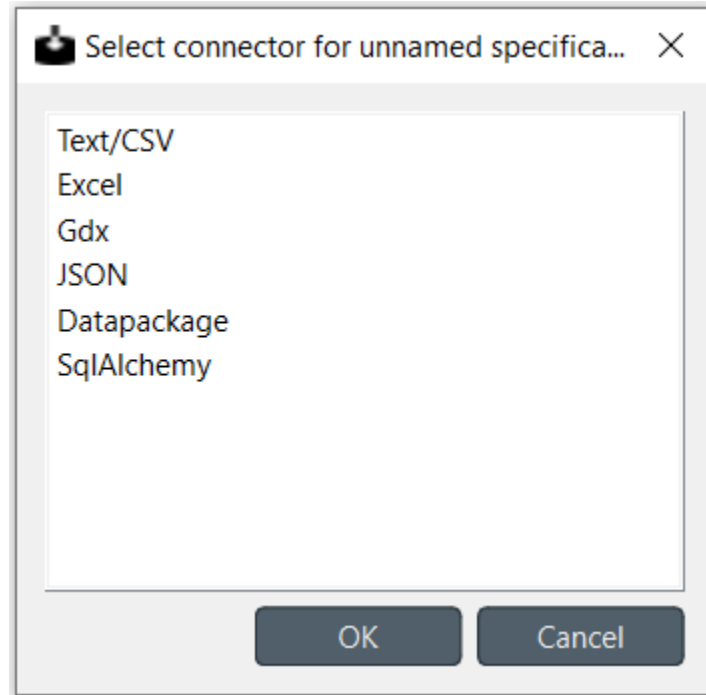
Replace

This option means the imported data replaces old data in the target Data Store.

Merge indexes

This option works mostly like the *Replace* option, but for indexed values like maps, time series and arrays the behavior is a little different. With these value types the old index-value pairs in the target Data Store will be kept and the new value pairs will be appended to it. If an index that is going to get imported already exist, the new imported value will overwrite the old value.

When the specification editor is opened for an Importer without a specification, a list of the supported connectors (file formats or other data sources) is presented. If the Importer is already connected to some data, it may have already selected the proper connector and is only waiting confirmation

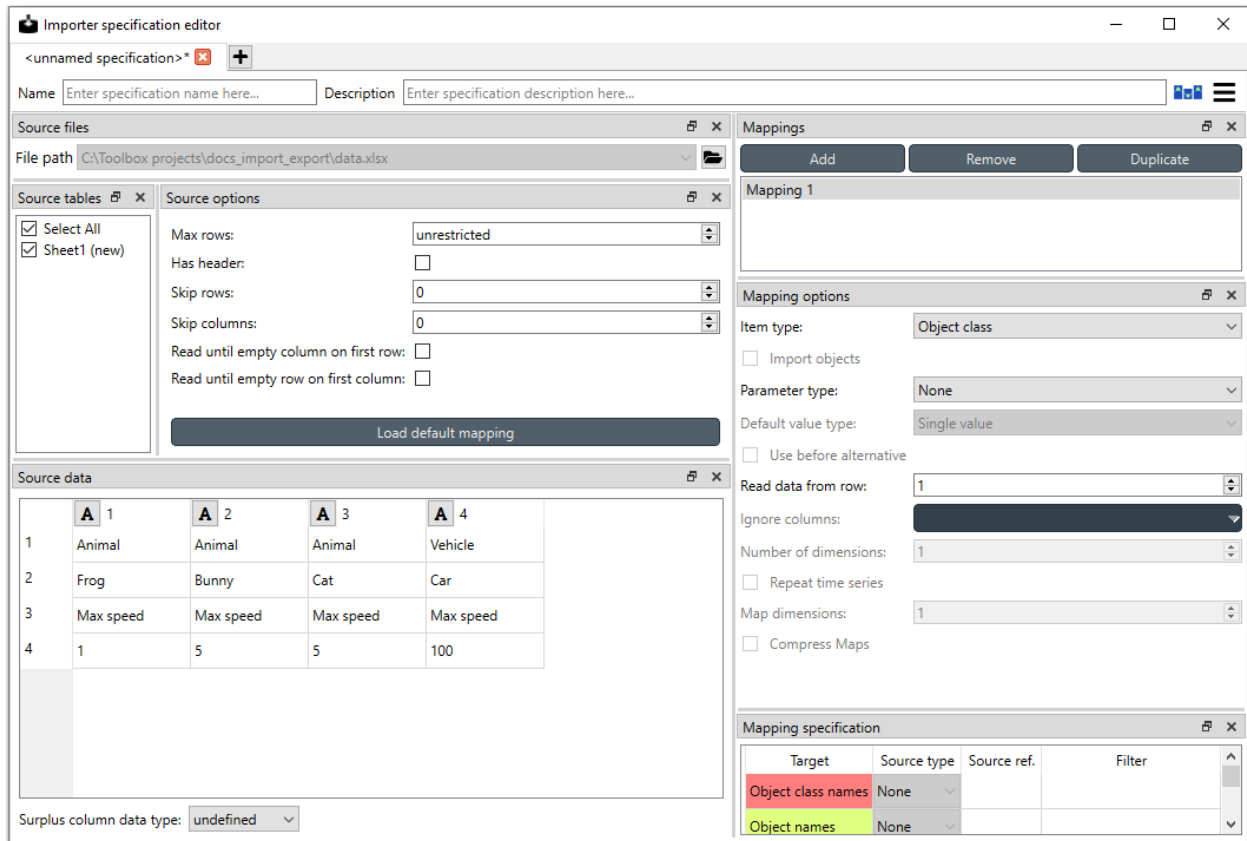


When creating a **Importer** specification, it is usually helpful to have the data already connected to the **Importer** in question so it is easier to visualize how the importer is handling the data.

15.1.1 Importer specification editor

The left side of the **Importer Specification Editor** contains four dock widgets that deal with the input source data. **Source files** contains the filepath to the current input file. **Source tables** contains the different tables the file has or that user has set up. **Source options** has a few options on how to handle the incoming data. The available options depend on the connector type. **Source data** allows for previewing of the data selected in **Source files** and also of the currently selected mapping.

The right side of the spec editor shows import mappings for the table selected in **Source tables**. **Mappings** lets you add, duplicate and remove mappings or select a mapping to modify its options. **Mapping options** allow you to specify things like what item type you are importing. In **Mapping specification** you can select the specific places in the source data where the object names, values etc. will be taken from.



All tables available in the file selected in **Source files** are listed in **Source tables**. If the file does not have tables or the file type does not support them (e.g. CSV), all of the file’s data will be in a single table called ‘data’. The tables can be selected and deselected using the check boxes and only the selected ones will be imported. The option *Select All* is useful for selecting or deselecting all tables. If the Importer is opened in *anonymous mode*, there is also the option to add tables.

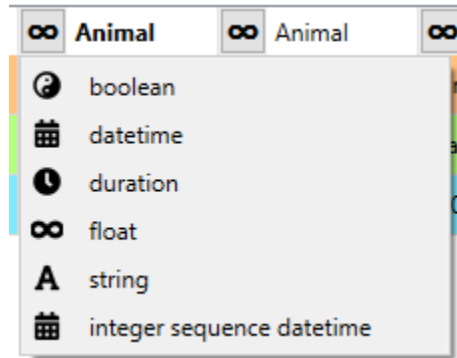
Tip: Multiple CSV files can be bundled into a *datapackage* which uses its own connector in Importer. Specifically, each CSV file in the datapackage shows up as a separate table in **Source tables**. See [Packing CSV files into datapackage](#) for more information on how to pack CSVs into a datapackage automatically within your workflow.

Source options contains options to “format” the incoming data. The available options can differ depending on the selected connector. The above picture shows the available options for Excel files. **Max rows** specifies the amount of rows from the input data that are considered by the Importer. The option **Has header** converts the first row into headers. **Skip rows** and **Skip columns** skip the first *N* specified rows or columns from the table. If the table has empty rows or columns and some other data after that that you don’t want to use, **Read until empty row/column on first column/row** options can be used to “crop” the imported data to the first relevant block of information. Other possible options for different connector types include **Encoding**, **Delimiter**, **Custom Delimiter**, **Quotechar** and **Maximum Depth**. **Load default mapping** sets all of the selections in the spec editor to their default values. Be careful not to press this button unless you want to wipe the whole specification clean.

Note: If you are working on a specification and accidentally press the *load default mapping* button you can undo previous changes for the specification from the hamburger menu or by pressing **Ctrl+Z**. To redo actions, or press **Ctrl+Y**.

Source data shows the selected table’s data and a preview of how the selected mapping will import the data. An

important aspect of data import is whether each item in the input data should be read as a string, a number, a time stamp or something else. By default all input data is read as strings. However, more often than not things like parameter values are actually numbers. Though types are usually casted automatically, it is possible to manually control what type of data each column (and sometimes each row) contains from the preview table. Clicking the data type indicator button on column or row headers pops up a menu with a selection of available data types. Right clicking the column/row header also gives the opportunity to change the data type of all columns/rows at once.

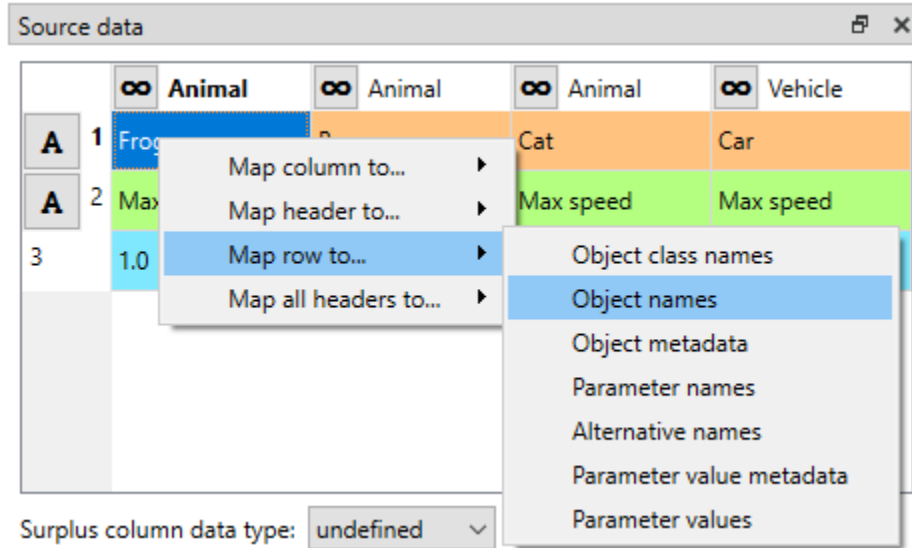


Under **Mappings** you can manage mappings by adding new ones and removing or duplicating existing ones. Each table has its own mappings and every mapping has its own options. In **Mappings** you can select the mapping that you want to start modifying. Having multiple mappings for a single table allows to for example import relationship classes and object classes at the same time from a single table in a file.

Mapping options helps the importer get a feel for what kind of data it will be importing. The available *item type* options are *Object class*, *Relationship class*, *Object group*, *Alternative*, *Scenario*, *Scenario alternative*, *Parameter value list*, *Feature*, *Tool*, *Tool feature* and *Tool feature method*. The other available options are dependent on the Item type. *Import objects* allows to import objects alongside relationships or object groups. *Parameter type* is used to specify what type of parameters, if any, the sheet contains. It has options *Value*, *Definition* and *None*. If *Value* or *Definition* is selected the value or respectively the default value type can be set from the drop-down list. *Use before alternative* is only available for *Scenario alternative* -item type. *Read data from row* lets you specify the row where the importer starts to read the data. *Ignore columns* allow you to select individual columns that you want to exclude from the whole importing process. *Number of dimensions* sets the amount of dimensions the relationship to be imported has. *Repeat time series* sets the repeat flag to true when importing time series. *Map dimensions* sets the number of map indexes when importing map values.

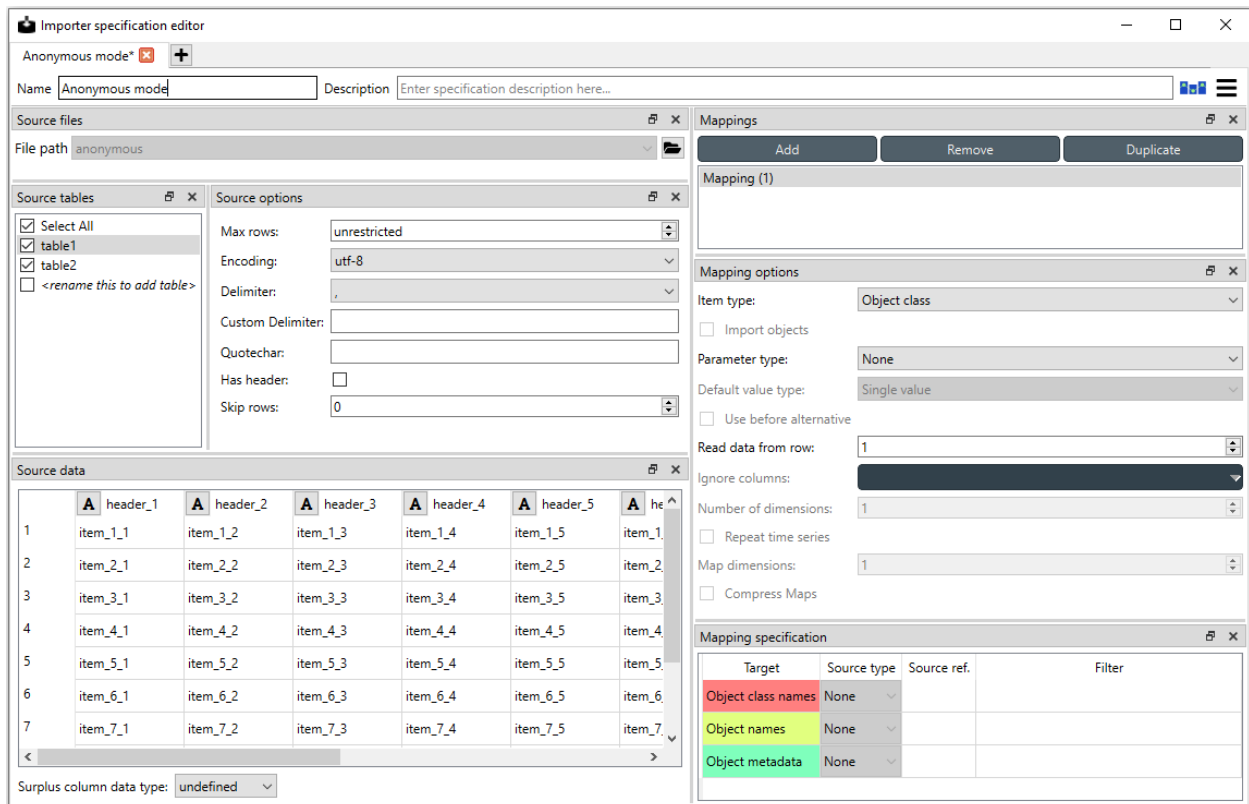
Once everything in **Mapping options** is in order, the next step is to set the mapping specification. **Mapping specification** is the part where the decisions are made on how the input data is interpreted: which row or column contains the entity class names, parameter values, time stamps and so on. The **Mapping specification** dock widget contains all of the targets that the selected mapping options specify. Each target has a *Source reference* and a *Source type*. *Source type* specifies if the data for the target is coming in the form of a column, row, table name etc. In the *Source ref.* section you can pinpoint to the exact row, column etc. to use as the data. The *Filter* section can be used to further specify which values to include using regular expressions. More on regular expressions in section [Basic regular expressions for filtering](#).

It might be helpful to fill in the *Source type* and *Source ref.* using the preview table in the *Sources data*. Right clicking on the table cells shows a popup menu that lets one to configure where the selected row/column/header is mapped to. It can also be used to simultaneously map all headers to one target.



15.1.2 Anonymous mode

The importer specification editor can be opened in a mode where there is no input data available. This might be useful when creating or modifying a generalized specifications. Anonymous mode entered when opening the specification of an Importer without incoming files or when opening the spec editor from Toolbox **Main Toolbar**.



In anonymous mode new tables can be created in **Source tables** by double clicking *<rename this to add table>* and writing in a name for the new table. The **Source data** will contain an infinite grid of cells on which you can create different mappings.

Note: You can exit the Anonymous mode by browsing to and selecting an existing file using the controls in **Source files** dock.

15.2 Exporting data with Exporter

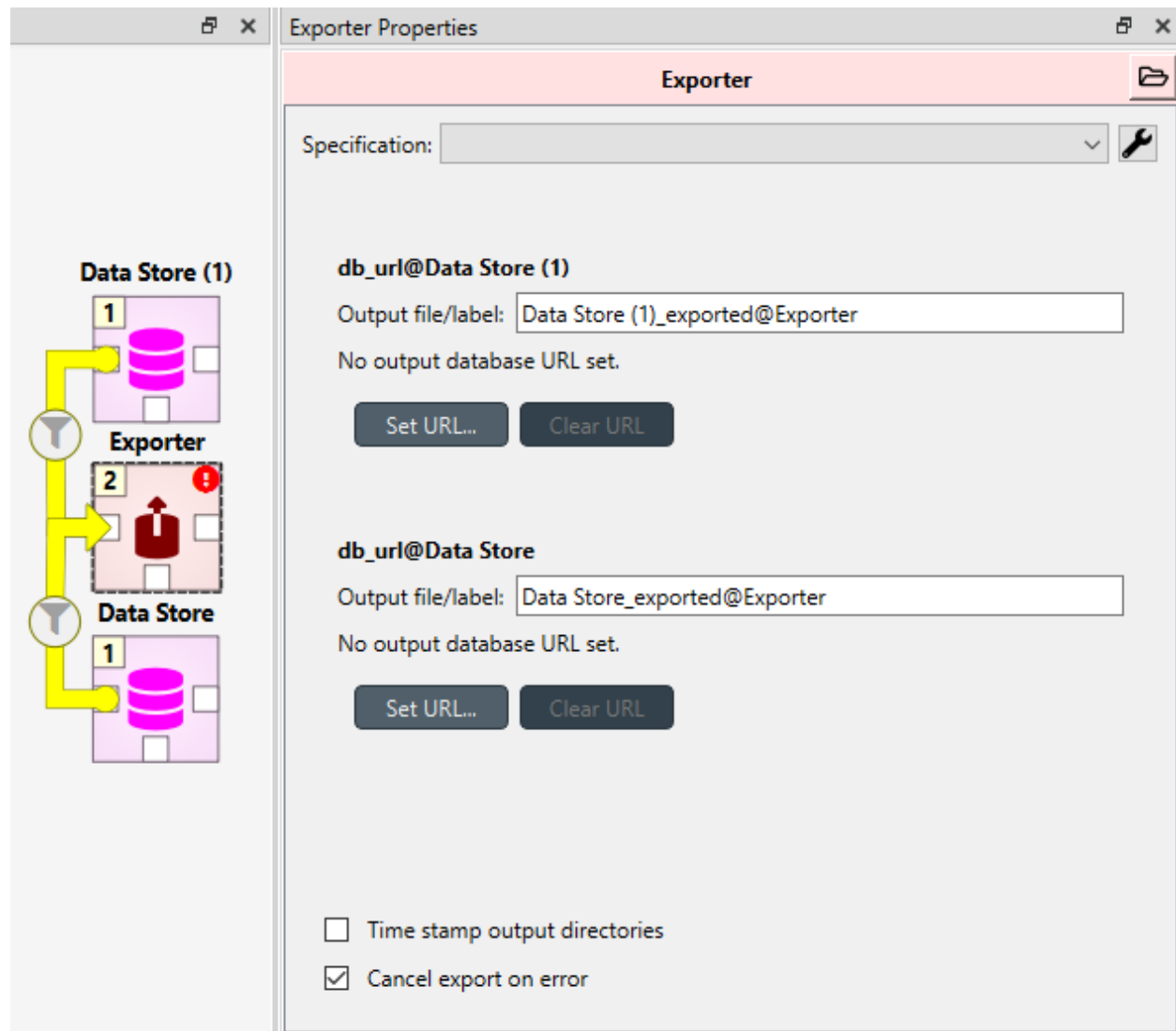
Exporter writes database data into regular files that can be used by Tools and external software that do not read the Spine database format. Various tabulated file formats are supported some of which require specific export settings; see below for more details.

At its heart Exporter maps database items such as entity class or entity names to an output table. Each item has a user given output **position** on the table, for example a column number. By default data is mapped to columns but it is also possible to create pivot tables.

Exporter also uses specifications so the same configurations can be reused by other exporters even in other projects. The specification can be edited in *Exporter specification editor* which is accessible by the button in the item's Properties dock or by double clicking Exporter's icon on the **Design View**. A specification that is not associated with any specific Exporter project item can be created and edited from the Main toolbar.

15.2.1 Properties dock

Exporter's Properties dock controls project item specific settings that are not part of the item's specification.



Specification used by the active Exporter item can be selected from the *Specification* combobox. The button opens *Exporter specification editor* where it is possible to edit the specification.

Data Stores that are connected to the exporter and are available for export are listed below the *Specification* combobox. An output label is required for each database and one Exporter can't have the same output label for two different Data Stores at the same time. Two different Exporters can have the same output label names since they are located in a different directories. The default label for the output files is of the format <name of input Data Store>_exported@<name of Exporter>.

Checking the *Time stamp output directories* box adds a time stamp to the item's output directories preventing output files from being overwritten. This may be useful for debugging purposes.

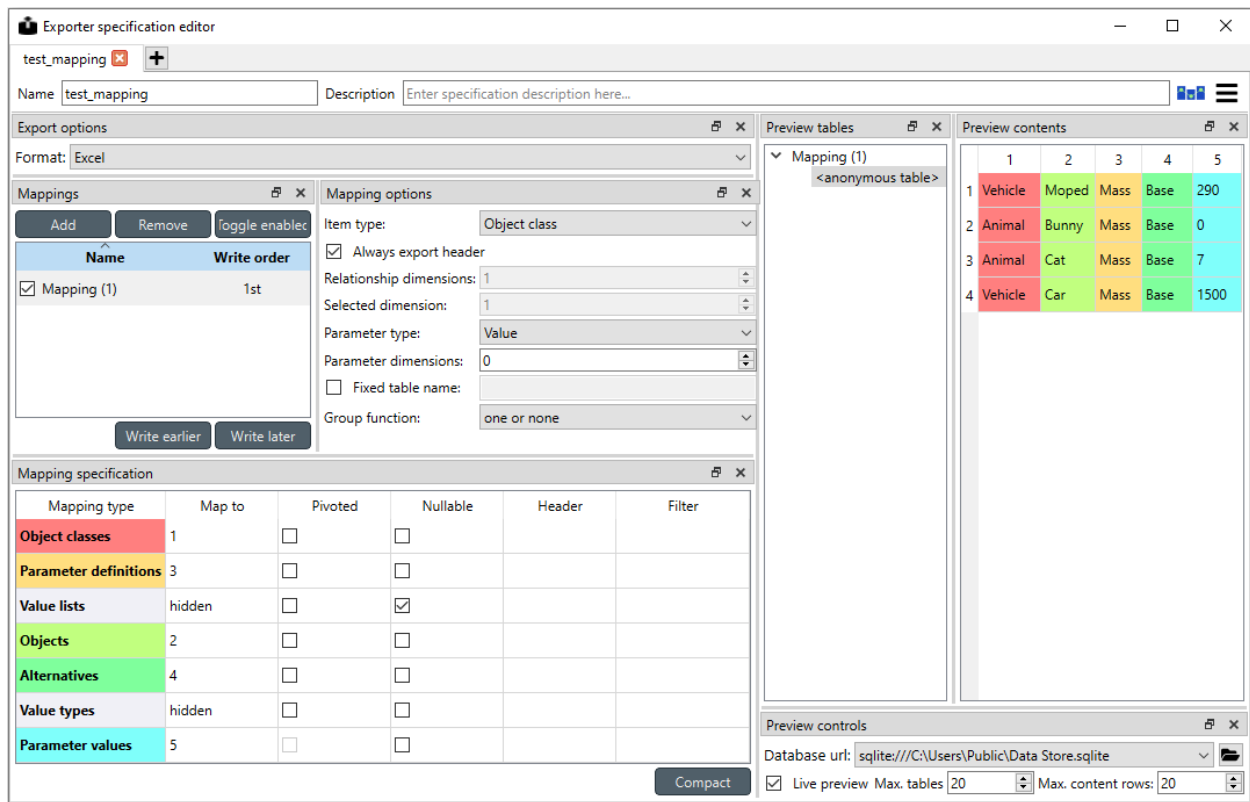
The *Cancel export on error* checkbox controls whether execution bails out on errors that may be otherwise non-fatal.

Exporter's data directory can be opened in system's file browser by the button. The output files are written in data directory's output subdirectory.

15.2.2 Exporter specification editor

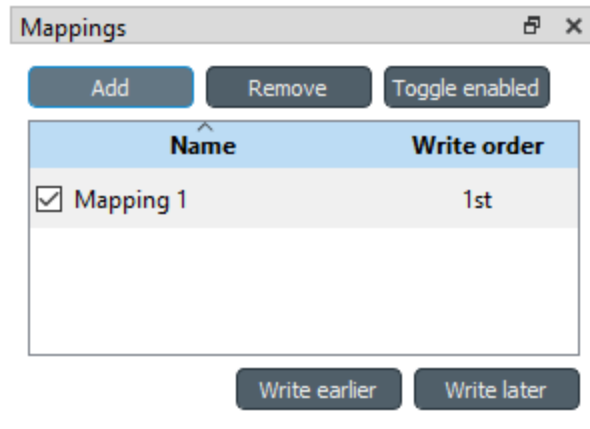
Specification editor is used to create **mappings** that define how data is exported to the output file. Mappings define one or more tables and their contents but are otherwise output format agnostic. Some output formats, e.g. SQL and gdx, interpret the tables in specific ways, however. Other formats which inherently cannot write multiple tables into a single file, such as CSV, may end up exporting multiple files. See the sections below for format specific intricacies.

When opened for the first time Specification editor looks like in the image below. The window is tabbed allowing multiple specifications to be edited at the same time. Each tab consists of dock widgets which can be reorganized to suit the user's needs. The 'hamburger' menu on the top right corner gives access to some important actions such as *Save* and *Close*. *Undo* and *redo* can be found from the menu as well. There is also a *Duplicate* option which creates a new tab in the spec editor that is otherwise the same but has no name and is missing the database url under *Preview controls*. This is handy if you want to create a new Exporter specification using an existing template instead of always starting from the beginning.

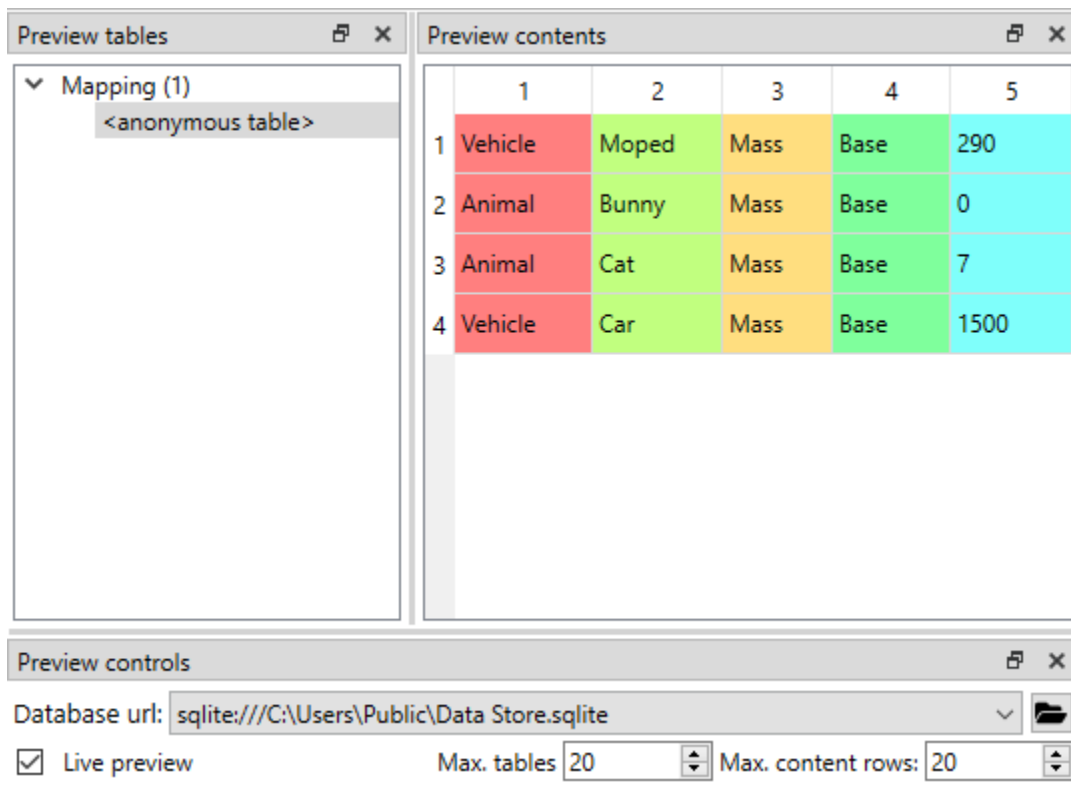


The only requirement for a specification is a name. This can be given on the *Name* field on the top bar. The *Description* field allows for an additional explanatory text. The current output format can be changed by the *Format* combobox on *Export options* dock.

Specification's mappings are listed in the *Mappings* dock widget shown below. The *Add* button adds a new mapping while the *Remove* button removes selected mappings. Mappings can be renamed by double clicking their names on the list. The checkbox in front of mapping's name shows if the mapping is currently enabled. Only enabled mappings are exported when the Exporter is executed. Use the *Toggle enabled* button to toggle the enabled state of all mappings at once.



The tables defined by the mappings are written in the order shown on the mapping list's *Write order* column. This may be important if the tables need to be in certain order in the output file or when multiple mappings output to a single table. Mappings can be sorted by their write order by clicking the header of the *Write order* column. The *Write earlier* and *Write later* buttons move the currently selected mapping up and down the list.



A preview of what will be written to the output is available in the preview dock widgets. To enable it, check the *Live preview* checkbox. A database connection is needed to generate the preview. The *Preview controls* dock provides widgets to choose an existing database or to load one from a file. Once a database is available and the preview is enabled the mappings and the tables they would output are listed on the *Preview tables* dock. Selecting a table from the list shows the table's contents on the *Preview contents* dock.

Note: The preview is oblivious of any filters possibly set up in the workflow. Therefore, it may show entries, e.g. parameter values, that would be filtered out during execution.

Mapping options



The currently selected mapping is edited using the controls in *Mapping options* and *Mapping specification* docks. The *Mapping options* dock contains controls that apply to the mapping as a whole, e.g. what data the output tables contain. It is important to choose *Item type* correctly since it determines what database items the mapping outputs and also dictates the mapping types that will be visible in the *Mapping specification* dock widget. It has options *Object class*, *Relationship class*, *Relationship class with object parameter*, *Object group*, *Alternative*, *Scenario*, *Scenario alternative*, *Parameter value list*, *Feature*, *Tool*, *Tool feature* and *Tool feature method*. The rest of the options besides *Group function* are item type specific and may not be available for all selections.

The screenshot shows the 'Mapping options' dock with the following settings:

- Item type:** Object class (dropdown menu)
- ☒ **Always export header** (checkbox)
- Relationship dimensions:** 1 (spinbox)
- Selected dimension:** 1 (spinbox)
- Parameter type:** Value (dropdown menu)
- Parameter dimensions:** 0 (spinbox)
- ☐ **Fixed table name:** (text field)
- Group function:** one or none (dropdown menu)

Checking the *Always export header* checkbox outputs a table that has fixed headers even if the table is otherwise empty. If *Item type* is *Relationship class*, the *Relationship dimensions* spinbox can be used to specify the maximum number of relationships' dimensions that the mapping is able to handle. *Selected dimensions* option is only available for the *Relationship class with object parameter* item type and it is used to specify the relationship dimension where the object parameters are selected from. Parameters can be outputted by choosing their value type using the *Parameter type* combobox. The *Value* choice adds rows to *Mapping specification* for parameter values associated with individual entities while *Default value* allows outputting parameters' default values. The maximum number of value dimensions in case of indexed values (time series, maps, time patterns, arrays) the mapping can handle is controlled by the *Parameter dimensions* spinbox. The *Fixed table name* checkbox enables giving a user defined table name to the mapping's output table. In case the mapping is pivoted and *Mapping specification* contains items that are *hidden*, it is possible that a number of data elements end up in the same output table cell. The *Group function* combobox offers some basic functions to aggregate such data into the cells.

Mapping specification

Mapping specification  

Mapping type	Map to	Pivoted	Nullable	Header	Filter
Object classes	1	<input type="checkbox"/>	<input type="checkbox"/>		
Parameter definitions	3	<input type="checkbox"/>	<input type="checkbox"/>		
Value lists	hidden	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
Objects	2	<input type="checkbox"/>	<input type="checkbox"/>		
Alternatives	4	<input type="checkbox"/>	<input type="checkbox"/>		
Value types	hidden	<input type="checkbox"/>	<input type="checkbox"/>		
Parameter values	5	<input type="checkbox"/>	<input type="checkbox"/>		

Compact

Mapping specification contains a table which defines the structure of the mapping's output tables. Like mentioned before, the contents of the table depends on choices on *Mapping options*, e.g. the item type, parameter type or dimensions. Each row corresponds to an item in the database: object class names, object names, parameter values etc. The item's name is given in the *Mapping type* column. The colors help to identify the corresponding elements in the preview.

The *Map to* column defines the **position** of the item, that is, where the item is written or otherwise used when the output tables are generated. By default, a plain integral number in this column means that the item is written to that column in the output table. From the other choices, *hidden* means that the item will not show on the output. *Table name*, on the other hand, uses the item as output table names. For example, outputting object classes as table names will generate one new table for every object class in the database, each named after the class. Each table in turn will contain the parameters and objects of the table's object class. If multiple mappings generate a table with a common name then each mapping appends to the same table in the order specified by the *Write order* column on *Mappings* dock.

The *column header* position makes the item a column header for a **buddy item**. Buddy items have some kind of logical relationship with their column header, for instance the buddy of an object class is its objects; setting the object class to *column header* will write the name of the class as the objects' column header.

Note: Currently, buddies are fixed and defined only for a small set database items. Therefore, *column header* will not always produce sensible results.

Changing the column and pivot header row positions leaves sometimes gaps in the output table. If such gaps are not desirable the **Compact** button reorders the positions by removing the gaps. This may be useful when the output format requires such gapless tables.

The checkboxes in *Pivoted* column on the *Mapping specification* dock toggle the mapping into pivoted mode. One or more items on the table can be set as pivoted. They then act as a pivot header for the data item which is the last non-hidden item on the list. Once checked as pivoted, an item's position column defines a pivot header row instead of output column.

By default a row ends up in the output table only when all mapping items yield some data. For example, when exporting object classes and objects, only classes that have objects get written to output. However, sometimes it is useful to export 'empty' object classes as well. For this purpose a mapping can be set as **nullable** in the *Nullable* column. Continuing the example, checking the *Nullable* checkbox for *Objects* would produce an output table with all object classes including ones without objects. The position where objects would normally be outputted are left empty for those classes.

Besides the *column header* position it is possible to give fixed column headers to items using the *Header* column in *Mapping specification* dock. Note that checking the *Always export header* option in the *Mapping options* dock outputs the fixed headers even if there is no other data in a table.

The *Mapping specification* dock's *Filter* column provides refined control on which database items the mapping outputs. The column uses regular expressions (see section [Basic regular expressions for filtering](#)) to filter what gets outputted.

15.2.3 CSV and multiple tables

CSV files are flat text files and therefore do not directly support multiple tables. Instead, multiple tables are handled as separate output files.

Only mappings that output an **anonymous table** actually write to the file/label specified on the Exporter's properties dock. Named tables get written to files named after the table plus the `.csv` extension. For example, a table named `node` would result in a file called `node.csv`.

15.2.4 SQL export

To set up export to a remote database, first an Exporter specification with SQL selected as the format needs to be saved. The Exporter needs to also be connected to some input Data Store. From the Exporters **Properties** dock widget an output database can be specified for each input Data Store respectively by clicking the **Set URL...** button. A small new window opens with a few settings to set up the output database. Currently only `mysql` and `sqlite` are supported, even though `mssql`, `postgres` and `oracle` are also listed as options for the dialect. Once a URL is set it can be removed by pressing the **Clear URL** button on the **Properties** tab.

The SQL backend writes the tables to the target database in a relatively straightforward way:

- Tables are named after the table name provided by the mappings. **Anonymous tables** are not supported.
- The first row of each table is used as column names in the database. Thus, each column in a mapping should have a fixed header or a header produced by an item set to *column header* position.
- Column data types are sniffed from the second row. Empty values or a missing row result in string type.
- There must be an item assigned to each column. Empty columns confuse the SQL backend.

- Pivot tables do not generally make sense with the SQL backend unless the resulting table somehow follows the above rules.

15.2.5 GAMS.gdx export

Note: You need to have GAMS installed to use this functionality. However, you do not need to own a GAMS license as the demo version works just as well. See [Setting up Consoles and External Tools](#) for more information.

The.gdx backend turns the output tables to GAMS sets, parameters and scalars following the rules below:

- Table names correspond the names of sets, parameters and scalars. Thus, **anonymous tables** are not supported.
- There must be an item assigned to each column. Empty columns confuse the.gdx backend.
- Pivot tables do not generally make sense with the.gdx backend unless the resulting table somehow follows the rules listed here.

Sets:

- Everything that is not identified as parameter or scalar is considered a GAMS set.
- Each column corresponds to a dimension.
- The first row is used to name the dimension's domain. Thus, each column in a mapping should have a fixed header or a header produced by an item set to *column header* position. Note that * is a valid fixed header and means that the dimension has no specific domain.

Parameters:

- A table that contains no header in the last (rightmost) column is considered a GAMS parameter.
- The last column should contain the parameter's values while the other columns contain the values' dimension.
- Dimensions' domains are taken from the header row, see **Sets** above. Note, that the value column must not have a header.

Scalars:

- A table that contains a numerical value in the top left cell is considered a GAMS scalar. Everything else (except the table name) is ignored.
- The data in the top left cell is the scalar's value.

15.3 Basic regular expressions for filtering

See regular expressions on [wikipedia](#) and on Python's [documentation](#). Both the Exporter and Importer have applications for regular expressions in their respective *Mapping specifications* dock widgets. Below are examples on how to create some basic filters for these applications.

Single item

Writing the item's name to the field filters out all other items. For example, to output the object class called 'node' only, write node to the *Filter* field.

OR operator

The vertical bar | serves as the OR operator. node|unit as a filter for object classes would output classes named 'node' and 'unit'.

Excluding an item

While perhaps not the most suitable task for regular expressions it is still possible to ‘negate’ a filter. As an example, `^(?!node)` excludes all item names that start with ‘node’.

SPINE ENGINE SERVER

16.1 Notes

Here is a list of items that you should be aware of when running projects on Spine Engine Server.

- **Projects must be self-contained.** The project directory must contain all input and output files, file/db references, Specification files and scripts.
- **Work or Source directory execution mode** setting is ignored. Tools are always executed in ‘source’ directory, i.e. in the directory where the Tool Spec main script resides.
- **Python Basic Console.** Interpreter setting in Tool Specification Editor is ignored. Basic Console runs the same Python that was used in starting the Server.
- **Python Jupyter Console.** Kernel spec setting in Tool Specification Editor is ignored. Jupyter Console is launched using the **python3** kernel spec. This must be installed before the server is started. See instructions below.
- **Julia Basic Console.** Interpreter setting in app settings (Tools page in File->Settings) is ignored. Basic Console runs the Julia that is found in PATH. See installation instructions below.
- **Julia Jupyter Console.** Kernel spec setting in app settings (Tools page in File->Settings) is ignored. Jupyter Console is launched using the **julia-1.8** kernel spec. This must be installed before the server is started. See instructions below.

16.2 Setting up Spine Engine Server

You can either install the entire Spine Toolbox or just the required parts to run the Spine Engine Server.

16.2.1 Minimal Installation

Spine Engine server does not need the entire Spine Toolbox installation. Only *spine-engine*, *spinedb-api* and *spine-items*. Note that the dependencies of *spine-items* are not needed. Here are the step-by-step instructions for a minimal installation:

1.1 Make a miniconda environment & activate it

1.2. Clone [spine-engine](#)

1.3. cd to *spine-engine* repo root, run:

```
pip install -e .
```

1.4. Clone `spine-items`

1.5. cd to `spine-items` repo root

1.6. Install `spine-items` **without dependencies** by running:

```
pip install --no-deps -e .
```

16.2.2 Full Installation

Install Spine Toolbox regularly

1.1. Make a miniconda environment & activate

1.2. Clone `Spine Toolbox`

1.3. Follow the [installation instructions in README.md](#)

16.2.3 Finalize Setting Up and Start Server

2. Create security credentials (optional)

- cd to `<spine_engine_repo_root>/spine_engine/server/`
- Create security certificates by running:

```
python certificate_creator.py
```

- The certificates are created into `<spine_engine_repo_root>/spine_engine/server/certs/` directory.
- Configure allowed endpoints by creating file `<spine_engine_repo_root>/spine_engine/server/connectivity/certs/allowEndpoints`
- Add IP addresses of the remote end points to the file

3. Install IPython kernel spec (`python3`) to enable Jupyter Console execution of Python Tools

- Run:

```
python -m pip install ipykernel
```

4. Install Julia 1.8

- Download from <https://julialang.org/downloads/> or run `apt-get install julia` on Ubuntu

5. Install IJulia kernel spec (`julia-1.8`) to enable Jupyter Console execution of Julia tools

- Open Julia REPL and press `/` to enter pkg mode. Run:

```
add IJulia
```

- This installs `julia-1.8` kernel spec to `~/.local/share/jupyter/kernels` on Ubuntu or to `%APP-DATA%jupyterkernels` on Windows

6. Start Spine Engine Server

- cd to `<spine_engine_repo_root>/spine_engine/server/`
- Without security, run:

```
python start_server.py 50001
```

- where 50001 is the server port number.
- With Stonehouse security, run:

```
python start_server.py 50001 StoneHouse ./certs
```

- where 50001 is an example server port number, StoneHouse is the security model, and the path is the folder containing the security credentials.

Note: Valid port range is 49152-65535.

16.3 Setting up Spine Toolbox (client)

1. (Optional) If server is started using StoneHouse security, copy security credentials from the server to some directory. Server's secret key does not need to be copied.
2. Start Spine Toolbox and open a project
3. Open the **Engine** page in Spine Toolbox Settings (**File -> Settings...**)
 - Enable remote execution from the checkbox (Enabled)
 - Set up the Spine Engine Server settings (host, port, security model, and security folder). Host is 127.0.0.1 when the Server runs on the same computer as the client
 - Click Ok, to close and save the new Settings
4. Click to execute the project

TERMINOLOGY

Here is a list of definitions related to Spine project, SpineOpt.jl, and Spine Toolbox.

- **Arc** Graph theory term. See *Connection*.
- **Case study** Spine project has 13 case studies that help to improve, validate and deploy different aspects of the SpineOpt.jl and Spine Toolbox.
- **Connection** an arrow on Spine Toolbox Design View that is used to connect project items to each other to form a DAG.
- **Data Connection** is a project item used to store a collection of data files that may or may not be in Spine data format. It facilitates data transfer from original data sources e.g. spreadsheet files to Spine Toolbox. The original data source file does not need to conform to the format that Spine Toolbox is capable of reading, since there we can use an interpreting layer (Importer) between the raw data and the Spine format database (Data Store).
- **Data Package** is a data container format consisting of a metadata descriptor file (`datapackage.json`) and resources such as data files.
- **Data sources** are all the original, unaltered, sources of data that are used to generate necessary input data for Spine Toolbox tools.
- **Data Store** is a project item. It's a Spine Toolbox internal data container which follows the Spine data model. A data store is implemented using a database, it may be, for example, an SQL database.
- **Design View** A *sub-window* on Spine Toolbox main window, where project items and connections are visualized.
- **Direct predecessor** Immediate predecessor. E.g. in DAG $x \rightarrow y \rightarrow z$, direct predecessor of node z is node y . See also predecessor.
- **Direct successor** Immediate successor. E.g. in DAG $x \rightarrow y \rightarrow z$, direct successor of node x is node y . See also successor.
- **Directed Acyclic Graph (DAG)** Finite directed graph with no directed cycles. It consists of vertices and edges. In Spine Toolbox, we use project items as vertices and connections as edges to build a DAG that represents a data processing chain (workflow).
- **Edge** Graph theory term. See *Connection*
- **GdxExporter** is a project item that allows exporting a Spine data structure from a Data Store into a `.gdx` file which can be used as an input file in a Tool.
- **Importer** is a project item that can be used to import data from e.g. an Excel file, transform it to Spine data structure, and into a Data Store.
- **Node** Graph theory term. See *Project item*.
- **Predecessor** Graph theory term that is also used in Spine Toolbox. Preceding project items of a certain project item in a DAG. For example, in DAG $x \rightarrow y \rightarrow z$, nodes x and y are the predecessors of node z .

- **Project** in Spine Toolbox consists of project items and connections, which are used to build a data processing chain for solving a particular problem. Data processing chains are built and executed using the rules of Directed Acyclic Graphs. There can be any number of project items in a project.
- **Project item** Spine Toolbox projects consist of project items. Project items together with connections are used to build Directed Acyclic Graphs (DAG). Project items act as nodes and connections act as edges in the DAG. See [Project Items](#) for an up-to-date list on project items available in Spine Toolbox.
- **Scenario** A scenario is a meaningful data set for the target tool.
- **Spine data structure** Spine data structure defines the format for storing and moving data within Spine Toolbox. A generic data structure allows representation of many different modelling entities. Data structures have a class defining the type of entity they represent, can have properties and can be related to other data structures. Spine data structures can be manipulated and visualized within Spine Toolbox while SpineOpt.jl will be able to directly utilize as well as output them.
- **SpineOpt.jl** An interpreter, which formulates a solver-ready mixed-integer optimization problem based on the input data and the equations defined in the SpineOpt.jl. Outputs the solver results.
- **Source directory** In context of Tool specifications, a source directory is the directory where the main program file of the Tool specification is located. This is also the recommended place for saving the Tool specification file (.json).
- **Successor** Graph theory term that is also used in Spine Toolbox. Following project items of a certain project item in a DAG. For example, in DAG $x \rightarrow y \rightarrow z$, nodes y and z are the successors of node x .
- **Tool** is a project item that is used to execute Python, Julia, GAMS, executable scripts, or simulation models. This is done by creating a Tool specification defining the script or program the user wants to execute in Spine Toolbox. Then you need to attach the Tool specification to a Tool project item. Tools can be used to execute a computational process or a simulation model, or it can also be a process that converts data or calculates a new variable. In general, Tools may take some data as input and produce an output.
- **Tool specification** is a JSON structure that contains metadata required by Spine Toolbox to execute a computational process or a simulation model. The metadata contains; type of the program (Python, Julia, GAMS, executable), main program file (which can be e.g. a Windows batch (.bat) file or for Python scripts this would be the .py file where the `__main__()` method is located), All additional required program files, any optional input files (e.g. data), and output files. Also any command line arguments can be defined in a Tool specification. SpineOpt.jl is a Tool specification from Spine Toolbox's point-of-view.
- **Use case** Potential way to use Spine Toolbox. Use cases together are used to test the functionality and stability of Spine Toolbox and SpineOpt.jl under different potential circumstances.
- **Vertex** Graph theory term. See *Project item*.
- **View** A project item that can be used for visualizing project data.
- **Work directory** Tool specifications can be executed in *Source directory* or in *work directory*. When a Tool specification is executed in a work directory, Spine Toolbox creates a new *work* directory, copies all required and optional files needed for running the Tool specification to this directory and executes it there. After execution has finished, output or result files can be copied into a timestamped (archive) directory from the work directory.

CONTRIBUTION GUIDE

All are welcome to contribute! This guide is based on a set of best practices for open source projects [JF18].

18.1 Reporting Bugs

18.1.1 Due Diligence

Before submitting a bug report, please do the following:

Perform basic troubleshooting steps.

1. **Make sure you're on the latest version.** If you're not on the most recent version, your problem may have been solved already! Upgrading is always the best first step.
2. **Try older versions.** If you're already on the latest release, try rolling back a few minor versions (e.g. if on 1.7, try 1.5 or 1.6) and see if the problem goes away. This will help the devs narrow down when the problem first arose in the commit log.
3. **Try switching up dependency versions.** If you think the problem may be due to a problem with a dependency (other libraries, etc.). Try upgrading/downgrading those as well.
4. **Search the project's bug/issue tracker to make sure it's not a known issue.** If you don't find a pre-existing issue, consider checking with the maintainers in case the problem is non-bug-related. [Spine Toolbox issue tracker is here](#).

18.1.2 What to Put in Your Bug Report

Make sure your report gets the attention it deserves: bug reports with missing information may be ignored or punted back to you, delaying a fix. The below constitutes a bare minimum; more info is almost always better:

1. What version of the Python interpreter are you using? E.g. Python 2.7.3, Python 3.6?
2. What operating system are you on? Windows? (Vista, 7, 8, 8.1, 10). 32-bit or 64-bit? Mac OS X? (e.g. 10.7.4, 10.9.0) Linux (Which distro? Which version of that distro? 32 or 64 bits?) Again, more detail is better.
3. Which version or versions of the software are you using? If you have forked the project from Git, which branch and which commit? Otherwise, supply the application version number (Help->About menu). Also, ideally you followed the advice above and have ruled out (or verified that the problem exists in) a few different versions.
4. How can the developers recreate the bug? What were the steps used to invoke it. A screenshot demonstrating the bug is usually the most helpful thing you can report (if applicable) Relevant output from the Event Log or debug messages from the console of your run, should also be included.

18.2 Feature Requests

The developers of Spine Toolbox are happy to hear new ideas for features or improvements to existing functionality. The format for requesting new features is free. Just fill out the required fields on the issue tracker and give a description of the new feature. A picture accompanying the description is a good way to get your idea into development faster. But before you make a new issue, check that there isn't a related idea already open in the issue tracker. If you have an idea on how to improve an existing idea, just join the conversation.

18.3 Submitting features/bugfixes

If you feel like you can fix a bug that's been bothering you or you want to add a new feature to the application but the devs seem to be too busy with something else, please follow the instructions in the following sections on how to contribute code.

18.3.1 Coding Style

Follow the style you see used in the repository! Consistency with the rest of the project always trumps other considerations. It doesn't matter if you have your own style or if the rest of the code breaks with the greater community - just follow along.

Spine Toolbox coding style follows [PEP-8](#) style guide for Python code with the following variations:

- Maximum line length is 120 characters. Longer lines are acceptable for a good reason.
- [Google style](#) docstrings with the title and input parameters are required for all classes, functions, and methods. For small functions or methods only the summary is necessary. Return types are highly recommended but not required if it is obvious what the function or method returns.
- Use double-quoted strings instead of single-quoted strings (e.g. "hello").
- Other deviations from PEP-8 can be discussed.

18.3.2 Commit messages

The commit message should tell *what* was changed and *why*. Details on *how* it was done can usually be left out, if the code itself is self-explanatory (remember source comments too!). Separate the subject line from the body with a blank line. The subject line (max. 50 chars) should explain in condensed form what happened using imperative mood, i.e. using verbs like 'change', 'fix' or 'add'. Start the subject line with a capital letter. Do not use the issue number on the subject line, as it does not tell much to a person who's not aware of that particular issue. For more info see Chris Beams' 'Seven rules of a great Git commit message' [[CB14](#)].

A good example (inspired by [[CB14](#)])

```
Fix bugs when updating parameters in foo and bar
```

```
Body of the commit message starts after a blank line. Explain here in more
detail the reasons why you made the change, how things worked before and how they work_
↪now.
```

```
Also explain why
```

```
You can use hyphens to make bulleted lists:
```

```
- Foo was added because of bar
```

(continues on next page)

(continued from previous page)

```
- Baz was not used so it was deleted
```

Add references to issue tracker (if any) at the end.

Solves: #123

See also: #456, #789

18.3.3 Contributing to the User Guide

Spine Toolbox uses Sphinx to create HTML pages from restructured text (.rst) files. The .rst files are plain text files that are formatted in a way that Sphinx understands and is able to turn them into HTML. Please see this [brief introduction](#) for more on reStructured text. You can modify the existing or create new .rst files into docs/source directory. When you are done editing, run bin/build_doc.bat on Windows or bin/build_doc.py on other systems to build the HTML pages to check the result before making a commit. The created pages are found in docs/build/html directory. After a commit, the User Guide is built automatically by readthedocs.org. The latest User Guide is available in <https://spine-toolbox.readthedocs.io/en/latest/>.

18.3.4 Contributing to the Spine Toolbox Graphical User Interface

If you want to change or add new widgets into the application, you need to use the bin\build_ui.bat (Windows) or bin/build_ui.py (other systems) scripts. The main design of the widgets should be done with Qt Designer (designer.exe or designer) that is included with PySide2. The files produced by Qt Designer are XML files (.ui). You can also embed graphics (e.g. icons, logos, etc.) into the application by using Qt Designer. When you are done modifying widgets in the designer, you need to run the build_ui script for the changes to take effect. This script uses tools provided in the PySide2 package to turn .ui files into Python files, in essence rebuilding the whole Spine Toolbox user interface.

Styling the widgets should be done with [Qt Style Sheets](#) in code. Please avoid using style sheets in Qt Designer.

18.3.5 Version Control Branching

Always make a new branch for your work, no matter how small. This makes it easy for others to take just that one set of changes from your repository, in case you have multiple unrelated changes floating around. A corollary: don't submit unrelated changes in the same branch/pull request! The maintainer shouldn't have to reject your awesome bugfix because the feature you put in with it needs more review.

Name your new branch descriptively, e.g. `issue#XXX-fixing-a-serious-bug` or `issue#ZZZ-cool-new-feature`. New branches should in general be based on the latest master branch. In case you want to include a new feature still in development, you can also start working from its branch. The developers will backport any relevant bug-fixes to previous or upcoming releases under preparation.

If you need to use code from an upstream branch, please use [git-rebase](#) *if you have not shared your work with others yet*. For example: You started working on an issue, but now the upstream branch (master) has some new commits you would like to have in your branch too. If you have not yet pushed your branch, you can now rebase your changes on top of the upstream branch:

```
$ git pull origin master:master
$ git checkout my_branch
$ git rebase master
```

Avoid merging the upstream branch to your issue branch if it's not necessary. This will lead to a more linear and cleaner history.

Finally, make a pull request from your branch so that the developers can review your changes. You might be asked to make additional changes or clarifications or add tests to prove the new feature works as intended.

18.3.6 Test-driven development is your friend

Any bug fix that does not include a test proving the existence of the bug being fixed, may be suspect. Ditto for new features that can't prove they actually work.

It is recommended to use test-first development as it really helps make features better designed and identifies potential edge cases earlier instead of later. Writing tests before the implementation is strongly encouraged.

See [Unit Testing Guidelines](#) for more information.

18.3.7 Full example

Here's an example workflow. Your username is `yourname` and you're submitting a basic bugfix.

Preparing your Fork

1. Click 'Fork' on Github, creating e.g. `yourname/Spine-Toolbox`
2. Clone your project: `git clone git@github.com:yourname/Spine-Toolbox`
3. `cd Spine-Toolbox`
4. Create a virtual environment and install requirements
5. Create a branch: `git checkout -b foo-the-bars master`

Making your Changes

1. Add an entry to `CHANGELOG.md`.
2. Write tests expecting the correct/fixed functionality; make sure they fail.
3. Hack, hack, hack.
4. Run tests again, making sure they pass.
5. Commit your changes: `git commit -m "Foo the bars"`

Creating Pull Requests

1. Push your commit to get it back up to your fork: `git push origin HEAD`
2. Visit Github, click handy 'Pull request' button that it will make upon noticing your new branch.
3. In the description field, write down issue number (if submitting code fixing an existing issue) or describe the issue + your fix (if submitting a wholly new bugfix).
4. Hit 'submit'! And please be patient - the maintainers will get to you when they can.

18.4 References

DEVELOPER DOCUMENTATION

Here you can find developer specific documentation on Spine Toolbox.

19.1 UI Guidelines

19.1.1 Keyboard shortcuts

Qt has a [list](#) of ‘standard’ keyboard shortcuts which can be used for inspiration.

- **F2**: edit current value in-place
- **Alt+F2**: open separate editor (e.g. Parameter value editor)
- **F3**: search
- **Alt+F4**: quit, close without saving changes
- **Esc**: close, exit without saving changes
- **Ctrl+Enter**: accept dialog

19.1.2 Action names

- **Edit...** should open an external editor, e.g. Parameter value editor in Database editor.

19.2 Unit Testing Guidelines

19.2.1 Test modules, directories

Spine project uses Python standard `unittest` framework for testing. The tests are organized into Python modules starting with the prefix `test_` under `<project root>/tests/`. The structure of `tests/` mirrors that of the package being tested. Note that all subdirectories containing test modules under `tests/` must have an (empty) `__init__.py` which makes them part of the project’s test package.

While there are no strict rules on how to name the individual test modules except for the `test_` prefix, `test_<module_name>.py` is preferred.

19.2.2 Running the tests

Tests are run as a GitHub action whenever a branch is pushed to GitHub. This process is configured by `<project root>/github/workflows/unittest_runner.yml`

To execute the tests manually, run `python -m unittest discover` in project's root.

19.2.3 Helpers

`mock_helpers` module in Toolbox's test package contains some helpful functions. Especially the methods to create mock `ToolboxUI` and `SpineToolboxProject` objects come very handy.

When instantiation of `QWidget` (this includes all GUI testing) is needed, Qt's main loop must be running during testing. This can be achieved by e.g. the `setUpClass` method below:

```
@classmethod
def setUpClass(cls):
    if not QApplication.instance():
        QApplication()
```

Sometimes an in-memory database can be handy because it does not require a temporary files or directories and it may be faster than an `.sqlite` file. To create an in-memory database, use `sqlite://` as the URL:

```
db_map = DiffDatabaseMapping("sqlite://", create=True)
```

Unfortunately, it is not possible to refer to the created database with the same URL prohibiting multiple database maps the access to the same in-memory database.

19.3 Execution Tests

Toolbox contains *execution tests* that test entire workflows in the headless mode. The tests can be found in `<toolbox repository root>/execution_tests/`. Execution tests are otherwise normal Toolbox projects except that the project root directories contain `__init__.py` and `execution_test.py` files. `__init__.py` makes the directory part of the execution test suite while `execution_test.py` contains actual test code. The tests utilize Python's `unittest` package so the test code is practically identical to any unit tests in Toolbox.

19.3.1 Executing the tests

Tests are run as a GitHub action whenever a branch is pushed to GitHub. This process is configured by `<project root>/github/workflows/executiontest_runner.yml`

To execute the tests manually, run `python -m unittest discover --pattern execution_test.py` in project's root.

19.4 Project Item Development

This document discusses the basics of *project item* development: what is required make one, how items interact with the Toolbox GUI and how they are executed.

The core of every project item consists of two classes: a *static* project item class which is responsible for integrating the item with the Toolbox GUI and an *executable* class which does the item's 'thing' and exists only during execution in Spine Engine. Some additional classes are needed for Toolbox to be able to instantiate project items and to communicate with the user via the Toolbox GUI.

Specifications are a way to make the settings of an item portable across projects. In a sense a specification is a template that can specialize an item for a specific purpose such as a Tool that runs certain model with known inputs and outputs. Items that support specifications need to implement some additional methods and classes.

19.4.1 Getting started

Probably the most convenient way to start developing a new project item is to work with a copy of some simple project item. For example, View provides a good starting point.

Project items are mostly self-contained Python packages. It is customary to structure the project item packages like the Toolbox itself: `mvcmodels` submodule for Qt's models, `ui` module for automatically generated UI forms and `widgets` for widgets' business logic. However, the only actual requirement is that Toolbox expects to find the item's factory and item info classes in the package's root modules as well as an `executable_item` module.

19.4.2 Item info

A subclass of `spine_engine.project_item.project_item_info.ProjectItemInfo` must be found in one of the root modules of an item's package. It is used by Toolbox to query the *type* and *category* of an item. Type identifies the project item while category is used by the Toolbox GUI to group project items with similar function. Categories are currently fixed and can be checked from `spine_items.category`.

19.4.3 Item Factory

The details of constructing a project item and related objects have been abstracted away from Toolbox by a factory that must be provided by every project item in a root module of the item's package. The factory is a subclass of `spinetoolbox.project_item.project_item_factory.ProjectItemFactory`. Note that methods in the factory that deal with specifications need to be implemented only by items that support them.

19.4.4 Executable item

A project item must have a root module called `executable_item` that contains a class named `ExecutableItem` which is a subclass of `spine_engine.project_item.executable_item_base.ExecutableItemBase`. `ExecutableItem` acts as an access point to Spine Engine and contains the item's execution logic.

19.4.5 Toolbox side project item

A project item must subclass `spinetoolbox.project_item.project_item.ProjectItem` and return the subclass in its factory's `item_class()` method. Also `make_item()` must return an instance of this class. This class forms the core of integrating the item with Toolbox.

19.4.6 Specifications

Items that support specifications need to subclass `spine_engine.project_item.project_item_specification_factory.ProjectItemSpecificationFactory` which provides an access point to Toolbox and Spine Engine to generate specifications. The factory must be called `SpecificationFactory` and be placed in `specification_factory` module under item package's root. The specification itself should be a subclass of `spine_engine.project_item.project_item_specification.ProjectItemSpecification`.

19.4.7 Toolbox GUI integration

`ProjectItemFactory.icon()` returns a URL to the item's icon resource. This is the item's 'symbol' shown e.g. on the main toolbar of Toolbox. It should not be confused with the actual icon on Design view which in turn is a subclass of `spinetoolbox.project_item.project_item_icon.ProjectItemIcon` and is returned by `ProjectItemFactory.make_icon()`.

When creating a new item on the Design view Toolbox shows the *Add item dialog* it gets from `ProjectItemFactory.make_add_item_widget()`. Toolbox provides `spinetoolbox.widgets.add_project_item_widget.AddProjectItemWidget` which is a general purpose widget for this purpose though project items are free to implement their own widgets as needed.

Once the item is on the Design view, the main interaction with it goes through the properties widget which is created by `ProjectItemFactory.make_properties_widget()`. The properties widget should have all controls needed to set up the item.

19.4.8 Saving and restoring project items

Project items are saved in JSON format as part of the `project.json` file. Item saving is handled by `ProjectItem.item_dict()` which should return a JSON compatible dict and contain at least the information returned by the base class method.

File system paths are handled specifically during saving: all paths outside the project directory should be absolute while the paths in the project directory should be relative. This is to enable self-contained projects which include all needed files and can be easily transferred from system to system. As such, paths are saved as special dictionaries. `spine_engine.utils.serialization.serialize_path()`, `spine_engine.utils.serialization.serialize_url()` and `spine_engine.utils.serialization.deserialize_path()` help with dealing with the paths.

`ProjectItem.from_dict()` is responsible for restoring a saved project item from the dictionary. `ProjectItem.parse_item_dict()` can help to deserialize the basic data needed by the base class.

19.4.9 Passing data between items: resources

Project items share data by files or via databases. One item writes a file which is then read by another item. **Project item resources** are used to communicate the URLs of these files and databases.

Resources are instances of the `spine.engine.project_item.project_item_resource.ProjectItemResource` class.

Both static items and their executable counterparts pass resources. The major difference is that static item's may pass resource *promises* such as files that are generated during the execution. The full path to the promised files or even their final names may not be known until the items are executed.

During execution resources are propagated only to item's *direct* predecessors and successors. Static items offer their resources to direct successors only. Resources that are communicated to successor items are basically output files that the successor items can use for input. Currently, the only resource that is propagated to predecessor items is database URLs by Data Store project items. As Data Stores leave the responsibility of writing to the database to other items it has to tell these items where to write their output data.

The table below lists the resources each project item type provides during execution.

Item	Notes	Provides to predecessor	Provides to successor
Data Connection	¹	n/a	File URLs
Data Store	²	Database URL	Database URL
Data Transformer	³	n/a	Database URL
Exporter		n/a	File URLs
Importer		n/a	n/a
Merger		n/a	n/a
Tool	⁴	n/a	File URLs
View		n/a	n/a

The table below lists the resources that might be used by each item type during execution.

Item	Notes	Accepts from predecessor	Accepts from successor
Data Connection		n/a	n/a
Data Store		n/a	n/a
Data Transformer		Database URL	n/a
Exporter		Database URL	n/a
Importer	⁵	File URLs	Database URL
Merger		Database URL	Database URL
Tool	⁶	File URLs, database URLs	Database URLs
View		Database URLs	n/a

¹ Data connection provides paths to local files.

² Data Store provides a database URL to direct successors and predecessors. Note, that this is the only project item that provides resources to it's predecessors.

³ Data Transformer provides its predecessors' database URLs modified by transformation configuration embedded in the URL.

⁴ Tool's output files are specified by a *Tool specification*.

⁵ Importer requires a database URL from its successor for writing the mapped data. This can be provided by a Data Store.

⁶ *Tool specification* specifies tool's optional and required input files. Database URLs can be passed to the tool *program* via command line arguments but are otherwise ignored by the Tool project item. Currently, there is no mechanism to know if a URL is actually required by a tool *program*. For more information, see [Tool Specification Editor](#).

19.4.10 Execution

Spine Engine instantiates the executable items in a DAG before the execution starts. Then, Engine declares forward and backward resources for each item using `ExecutableItemBase.output_resources()`. During execution, `ExecutableItemBase.execute()` is invoked with lists of available resources if an item is selected for execution. Otherwise, `ExecutableItemBase.exclude_execution()` is called.

19.5 Publishing to PyPI

This document describes the prerequisites and workflow to publish Spine Toolbox to [The Python Package Index \(PyPI\)](#).

19.5.1 Versioning of Spine Toolbox packages

Spine Toolbox packages use the latest Git tag to dynamically generate the version number. During the build process Git tags of the form `X.Y.Z` are sorted and the latest is used to generate the package version. If the tip of the current branch (HEAD) is at a tag, the version number is the tag. However, if there have been commits since the latest tag, the next version is guessed and a `dev??-*` component is included (e.g. `'0.7.0.dev77+gf9538fee.d20230816'`). Note that the `dev*` component also includes an indication of the number of commits since the last tag.

Under this scheme, the release process is simply to create a new Git tag, and publish it. However since the different Spine packages depend on each other, you need to update the different version number requirements in their respective `pyproject.toml` files. This can be done conveniently by using the CLI tools available in the [spine-conductor](#) repo.

19.5.2 Creating Git tags and publishing to PyPI

1. Check out the [spine-conductor](#) repo, and install it, either in a virtual environment or using `pipx`.
2. You can create a TOML configuration file as mentioned in the README of the repo; say `release.toml`. Something like the sample below should work.

Listing 1: `release.toml`

```
[tool.conductor]
packagename_regex = "spine(toolbox|db){0,1}{_[-][a-z]+)" # package name on PyPI

[tool.conductor.dependency_graph]
spinetoolbox = ["spine_items", "spine_engine", "spinedb_api"]
spine_items = ["spinetoolbox", "spine_engine", "spinedb_api"]
spine_engine = ["spinedb_api"]
spinedb_api = []

[tool.conductor.repos]
spinetoolbox = "."
spine_items = "venv/src/spine-items"
spine_engine = "venv/src/spine-engine"
spinedb_api = "venv/src/spinedb-api"

# # default
# [tool.conductor.branches]
# spinetoolbox = "master"
# spine_items = "master"
```

(continues on next page)

(continued from previous page)

```
# spine_engine = "master"
# spinedb_api = "master"
```

- Now you can create a release by calling the `conduct release -c release.toml` command with the TOML file as config. This starts a guided session where the `spine-conductor` CLI tool deduces the next version numbers from existing Git tags, updates the corresponding `pyproject.toml` files in all the repos to reflect the new package versions, and finally prompts you to add any edited files, and create the new Git tag. A typical session would like this:

Listing 2: A typical release session; note the JSON summary in the end.

```
$ cd /path/to/repo/Spine-Toolbox
$ conduct release --bump patch -c release.toml # or include in pyproject.toml
Repository: /path/to/repo/Spine-Toolbox
## master...origin/master
M pyproject.toml (1)
Select the files to add (comma/space separated list): 1
Creating tag: 0.6.19 @ 034fb4b
Repository: /path/to/repo/venv/src/spine-items
## master...origin/master
M pyproject.toml (1)
Select the files to add (comma/space separated list): 1
Creating tag: 0.20.1 @ 5848e25
Repository: /path/to/repo/venv/src/spine-engine
## master...origin/master
M pyproject.toml (1)
Select the files to add (comma/space separated list): 1
Creating tag: 0.22.1 @ e312db2
Repository: /path/to/repo/venv/src/spinedb-api
## master...origin/master
Select the files to add (comma/space separated list):
Creating tag: 0.29.1 @ d9ed86e

Package Tags summary    'pkgtags.json':
{
  "Spine-Toolbox": "0.6.19",
  "spine-items": "0.20.1",
  "spine-engine": "0.22.1",
  "Spine-Database-API": "0.29.1"
}
```

If the session completes successfully, you will see a session summary with the newest Git tags that were created for each package.

- Push the newly created tags to GitHub. On sh-like shells like: `bash`, `zsh`, or `git-bash` (Windows):

```
for repo in . venv/src/{spinedb-api,spine-{items,engine}}; do
  pushd $repo;
  git push origin master --tags;
  popd
done
```

With Powershell on Windows, something like this should work:

```

"." , "venv/src/spinedb-api", "venv/src/spine-items", "venv/src/spine-engine" | % {
  pushd $_;
  git push origin master --tags;
  popd;
}

```

- Now you can trigger the workflow to publish the packages to PyPI either by using GitHub CLI, or from the [workflow dispatch menu](#) in the `spine-conductor` repo.

```

cat pkgtags.json | gh workflow run --repo spine-tools/spine-conductor test-n-
➔publish.yml --json

```

If you are using the [workflow dispatch menu](#), make sure you input the exact same package versions as shown in the summary.

Done! **Note:** Soon, (4) & (5) will be wrapped in a separate command provided by `spine-conductor`.

19.5.3 The `release.toml` file

The config file is a standard TOML file conformant with `pyproject.toml`, meaning all configuration goes under the section `tool.conductor`. The configuration is split into 4 sections: a regex to identify our packages, dependency graph between our packages, path to the repos to be used for the release, and the branches to be used (optional).

- You can specify a regular expression that will be used to identify “our” packages. Something like the following should work:

Listing 3: Spine package name regular expression

```

[tool.conductor]
package_name_regex = "spine(toolbox|(db){0,1}[_-][a-z]+)" # package name on PyPI

```

Note that PyPI treats - (hyphen) and _ (underscore) as equivalent in package names; i.e. `spinedb_api` and `spinedb-api` are equivalent, the regex should accomodate that.

- The dependency graph between our packages should be specified under the `dependency_graph` section:

Listing 4: Spine package dependency graph

```

[tool.conductor.dependency_graph]
spinetoolbox = ["spine_items", "spine_engine", "spinedb_api"]
spine_items = ["spinetoolbox", "spine_engine", "spinedb_api"]
spine_engine = ["spinedb_api"]
spinedb_api = []

```

Essentially it is a mapping of the “primary” package, and a list of its Spine dependencies.

- Point to the repository directories *relative* to your current working directory. The following example would be valid if you are preparing the release from the Toolbox repo, and the other Spine package repos are in the virtual environment.

Listing 5: Repository paths

```

[tool.conductor.repos]
spinetoolbox = "."
spine_items = "venv/src/spine-items"

```

(continues on next page)

(continued from previous page)

```
spine_engine = "venv/src/spine-engine"
spinedb_api  = "venv/src/spinedb-api"
```

4. You can also specify the branches for each repository that should be used for the release. This section is optional, and if left unspecified, the branch name is assumed to be `master`.

Listing 6: Release branches on Spine repositories

```
# default: master
[tool.conductor.branches]
spinetoolbox = "release"
spine_items  = "release"
spine_engine = "release"
spinedb_api  = "release"
```

19.5.4 Manual release (in case of emergency)

This section documents what the `spine-conductor` CLI tool does under the hood. It is here in case of an emergency (e.g. there's a bug), and the release has to be done manually.

As mentioned earlier, the package version is now derived from Git tags. However, because of the internal dependency between the Spine packages, the versions of the dependencies have to be synchronised with the new version. The steps are as follows:

1. Determine the next version for each Spine package. This can be done manually with Git, or you can use `setuptools_scm` in a Python REPL.
 - You can run `git describe --tags` in the repo. This will print out the latest tag followed by a trailer with metadata on distance from the tag; something like this: `0.6.18-100-g411c13e1`. If you want to make a patch release, the next version would be `0.6.19` and a minor release would be `0.7.0`. Repeat this process for all 4 Spine repos.
 - If using a Python REPL, you can do the following for a minor release:

```
>>> from setuptools_scm import get_version
>>> get_version(".", version_scheme="release-branch-semver")
'0.7.0.dev100+g411c13e1.d20230823'
```

For a patch release, do the following:

```
>>> get_version(".", version_scheme="guess-next-dev")
'0.6.19.dev100+g411c13e1.d20230823'
```

Note the first argument to `get_version` is the path to the repository. The above examples assume the repository is your current directory. If it's not, you can provide the path as the first argument.

2. Once the new package versions are determined, you need to edit the `pyproject.toml` files in all 4 repositories with the correct version numbers. For example, in the `Spine-Toolbox` repo if you were to do a minor release, i.e. `0.6.18` → `0.7.0`, the following change would be sufficient:

Listing 7: Example edit to `pyproject.toml` for Spine-Toolbox

```
diff --git a/pyproject.toml b/pyproject.toml
index bd38a2b7..dd9c228e 100644
```

(continues on next page)

(continued from previous page)

```

--- a/pyproject.toml
+++ b/pyproject.toml
@@ -20,8 +20,8 @@ dependencies = [
    "jupyter-client >=6.0",
    "qtconsole >=5.1",
    "sqlalchemy >=1.3",
-   "spinedb_api >=0.29.0",
-   "spine_engine >=0.22.0",
+   "spinedb_api >=0.30.0",
+   "spine_engine >=0.23.0",
    "numpy >=1.20.2",
    "matplotlib >= 3.5",
    "scipy >=1.7.1",
@@ -30,7 +30,7 @@ dependencies = [
    "pygments >=2.8",
    "jill >=0.9.2",
    "pyzmq >=21.0",
-   "spine-items >= 0.20.0",
+   "spine-items >= 0.21.0",
    ]

[project.urls]

```

3. After updating the `pyproject.toml` file for all 4 Spine repos as above, add and commit the changes in all repos:

```
git commit -i pyproject.toml -m "Release 0.7.0"
```

4. Create a Git tag on the latest commit:

```
git tag 0.7.0 HEAD
```

5. Push the tags to GitHub. On sh-like shells like: `bash`, `zsh`, or `git-bash` (Windows):

Listing 8: Recipe to push Git tags to GitHub on sh-like shells (`bash`, `zsh`, `git-bash`)

```

for repo in . venv/src/{spinedb-api,spine-{items,engine}}; do
    pushd $repo;
    git push origin master --tags;
    popd
done

```

With Powershell on Windows:

Listing 9: Recipe to push Git tags to GitHub on Powershell

```

"." , "venv/src/spinedb-api", "venv/src/spine-items", "venv/src/spine-engine" | % {
    pushd $_;
    git push origin master --tags;
    popd;
}

```

6. Now you can trigger the workflow to publish the packages to PyPI from the [workflow dispatch menu](#) in the [spine-conductor](#) repo. Ensure you input the exact same package versions as in the tags.

7. In case the workflow above also fails, you have to build the source distribution archive and wheels locally and upload to PyPI manually.

To build, ensure you have `build` installed. The build backend ensures build isolation, and reproducibility of the wheels given a source distribution.

Listing 10: Build distribution archives and wheels

```
python -m pip install build
python -m build
```

Once the build completes, you can find the source tarball and the wheel in `dist/`. Now you may upload these files to PyPI.

It is good practise to first test using TestPyPI before uploading to PyPI, since releases on PyPI are read-only. You want to avoid mistakes.

[Register an account](#) and ask some of the owners of the [Spine Toolbox package](#) (or other relevant package) to add you as a maintainer.

Upload the distribution using

```
twine upload --repository testpypi dist/*
```

See [Using TestPyPI](#) for more information. To avoid entering your username and password every time, see [Keyring support in twine documentation](#) or generate an [API key](#). If everything went smoothly, you are ready to upload the real index. Again, you need to register to PyPI and ask to become a maintainer of the package you want to upload to. Upload the distribution using

```
$ twine upload dist/*
```

Done! Now fix the bug that forced you to do the manual release ;)

API REFERENCE

This page contains auto-generated API reference documentation¹.

20.1 spinetoolbox

spinetoolbox package.

20.1.1 Subpackages

`spinetoolbox.mvcmodels`

Modules in this package contain classes that represent Spine Toolbox's models (internal data structures) in the Model-View-Controller design pattern. The model classes define an interface that is used by views and delegates to access data in the application.

Submodules

`spinetoolbox.mvcmodels.array_model`

Contains model for the Array editor widget.

Module Contents

Classes

<i>ArrayModel</i>	Model for the Array parameter_value type.
-----------------------------------	---

class `spinetoolbox.mvcmodels.array_model.ArrayModel`(*parent*)

Bases: `PySide6.QtCore.QAbstractTableModel`

Model for the Array parameter_value type.

Even if the array is empty this model's `rowCount()` will still return 1. This is to show an empty row in the table view.

¹ Created with `sphinx-autoapi`

Parameters

parent (*QObject*) – parent object

array()

Returns the array modeled by this model.

batch_set_data(*indexes, values*)

Sets data at multiple indexes at once.

Parameters

- **indexes** (*list of QModelIndex*) – indexes to set
- **values** (*list of str*) – values corresponding to the indexes

columnCount(*parent=QModelIndex()*)

Returns 2.

_convert_to_data_type(*indexes, values*)

Converts values from string to current data type filtering failed conversions.

Parameters

- **indexes** (*list of QModelIndex*) – indexes
- **values** (*list of str*) – values to convert

Returns

indexes and converted values

Return type

tuple

data(*index, role=Qt.ItemDataRole.DisplayRole*)

Returns model's data for given role.

flags(*index*)

Returns table cell's flags.

headerData(*section, orientation, role=Qt.ItemDataRole.DisplayRole*)

Returns header data.

insertRows(*row, count, parent=QModelIndex()*)

Inserts rows to the array.

is_expense_row(*row*)

Returns True if row is the expense row.

Parameters

row (*int*) – a row

Returns

True is row is expense row, False otherwise

Return type

bool

removeRows(*row, count, parent=QModelIndex()*)

Removes rows from the array.

reset(*value*)

Resets the model to a new array.

Parameters

value (*Array*) – a new array to model

rowCount(*parent=QModelIndex()*)

Returns the length of the array.

Note: returns 1 even if the array is empty.

set_array_type(*new_type*)

Changes the data type of array's elements.

Parameters

new_type (*Type*) – new element type

setHeaderData(*section, orientation, value, role=Qt.ItemDataRole.EditRole*)

setData(*index, value, role=Qt.ItemDataRole.EditRole*)

Sets the value at given index.

spinetoolbox.mvcmodels.compound_table_model

Models that vertically concatenate two or more table models.

Module Contents

Classes

<i>CompoundTableModel</i>	A model that concatenates several sub table models vertically.
<i>CompoundWithEmptyTableModel</i>	A compound parameter table model where the last model is an empty row model.

class spinetoolbox.mvcmodels.compound_table_model.**CompoundTableModel**(*parent=None, header=None*)

Bases: *spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel*

A model that concatenates several sub table models vertically.

Initializes model.

Parameters

- **parent** (*QObject, optional*) – the parent object
- **header** (*list of str, optional*) – header labels

refreshed

map_to_sub(*index*)

Returns an equivalent submodel index.

Parameters

index (*QModelIndex*) – the compound model index.

Returns

the equivalent index in one of the submodels

Return type

QModelIndex

map_from_sub(*sub_model*, *sub_index*)

Returns an equivalent compound model index.

Parameters

- **sub_model** ([MinimalTableModel](#)) – the submodel
- **sub_index** ([QModelIndex](#)) – the submodel index.

Returns

the equivalent index in the compound model

Return type

QModelIndex

item_at_row(*row*)

Returns the item at given row.

Parameters

row (*int*) –

Returns

object

sub_model_at_row(*row*)

Returns the submodel corresponding to the given row in the compound model.

Parameters

row (*int*) –

Returns

MinimalTableModel

sub_model_row(*row*)

Calculates sub model row.

Parameters

row (*int*) – row in compound model

Returns

row in sub model

Return type

int

refresh()

Refreshes the layout by computing a new row map.

_do_refresh()

Recomputes the row and inverse row maps.

_append_row_map(*row_map*)

Appends given row map to the tail of the model.

Parameters

row_map (*list*) – tuples (model, row number)

`_row_map_iterator_for_model(model)`

Yields row map for given model. The base class implementation just yields all model rows.

Parameters

model (`MinimalTableModel`) –

Yields

tuple – (model, row number)

`_row_map_for_model(model)`

Returns row map for given model. The base class implementation just returns all model rows.

Parameters

model (`MinimalTableModel`) –

Returns

tuples (model, row number)

Return type

list

`canFetchMore(parent)`

Returns True if any of the submodels that haven't been fetched yet can fetch more.

`fetchMore(parent)`

Fetches the next sub model and increments the fetched counter.

`flags(index)`

Return index flags.

`data(index, role=Qt.ItemDataRole.DisplayRole)`

Returns the data stored under the given role for the item referred to by the index.

Parameters

- **index** (`QModelIndex`) – Index of item
- **role** (`int`) – Data role

Returns

Item data for given role.

`rowCount(parent=QModelIndex())`

Returns the sum of rows in all models.

`batch_set_data(indexes, data)`

Sets data for indexes in batch. Distributes indexes and values among the different submodels and calls `batch_set_data` on each of them.

`insertRows(row, count, parent=QModelIndex())`

Inserts count rows after the given row under the given parent. Localizes the appropriate submodel and calls `insertRows` on it.

`removeRows(row, count, parent=QModelIndex())`

Removes count rows starting with the given row under parent. Localizes the appropriate submodels and calls `removeRows` on it.

`class spinetoolbox.mvcmodels.compound_table_model.CompoundWithEmptyTableModel` (*parent=None*,
header=None)

Bases: [*CompoundTableModel*](#)

A compound parameter table model where the last model is an empty row model.

Initializes model.

Parameters

- **parent** (*QObject*, *optional*) – the parent object
- **header** (*list of str*, *optional*) – header labels

property `single_models`

property `empty_model`

abstract `_create_empty_model()`

Creates and returns an empty model.

Returns

model

Return type

[*EmptyRowModel*](#)

init_model()

Initializes the compound model.

Basically populates the `sub_models` list attribute with the result of `_create_empty_model`.

_connect_single_model(*model*)

Connects signals so changes in the submodels are acknowledged by the compound.

_recompute_empty_row_map()

Recomputes the part of the row map corresponding to the empty model.

_handle_empty_rows_removed(*parent*, *empty_first*, *empty_last*)

Updates `row_map` when rows are removed from the empty model.

_handle_empty_rows_inserted(*parent*, *empty_first*, *empty_last*)

Runs when rows are inserted to the empty model. Updates `row_map`, then emits `rowsInserted` so the new rows become visible.

_handle_single_model_about_to_be_reset(*model*)

Runs when given model is about to reset.

_handle_single_model_reset(*model*)

Runs when given model is reset.

_refresh_single_model(*model*)

_get_insert_position(*model*)

_insert_single_model(*model*)

_get_row_for_insertion(*pos*)

_insert_row_map(*pos*, *single_row_map*)

clear_model()

Clears the model.

`spinetoolbox.mvcmodels.empty_row_model`

Contains a table model with an empty last row.

Module Contents

Classes

`EmptyRowModel`

A table model with a last empty row.

class `spinetoolbox.mvcmodels.empty_row_model.EmptyRowModel`(*parent=None, header=None*)

Bases: *`spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel`*

A table model with a last empty row.

Init class.

canFetchMore(*_parent*)

Return True if the model hasn't been fetched.

fetchMore(*parent*)

Fetch data and use it to reset the model.

flags(*index*)

Return default flags except if forcing defaults.

set_default_row(***kwargs*)

Set default row data.

clear()

Clear all data in model.

reset_model(*main_data=None*)

Reset model.

_handle_data_changed(*top_left, bottom_right, roles=None*)

Insert a new last empty row in case the previous one has been filled with any data other than the defaults.

removeRows(*row, count, parent=QModelIndex()*)

Don't remove the last empty row.

_handle_rows_inserted(*parent, first, last*)

Handle rowsInserted signal.

set_rows_to_default(*first, last=None*)

Set default data in newly inserted rows.

spinetoolbox.mvcmodels.file_list_models

Common models. Contains a generic File list model and an Item for that model. Used by the Importer and Tool project items but this may be handy for other project items as well.

Module Contents

Classes

<i>FileListModel</i>	A model for files to be shown in a file tree view.
<i>CommandLineArgItem</i>	
<i>NewCommandLineArgItem</i>	
<i>CommandLineArgsModel</i>	
<i>JumpCommandLineArgsModel</i>	

class spinetoolbox.mvcmodels.file_list_models.**FileListModel**(*header_label*='', *draggable*=False)

Bases: PySide6.QtCore.QAbstractItemModel

A model for files to be shown in a file tree view.

Parameters

- **header_label** (*str*) – header label
- **draggable** (*bool*) – if True, the top level items are drag and droppable

FileItem

PackItem

rowCount(*parent*=QModelIndex())

columnCount(*parent*=QModelIndex())

headerData(*section*, *orientation*, *role*=Qt.ItemDataRole.DisplayRole)

Returns header information.

data(*index*, *role*=Qt.ItemDataRole.DisplayRole)

Returns data associated with given role at given index.

flags(*index*)

mimeData(*indexes*)

resource(*index*)

Returns the resource at given index.

Parameters

index (QModelIndex) – index

Returns

resource

Return type

ProjectItemResource

parent(*index*)**index**(*row*, *column*, *parent*=QModelIndex())**update**(*resources*)

Updates the model according to given list of resources.

Parameters**resources** (*Iterable of ProjectItemResource*) – resources**duplicate_paths**()

Checks if resources in the model have duplicate file paths.

Returns

set of duplicate file paths

Return type

set of str

_pack_index(*pack_label*)

Finds a pack's index in pack resources list.

Parameters**pack_label** (*str*) – pack label**Returns**

index to pack resources list

Return type

int

```
class spinetoolbox.mvcmodels.file_list_models.CommandLineArgItem(text="", rank=None,
                                                                    selectable=False,
                                                                    editable=False,
                                                                    drag_enabled=False,
                                                                    drop_enabled=False)
```

Bases: PySide6.QtGui.QStandardItem

set_rank(*rank*)**static _make_icon**(*rank*=None)**setData**(*value*, *role*=Qt.ItemDataRole.UserRole + 1)

```
class spinetoolbox.mvcmodels.file_list_models.NewCommandLineArgItem
```

Bases: [CommandLineArgItem](#)**setData**(*value*, *role*=Qt.ItemDataRole.UserRole + 1)

```
class spinetoolbox.mvcmodels.file_list_models.CommandLineArgsModel(parent=None)
```

Bases: PySide6.QtGui.QStandardItemModel

property args**args_updated****append_arg**(*arg*)

```
replace_arg(row, arg)
```

```
mimeData(indexes)
```

```
dropMimeData(data, drop_action, row, column, parent)
```

```
static _reset_root(root, args, child_params, has_empty_row=True)
```

```
class spinetoolbox.mvcmodels.file_list_models.JumpCommandLineArgsModel(parent=None)
```

```
Bases: CommandLineArgsModel
```

```
reset_model(args)
```

```
canDropMimeData(data, drop_action, row, column, parent)
```

```
spinetoolbox.mvcmodels.filter_checkbox_list_model
```

Provides FilterCheckboxListModel for FilterWidget.

Module Contents

Classes

<i>SimpleFilterCheckboxListModel</i>	Init class.
<i>LazyFilterCheckboxListModel</i>	Extends SimpleFilterCheckboxListModel to allow for lazy loading in synch with another model.
<i>DataToValueFilterCheckboxListModel</i>	Extends SimpleFilterCheckboxListModel to allow for translating internal data to a value for display role.

```
class spinetoolbox.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel(parent,
                                                                                       show_empty=True)
```

```
Bases: PySide6.QtCore.QAbstractListModel
```

```
Init class.
```

Parameters

```
parent (QWidget) –
```

```
property _show_empty
```

```
property _show_add_to_selection
```

```
_SELECT_ALL_STR = '(Select all)'
```

```
_SELECT_ALL_FILTERED_STR = '(Select all filtered)'
```

```
_EMPTY_STR = '(Empty)'
```

```
_ADD_TO_SELECTION_STR = 'Add current selection to filter'
```

```
reset_selection()
```

```
_handle_select_all_clicked()
```

`_check_all_selected()`

`rowCount(parent=QModelIndex())`

`data(index, role=Qt.ItemDataRole.DisplayRole)`

`_handle_index_clicked(index)`

`set_list(data, all_selected=True)`

`filter_by_condition(condition)`

Updates selected items by applying a condition.

Parameters

condition (*function*) – Filter acceptance condition.

`set_selected(selected, select_empty=None)`

`get_selected()`

`get_not_selected()`

`set_filter(filter_expression)`

`search_filter_expression(item)`

`apply_filter()`

`_remove_and_add_filtered()`

`_remove_and_replace_filtered()`

`remove_filter()`

`_do_add_items(data)`

`add_items(data, selected=None)`

`remove_items(data)`

```
class spinetoolbox.mvcmodels.filter_checkbox_list_model.LazyFilterCheckboxListModel(parent,
                                                                                   db_mgr,
                                                                                   db_maps,
                                                                                   fetch_parent,
                                                                                   show_empty=True)
```

Bases: [SimpleFilterCheckboxListModel](#)

Extends SimpleFilterCheckboxListModel to allow for lazy loading in synch with another model.

Init class.

Parameters

- **parent** ([SpineDBEditor](#)) –
- **fetch_parent** ([FetchParent](#)) –

`canFetchMore(_parent)`

`fetchMore(_parent)`

_do_add_items(*data*)

Adds items so the list is always sorted, while assuming that both existing and new items are sorted.

class spinetoolbox.mvcmodels.filter_checkbox_list_model.**DataToValueFilterCheckboxListModel**(*parent*,
data_to_value,
show_empty=Tr)

Bases: *SimpleFilterCheckboxListModel*

Extends SimpleFilterCheckboxListModel to allow for translating internal data to a value for display role.

Init class.

Parameters

- **parent** (*SpineDBEditor*) –
- **data_to_value** (*method*) – a method to translate item data to a value for display role

data(*index*, *role=Qt.ItemDataRole.DisplayRole*)

search_filter_expression(*item*)

spinetoolbox.mvcmodels.filter_execution_model

Contains FilterExecutionModel.

Module Contents

Classes

FilterExecutionModel

class spinetoolbox.mvcmodels.filter_execution_model.**FilterExecutionModel**

Bases: PySide6.QtCore.QAbstractListModel

_filter_consoles

reset_model(*filter_consoles*)

rowCount(*parent=QModelIndex()*)

headerData(*section*, *orientation*, *role=Qt.ItemDataRole.DisplayRole*)

data(*index*, *role=Qt.ItemDataRole.DisplayRole*)

find_index(*console_key*)

get_console(*filter_id*)

spinetoolbox.mvcmodels.indexed_value_table_model

A model for indexed parameter values, used by the parameter_value editors.

Module Contents**Classes**

IndexedValueTableModel

A base class for time pattern and time series models.

Attributes

EXPANSE_COLOR

spinetoolbox.mvcmodels.indexed_value_table_model.EXPANSE_COLOR

class spinetoolbox.mvcmodels.indexed_value_table_model.IndexedValueTableModel(*value*, *parent*)

Bases: PySide6.QtCore.QAbstractTableModel

A base class for time pattern and time series models.

Parameters

- **value** (*IndexedValue*) – a parameter_value
- **parent** (*QObject*) – parent object

property value

Returns the parameter_value associated with the model.

columnCount(*parent=QModelIndex()*)

Returns the number of columns which is two.

data(*index*, *role=Qt.ItemDataRole.DisplayRole*)

Returns the data at index for given role.

headerData(*section*, *orientation=Qt.Orientation.Horizontal*, *role=Qt.ItemDataRole.DisplayRole*)

Returns a header.

is_expense_row(*row*)

Returns True if row is the expense row.

Parameters

- **row** (*int*) – a row

Returns

True if row is the expense row, False otherwise

Return type

bool

reset(*value*)

Resets the model.

rowCount(*parent=QModelIndex()*)

Returns the number of rows.

setHeaderData(*section, orientation, value, role=Qt.ItemDataRole.EditRole*)

spinetoolbox.mvcmodels.map_model

A model for maps, used by the `parameter_value` editors.

Module Contents

Classes

<i>MapModel</i>	A model for Map type parameter values.
-----------------	--

Functions

<i>_rows_to_dict</i> (rows)	Turns table into nested dictionaries.
<i>_reconstruct_map</i> (tree)	Constructs a <code>Map</code> from a nested dictionary.
<i>_data_length</i> (row)	Counts the number of non-empty elements at the beginning of row.
<i>_gather_index_names</i> (map_value)	Collects index names from <code>Map</code> .
<i>_apply_index_names</i> (map_value, index_names)	Applies index names to <code>Map</code> .
<i>_numpy_string_to_python_strings</i> (rows)	Converts instances of <code>numpy.str_</code> to regular Python strings.

Attributes

<i>empty</i>	Sentinel for empty cells.
--------------	---------------------------

spinetoolbox.mvcmodels.map_model.empty

Sentinel for empty cells.

class `spinetoolbox.mvcmodels.map_model.MapModel`(*map_value, parent*)

Bases: `PySide6.QtCore.QAbstractTableModel`

A model for Map type parameter values.

This model represents the `Map` as a 2D table. Each row consists of one or more index columns and a value column. The last columns of a row are padded with Nones.

Example

```
Map {
  "A": 1.0
  "B": Map {"a": -1.0}
  "C": 3.0
}
```

The table corresponding to the above map:

"A"	1.0	None
"B"	"a"	-1.0
"C"	3.0	None

Parameters

- **map_value** (*Map*) – a map
- **parent** (*QObject*) – parent object

append_column()

Appends a new column to the right.

clear(*indexes*)

Clears table cells.

Parameters

indexes (*list of QModelIndex*) – indexes to clear

columnCount(*index=QModelIndex()*)

Returns the number of columns in this model.

convert_leaf_maps()

data(*index, role=Qt.ItemDataRole.DisplayRole*)

Returns the data associated with the given role.

flags(*index*)

Returns flags at index.

headerData(*section, orientation, role=Qt.ItemDataRole.DisplayRole*)

Returns row numbers for vertical headers and column titles for horizontal ones.

insertColumns(*column, count, parent=QModelIndex()*)

Inserts new columns into the map.

Parameters

- **column** (*int*) – column index where to insert
- **count** (*int*) – number of new columns
- **parent** (*QModelIndex*) – ignored

Returns

True if insertion was successful, False otherwise

Return type

bool

insertRows(*row*, *count*, *parent*=*QModelIndex()*)

Inserts new rows into the map.

Parameters

- **row** (*int*) – an index where to insert the new data
- **count** (*int*) – number of rows to insert
- **parent** (*QModelIndex*) – an index to a parent model

Returns

True if the operation was successful

Return type

bool

is_leaf_value(*index*)

Checks if given model index contains a leaf value.

Parameters

index (*QModelIndex*) – index to check

Returns

True if index points to leaf value, False otherwise

Return type

bool

_is_in_expense(*row*, *column*)

Returns True, if given row and column is in the right or bottom ‘expanding’ zone.

Parameters

- **row** (*int*) – row index
- **column** (*int*) – column index

Returns

True if the cell is in the expense, False otherwise

Return type

bool

is_expense_column(*column*)

Returns True if given column is the expense column.

Parameters

column (*int*) – column

Returns

True if column is expense column, False otherwise

Return type

bool

is_expense_row(*row*)

Returns True if given row is the expense row.

Parameters

row (*int*) – row

Returns

True if row is the expense row, False otherwise

Return type

bool

removeColumns(*column*, *count*, *parent*=*QModelIndex()*)

Removes columns from the map.

Parameters

- **column** (*int*) – first column to remove
- **count** (*int*) – number of columns to remove
- **parent** (*QModelIndex*) – an index to a parent model

Returns

True if the operation was successful

removeRows(*row*, *count*, *parent*=*QModelIndex()*)

Removes rows from the map.

Parameters

- **row** (*int*) – first row to remove
- **count** (*int*) – number of rows to remove
- **parent** (*QModelIndex*) – an index to a parent model

Returns

True if the operation was successful

reset(*map_value*)

Resets the model to given map_value.

rowCount(*parent*=*QModelIndex()*)

Returns the number of rows.

set_box(*top_left*, *bottom_right*, *data*)

Sets data for several indexes at once.

Parameters

- **top_left** (*QModelIndex*) – a sequence of model indexes
- **bottom_right** (*QModelIndex*) – a sequence of values corresponding to the indexes
- **data** (*list of list*) – box of data

setData(*index*, *value*, *role*=*Qt.ItemDataRole.EditRole*)

Sets data in the map.

Parameters

- **index** (*QModelIndex*) – an index to the model
- **value** (*object*) – JSON representation of the value
- **role** (*int*) – a role

Returns

True if the operation was successful

Return type

bool

setHeaderData(*section, orientation, value, role=Qt.ItemDataRole.EditRole*)

trim_columns()

Removes empty columns from the right.

value()

Returns the Map.

index_name(*index*)

`spinetoolbox.mvcmodels.map_model._rows_to_dict(rows)`

Turns table into nested dictionaries.

Parameters

rows (*list*) – a list of row data

Returns

a nested dictionary

Return type

dict

`spinetoolbox.mvcmodels.map_model._reconstruct_map(tree)`

Constructs a Map from a nested dictionary.

Parameters

tree (*dict*) – a nested dictionary

Returns

reconstructed Map

Return type

Map

`spinetoolbox.mvcmodels.map_model._data_length(row)`

Counts the number of non-empty elements at the beginning of row.

Parameters

row (*list*) – a row of data

Returns

data length

Return type

int

`spinetoolbox.mvcmodels.map_model._gather_index_names(map_value)`

Collects index names from Map.

Returns only the ‘first’ index name for nested maps at the same depth.

Parameters

map_value (*Map*) – map to investigate

Returns

index names

Return type

list of str

`spinetoolbox.mvcmodels.map_model._apply_index_names(map_value, index_names)`

Applies index names to Map.

Parameters

- **map_value** (*Map*) – target Map
- **index_names** (*list of str*) – index names

`spinetoolbox.mvcmodels.map_model._numpy_string_to_python_strings(rows)`

Converts instances of `numpy.str_` to regular Python strings.

Parameters

rows (*list of list*) – table rows

Returns

converted rows

Return type

list of list

`spinetoolbox.mvcmodels.minimal_table_model`

Contains a minimal table model.

Module Contents

Classes

MinimalTableModel

Table model for outlining simple tabular data.

class `spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel`(*parent=None, header=None, lazy=True*)

Bases: `PySide6.QtCore.QAbstractTableModel`

Table model for outlining simple tabular data.

Parameters

- **parent** (*QObject, optional*) – the parent object
- **header** (*list of str*) – header labels
- **lazy** (*boolean*) – if True, fetches data lazily

clear()

Clear all data in model.

flags(index)

Return index flags.

canFetchMore(parent)

Return True if the model hasn't been fetched.

fetchMore(parent)

Fetch data and use it to reset the model.

rowCount(*parent=QModelIndex()*)

Number of rows in the model.

columnCount(*parent=QModelIndex()*)

Number of columns in the model.

headerData(*section, orientation=Qt.Orientation.Horizontal, role=Qt.ItemDataRole.DisplayRole*)

Returns headers.

set_horizontal_header_labels(*labels*)

Set horizontal header labels.

insert_horizontal_header_labels(*section, labels*)

Insert horizontal header labels at the given section.

horizontal_header_labels()

setHeaderData(*section, orientation, value, role=Qt.ItemDataRole.EditRole*)

Sets the data for the given role and section in the header with the specified orientation to the value supplied.

data(*index, role=Qt.ItemDataRole.DisplayRole*)

Returns the data stored under the given role for the item referred to by the index.

Parameters

- **index** (*QModelIndex*) – Index of item
- **role** (*int*) – Data role

Returns

Item data for given role.

row_data(*row, role=Qt.ItemDataRole.DisplayRole*)

Returns the data stored under the given role for the given row.

Parameters

- **row** (*int*) – Item row
- **role** (*int*) – Data role

Returns

Row data for given role.

setData(*index, value, role=Qt.ItemDataRole.EditRole*)

Set data in model.

batch_set_data(*indexes, data*)

Batch set data for indexes.

Parameters

- **indexes** (*Iterable of QModelIndex*) – model indexes
- **data** (*Iterable*) – data at each index

Returns

True if data was set successfully, False otherwise

Return type

boolean

insertRows(*row*, *count*, *parent*=*QModelIndex()*)

Inserts count rows into the model before the given row. Items in the new row will be children of the item represented by the parent model index.

Parameters

- **row** (*int*) – Row number where new rows are inserted
- **count** (*int*) – Number of inserted rows
- **parent** (*QModelIndex*) – Parent index

Returns

True if rows were inserted successfully, False otherwise

insertColumns(*column*, *count*, *parent*=*QModelIndex()*)

Inserts count columns into the model before the given column. Items in the new column will be children of the item represented by the parent model index.

Parameters

- **column** (*int*) – Column number where new columns are inserted
- **count** (*int*) – Number of inserted columns
- **parent** (*QModelIndex*) – Parent index

Returns

True if columns were inserted successfully, False otherwise

removeRows(*row*, *count*, *parent*=*QModelIndex()*)

Removes count rows starting with the given row under parent.

Parameters

- **row** (*int*) – Row number where to start removing rows
- **count** (*int*) – Number of removed rows
- **parent** (*QModelIndex*) – Parent index

Returns

True if rows were removed successfully, False otherwise

removeColumns(*column*, *count*, *parent*=*QModelIndex()*)

Removes count columns starting with the given column under parent.

Parameters

- **column** (*int*) – Column number where to start removing columns
- **count** (*int*) – Number of removed columns
- **parent** (*QModelIndex*) – Parent index

Returns

True if columns were removed successfully, False otherwise

reset_model(*main_data*=*None*)

Reset model.

`spinetoolbox.mvcmodels.minimal_tree_model`

Models to represent items in a tree.

Module Contents

Classes

<i>TreeItem</i>	A tree item that can fetch its children.
<i>MinimalTreeModel</i>	Base class for all tree models.

class `spinetoolbox.mvcmodels.minimal_tree_model.TreeItem(model=None)`

A tree item that can fetch its children.

Parameters

model (`MinimalTreeModel`, *optional*) – The model where the item belongs.

property model

property child_item_class

Returns the type of child items. Reimplement in subclasses to return something more meaningful.

property children

property parent_item

property display_data

property edit_data

has_children()

Returns whether this item has or could have children.

child(row)

Returns the child at given row or None if out of bounds.

last_child()

Returns the last child.

child_count()

Returns the number of children.

child_number()

Returns the rank of this item within its parent or -1 if it's an orphan.

find_children(cond=lambda child: ...)

Returns children that meet condition expressed as a lambda function.

find_child(cond=lambda child: ...)

Returns first child that meet condition expressed as a lambda function or None.

next_sibling()

Returns the next sibling or None if it's the last.

previous_sibling()

Returns the previous sibling or None if it's the first.

index()**set_up()****_do_set_up()**

Do stuff after the item has been inserted.

insert_children(*position*, *children*)

Insert new children at given position. Returns a boolean depending on how it went.

Parameters

- **position** (*int*) – insert new items here
- **children** (*list of TreeItem*) – insert items from this iterable

Returns

True if the children were inserted successfully, False otherwise

Return type

bool

append_children(*children*)

Append children at the end.

tear_down()

Do stuff after the item has been removed.

tear_down_recursively()**remove_children(*position*, *count*)**

Removes count children starting from the given position.

Parameters

- **position** (*int*) – position of the first child to remove
- **count** (*int*) – number of children to remove

Returns

True if operation was successful, False otherwise

Return type

bool

clear_children()

Clear children list.

flags(*column*)

Enables the item and makes it selectable.

data(*column*, *role*=*Qt.ItemDataRole.DisplayRole*)

Returns data for given column and role.

can_fetch_more()

Returns whether this item can fetch more.

fetch_more()

Fetches more children.

abstract `set_data(column, value, role)`

Sets data for this item.

Parameters

- **column** (*int*) – column index
- **value** (*object*) – a new value
- **role** (*int*) – role of the new value

Returns

True if data was set successfully, False otherwise

Return type

bool

class `spinetoolbox.mvcmmodels.minimal_tree_model.MinimalTreeModel(parent)`

Bases: `PySide6.QtCore.QAbstractItemModel`

Base class for all tree models.

Init class.

Parameters

parent (`SpineDBEditor`) –

visit_all (*index=QModelIndex(), view=None*)

Iterates all items in the model including and below the given index. Iterative implementation so we don't need to worry about Python recursion limits.

Parameters

- **index** (*QModelIndex*) – an index to start. If not given, we start at the root
- **view** (*QTreeView*) – a tree view. If given, we only yield items that are visible to that view. So for example, if a tree item is not expanded then we don't yield its children.

Yields

TreeItem

item_from_index (*index*)

Return the item corresponding to the given index.

index_from_item (*item*)

Return a model index corresponding to the given item.

Parameters

item (`StandardTreeItem`) – item

Returns

item's index

Return type

`QModelIndex`

index (*row, column, parent=QModelIndex()*)

Returns the index of the item in the model specified by the given row, column and parent index.

parent (*index*)

Returns the parent of the model item with the given index.

columnCount (*parent=QModelIndex()*)

rowCount(*parent=QModelIndex()*)

data(*index, role=Qt.ItemDataRole.DisplayRole*)

Returns the data stored under the given role for the index.

setData(*index, value, role=Qt.ItemDataRole.EditRole*)

Sets data for given index and role. Returns True if successful; otherwise returns False.

flags(*index*)

Returns the item flags for the given index.

hasChildren(*parent*)

canFetchMore(*parent*)

fetchMore(*parent*)

spinetoolbox.mvcmodels.project_item_model

Contains a class for storing project items.

Module Contents

Classes

ProjectItemModel

Class to store project tree items and ultimately project items in a tree structure.

class spinetoolbox.mvcmodels.project_item_model.**ProjectItemModel**(*root, parent=None*)

Bases: PySide6.QtCore.QAbstractItemModel

Class to store project tree items and ultimately project items in a tree structure.

Parameters

- **root** ([RootProjectTreeItem](#)) – Root item for the project item tree
- **parent** (*QObject*) – parent object

root()

Returns the root item.

connect_to_project(*project*)

Connects the model to a project.

Parameters

- **project** ([SpineToolboxProject](#)) – project to connect to

_add_leaf_item(*name*)

Adds a leaf item to the model

Parameters

- **name** (*str*) – project item's name

_remove_leaf_item(*name*)

Removes a leaf item from the model.

Parameters

name (*str*) – project item’s name

_rename_item(*old_name*, *new_name*)

Renames a leaf item.

Parameters

- **old_name** (*str*) – item’s old name
- **new_name** (*str*) – item’s new name

rowCount(*parent=QModelIndex()*)

Reimplemented rowCount method.

Parameters

parent (*QModelIndex*) – Index of parent item whose children are counted.

Returns

Number of children of given parent

Return type

int

columnCount(*parent=QModelIndex()*)

Returns model column count which is always 1.

flags(*index*)

Returns flags for the item at given index

Parameters

index (*QModelIndex*) – Flags of item at this index.

parent(*index=QModelIndex()*)

Returns index of the parent of given index.

Parameters

index (*QModelIndex*) – Index of item whose parent is returned

Returns

Index of parent item

Return type

QModelIndex

index(*row*, *column*, *parent=QModelIndex()*)

Returns index of item with given row, column, and parent.

Parameters

- **row** (*int*) – Item row
- **column** (*int*) – Item column
- **parent** (*QModelIndex*) – Parent item index

Returns

Item index

Return type

QModelIndex

data(*index*, *role=None*)

Returns data in the given index according to requested role.

Parameters

- **index** (*QModelIndex*) – Index to query
- **role** (*int*) – Role to return

Returns

Data depending on role.

Return type

object

item(*index*)

Returns item at given index.

Parameters

index (*QModelIndex*) – Index of item

Returns

Item at given index or root project

item if index is not valid

Return type

RootProjectTreeItem, *CategoryProjectTreeItem* or *LeafProjectTreeItem*

find_category(*category_name*)

Returns the index of the given category name.

Parameters

category_name (*str*) – Name of category item to find

Returns

index of a category item or None if it was not found

Return type

QModelIndex

find_item(*name*)

Returns the *QModelIndex* of the leaf item with the given name

Parameters

name (*str*) – The searched project item (long) name

Returns

Index of a project item with the given name or None if not found

Return type

QModelIndex

get_item(*name*)

Returns leaf item with given name or None if it doesn't exist.

Parameters

name (*str*) – Project item name

Returns

LeafProjectTreeItem, *NoneType*

category_of_item(*name*)

Returns the category item of the category that contains project item with given name

Parameters

name (*str*) – Project item name

Returns

category item or None if the category was not found

Return type

CategoryProjectTreeItem

insert_item(*item*, *parent=QModelIndex()*)

Adds a new item to model. Fails if given parent is not a category item nor a leaf item. New item is inserted as the last item of its branch.

Parameters

- **item** (*CategoryProjectTreeItem* or *LeafProjectTreeItem*) – Project item to add to model
- **parent** (*QModelIndex*) – Parent project item

Returns

True if successful, False otherwise

Return type

bool

remove_item(*item*, *parent=QModelIndex()*)

Removes item from project.

Parameters

- **item** (*BaseProjectTreeItem*) – Item to remove
- **parent** (*QModelIndex*) – Parent of item that is to be removed

Returns

True if item removed successfully, False if item removing failed

Return type

bool

items(*category_name=None*)

Returns a list of leaf items in model according to category name. If no category name given, returns all leaf items in a list.

Parameters

category_name (*str*) – Item category. Data Connections, Data Stores, Importers, Exporters, Tools or Views permitted.

Returns

obj:'list' of obj:'LeafProjectTreeItem': Depending on category_name argument, returns all items or only items according to category. An empty list is returned if there are no items in the given category or if an unknown category name was given.

n_items()

Returns the number of all items in the model excluding category items and root.

Returns

Number of items

Return type

int

item_names()

Returns all leaf item names in a list.

Returns

'list' of obj:'str': Item names

Return type

obj

items_per_category()

Returns a dict mapping category indexes to a list of items in that category.

Returns

dict(QModelIndex,list(LeafProjectTreeItem))

leaf_indexes()

Yields leaf indexes.

remove_leaves()**spinetoolbox.mvcmodels.project_item_specification_models**

Contains a class for storing Tool specifications.

Module Contents**Classes***ProjectItemSpecificationModel*

Class to store specs that are available in a project e.g. GAMS or Julia models.

*FilteredSpecificationModel***class** spinetoolbox.mvcmodels.project_item_specification_models.**ProjectItemSpecificationModel**(*icons*)

Bases: PySide6.QtCore.QAbstractListModel

Class to store specs that are available in a project e.g. GAMS or Julia models.

specification_replaced**add_specification**(*name*)

Adds a specification to the model.

Parameters**name** (*str*) – specification's name**remove_specification**(*name*)

Removes a specification from the model

Parameters**name** (*str*) – specification's name

replace_specification(*old_name*, *new_name*)

Replaces a specification.

Parameters

- **old_name** (*str*) – previous name
- **new_name** (*str*) – new name

connect_to_project(*project*)

Connects the model to a project.

Parameters

- project** ([SpineToolboxProject](#)) – project to connect to

clear()

rowCount(*parent=None*)

Returns the number of specs in the model.

Parameters

- parent** (*QModelIndex*) – Not used (because this is a list)

Returns

Number of rows (available specs) in the model

data(*index*, *role=None*)

Must be reimplemented when subclassing.

Parameters

- **index** (*QModelIndex*) – Requested index
- **role** (*int*) – Data role

Returns

Data according to requested role

flags(*index*)

Returns enabled flags for the given index.

Parameters

- index** (*QModelIndex*) – Index of spec

insertRow(*spec_name*, *row=None*, *parent=QModelIndex()*)

Insert row (specification) into model.

Parameters

- **spec_name** (*str*) – name of spec added to the model
- **row** (*int*, *optional*) – Row to insert spec to
- **parent** (*QModelIndex*) – Parent of child (not used)

Returns

Void

removeRow(*row*, *parent=QModelIndex()*)

Remove row (spec) from model.

Parameters

- **row** (*int*) – Row to remove the spec from

- **parent** (*QModelIndex*) – Parent of spec on row (not used)

Returns

Boolean variable

specification(*row*)

Returns spec on given row.

Parameters

row (*int*) – Row of spec specification

Returns

ProjectItemSpecification from specification list or None if given row is zero

specification_row(*name*)

Returns the row on which the given specification is located or -1 if it is not found.

specification_index(*name*)

Returns the QModelIndex on which a specification with the given name is located or invalid index if it is not found.

class spinetoolbox.mvcmodels.project_item_specification_models.**FilteredSpecificationModel**(*item_type*)

Bases: PySide6.QtCore.QSortFilterProxyModel

filterAcceptsRow(*source_row*, *source_parent*)

get_mime_data_text(*index*)

specifications()

Yields all specs.

specification(*row*)

spinetoolbox.mvcmodels.project_tree_item

Project Tree items.

Module Contents**Classes**

<i>BaseProjectTreeItem</i>	Base class for all project tree items.
<i>RootProjectTreeItem</i>	Class for the root project tree item.
<i>CategoryProjectTreeItem</i>	Class for category project tree items.
<i>LeafProjectTreeItem</i>	Class for leaf items in the project item tree.

class spinetoolbox.mvcmodels.project_tree_item.**BaseProjectTreeItem**(*name*, *description*)

Bases: *spinetoolbox.metaobject.MetaObject*

Base class for all project tree items.

Parameters

- **name** (*str*) – Object name
- **description** (*str*) – Object description

flags()

Returns the item flags.

parent()

Returns parent project tree item.

child_count()

Returns the number of child project tree items.

children()

Returns the children of this project tree item.

child(row)

Returns child BaseProjectTreeItem on given row.

Parameters

row (*int*) – Row of child to return

Returns

item on given row or None if it does not exist

Return type

BaseProjectTreeItem

row()

Returns the row on which this item is located.

abstract add_child(child_item)

Base method that shall be overridden in subclasses.

remove_child(row)

Remove the child of this BaseProjectTreeItem from given row. Do not call this method directly. This method is called by ProjectItemTreeModel when items are removed.

Parameters

row (*int*) – Row of child to remove

Returns

True if operation succeeded, False otherwise

Return type

bool

abstract custom_context_menu(toolbox)

Returns the context menu for this item. Implement in subclasses as needed.

Parameters

toolbox (*QWidget*) – The widget that is controlling the menu

Returns

context menu

Return type

QMenu

class spinetoolbox.mvcmodels.project_tree_item.RootProjectTreeItem

Bases: *BaseProjectTreeItem*

Class for the root project tree item.

Parameters

- **name** (*str*) – Object name
- **description** (*str*) – Object description

add_child(*child_item*)

Adds given category item as the child of this root project tree item. New item is added as the last item.

Parameters

child_item ([CategoryProjectTreeItem](#)) – Item to add

Returns

True for success, False otherwise

abstract custom_context_menu(*toolbox*)

See base class.

class `spinetoolbox.mvcmodels.project_tree_item.CategoryProjectTreeItem`(*name*, *description*)

Bases: [BaseProjectTreeItem](#)

Class for category project tree items.

Parameters

- **name** (*str*) – Object name
- **description** (*str*) – Object description

flags()

Returns the item flags.

add_child(*child_item*)

Adds given project tree item as the child of this category item. New item is added as the last item.

Parameters

child_item ([LeafProjectTreeItem](#)) – Item to add

Returns

True for success, False otherwise

custom_context_menu(*toolbox*)

Returns the context menu for this item.

Parameters

toolbox ([ToolboxUI](#)) – Toolbox main window

Returns

context menu

Return type

QMenu

class `spinetoolbox.mvcmodels.project_tree_item.LeafProjectTreeItem`(*project_item*)

Bases: [BaseProjectTreeItem](#)

Class for leaf items in the project item tree.

Parameters

project_item ([ProjectItem](#)) – the real project item this item represents

property project_item

the project item linked to this leaf

abstract `add_child(child_item)`

See base class.

flags()

Returns the item flags.

custom_context_menu(*toolbox*)

Returns the context menu for this item.

Parameters

toolbox ([ToolboxUI](#)) – Toolbox main window

Returns

context menu

Return type

QMenu

`spinetoolbox.mvcmodels.resource_filter_model`

Contains ResourceFilterModel.

Module Contents

Classes

ResourceFilterModel

param connection

connection whose resources to model

class `spinetoolbox.mvcmodels.resource_filter_model.ResourceFilterModel(connection, project, undo_stack, logger)`

Bases: `PySide6.QtGui.QStandardItemModel`

Parameters

- **connection** ([LoggingConnection](#)) – connection whose resources to model
- **project** ([SpineToolboxProject](#)) – project
- **undo_stack** ([QUndoStack](#)) – an undo stack
- **logger** ([LoggerInterface](#)) – a logger

property connection

tree_built

_SELECT_ALL = 'Select all'

_FILTER_TYPES

_FILTER_TYPE_TO_TEXT

build_tree()

Rebuilds model's contents.

fetch_filters()

setData(*index*, *value*, *role=Qt.ItemDataRole.EditRole*)

_change_filter_checked_state(*index*, *is_on*)

Changes the online status of the filter item at index.

Parameters

- **index** (*QModelIndex*) – item's index
- **is_on** (*bool*) – True if filter are turned online, False otherwise

set_online(*resource*, *filter_type*, *online*)

Sets the given filters online or offline.

Parameters

- **resource** (*str*) – Resource label
- **filter_type** (*str*) – Either SCENARIO_FILTER_TYPE or TOOL_FILTER_TYPE, for now.
- **online** (*dict*) – mapping from scenario/tool id to online flag

_find_filter_type_item(*resource*, *filter_type*)

Searches for filter type item.

Parameters

- **resource** (*str*) – resource label
- **filter_type** (*str*) – filter type identifier

Returns

filter type item or None if not found

Return type

QStandardItem

_set_all_selected_item(*resource*, *filter_type_item*, *emit_data_changed=False*)

Updates 'Select All' item's checked state.

Parameters

- **resource** (*str*) – resource label
- **filter_type_item** (*QStandardItem*) – filter type item
- **emit_data_changed** (*bool*) – if True, emit dataChanged signal if the state was updated

`spinetoolbox.mvcmodels.shared`

Contains stuff that is used by more than one model

Module Contents

`spinetoolbox.mvcmodels.shared.PARSED_ROLE`

`spinetoolbox.mvcmodels.shared.DB_MAP_ROLE`

`spinetoolbox.mvcmodels.time_pattern_model`

A model for time patterns, used by the `parameter_value` editors.

Module Contents

Classes

<i><code>TimePatternModel</code></i>	A model for time pattern type parameter values.
--------------------------------------	---

class `spinetoolbox.mvcmodels.time_pattern_model.TimePatternModel`(*value*, *parent*)

Bases: `spinetoolbox.mvcmodels.indexed_value_table_model.IndexedValueTableModel`

A model for time pattern type parameter values.

Parameters

- **value** (*IndexedValue*) – a `parameter_value`
- **parent** (*QObject*) – parent object

flags(*index*)

Returns flags at index.

insertRows(*row*, *count*, *parent=QModelIndex()*)

Inserts new time period - value pairs into the pattern.

New time periods are initialized to empty strings and the corresponding values to zeros.

Parameters

- **row** (*int*) – an index where to insert the new data
- **count** (*int*) – number of time period - value pairs to insert
- **parent** (*QModelIndex*) – an index to a parent model

Returns

True if the operation was successful

Return type

bool

removeRows(*row*, *count*, *parent*=*QModelIndex()*)

Removes time period - value pairs from the pattern.

Parameters

- **row** (*int*) – an index where to remove the data
- **count** (*int*) – number of time period - value pairs to remove
- **parent** (*QModelIndex*) – an index to a parent model

Returns

True if the operation was successful

Return type

bool

setData(*index*, *value*, *role*=*Qt.ItemDataRole.EditRole*)

Sets a time period or a value in the pattern.

Column index 0 corresponds to the time periods while 1 corresponds to the values.

Parameters

- **index** (*QModelIndex*) – an index to the model
- **value** (*str*, *float*) – a new time period or value
- **role** (*int*) – a role

Returns

True if the operation was successful

Return type

bool

batch_set_data(*indexes*, *values*)

Sets data for several indexes at once.

Parameters

- **indexes** (*Sequence*) – a sequence of model indexes
- **values** (*Sequence*) – a sequence of time periods/floats corresponding to the indexes

spinetoolbox.mvcmodels.time_series_model_fixed_resolution

A model for fixed resolution time series, used by the parameter_value editors.

Module Contents

Classes

TimeSeriesModelFixedResolution

A model for fixed resolution time series type parameter values.

class `spinetoolbox.mvcmodels.time_series_model_fixed_resolution.TimeSeriesModelFixedResolution`(*series*,
parent)

Bases: `spinetoolbox.mvcmodels.indexed_value_table_model.IndexedValueTableModel`

A model for fixed resolution time series type parameter values.

Parameters

- **series** (*TimeSeriesFixedResolution*) – a time series
- **parent** (*QObject*) – parent object

property indexes

Returns the time stamps as an array.

property values

Returns the values of the time series as an array.

flags(*index*)

Returns flags at index.

insertRows(*row*, *count*, *parent=QModelIndex()*)

Inserts new values to the series.

The new values are set to zero. Start time or resolution are left unchanged.

Parameters

- **row** (*int*) – a numeric index to the first stamp/value to insert
- **count** (*int*) – number of stamps/values to insert
- **parent** (*QModelIndex*) – index to a parent model

Returns

True if the operation was successful

removeRows(*row*, *count*, *parent=QModelIndex()*)

Removes values from the series.

Parameters

- **row** (*int*) – a numeric index to the series where to begin removing
- **count** (*int*) – how many stamps/values to remove
- **parent** (*QModelIndex*) – an index to the parent model

Returns

True if the operation was successful.

reset(*value*)

Resets the model with new time series data.

setData(*index*, *value*, *role=Qt.ItemDataRole.EditRole*)

Sets a given value in the series.

Column index 1 refers to values. Note it does not make sense to set the time stamps in fixed resolution series.

Parameters

- **index** (*QModelIndex*) – an index to the model

- **value** (*numpy.datetime64, float*) – a new stamp or value
- **role** (*int*) – a role

Returns

True if the operation was successful

batch_set_data(*indexes, values*)

Sets data for several indexes at once.

Only the values of the series are modified as the time stamps are immutable.

Parameters

- **indexes** (*Sequence*) – a sequence of model indexes
- **values** (*Sequence*) – a sequence of floats corresponding to the indexes

set_ignore_year(*ignore_year*)

Sets the ignore_year option of the time series.

set_repeat(*repeat*)

Sets the repeat option of the time series.

set_resolution(*resolution*)

Sets the resolution.

set_start(*start*)

Sets the start datetime.

`spinetoolbox.mvcmodels.time_series_model_variable_resolution`

A model for variable resolution time series, used by the parameter_value editors.

Module Contents

Classes

TimeSeriesModelVariableResolution

A model for variable resolution time series type parameter values.

class `spinetoolbox.mvcmodels.time_series_model_variable_resolution.TimeSeriesModelVariableResolution`(*value*, *parent*)

Bases: `spinetoolbox.mvcmodels.indexed_value_table_model.IndexedValueTableModel`

A model for variable resolution time series type parameter values.

Parameters

- **value** (*IndexedValue*) – a parameter_value
- **parent** (*QObject*) – parent object

property indexes

Returns the time stamps as an array.

property values

Returns the values of the time series as an array.

flags(*index*)

Returns the flags for given model index.

insertRows(*row*, *count*, *parent*=*QModelIndex()*)

Inserts new time stamps and values to the series.

When inserting in the middle of the series the new time stamps are distributed evenly among the time span between the two time stamps around the insertion point. When inserting at the beginning or at the end of the series the duration between the new time stamps is set equal to the first/last duration in the original series.

The new values are set to zero.

Parameters

- **row** (*int*) – a numeric index to the first stamp/value to insert
- **count** (*int*) – number of stamps/values to insert
- **parent** (*QModelIndex*) – index to a parent model

Returns

True if the insertion was successful

Return type

bool

removeRows(*row*, *count*, *parent*=*QModelIndex()*)

Removes time stamps/values from the series.

Parameters

- **row** (*int*) – a numeric index to the series where to begin removing
- **count** (*int*) – how many stamps/values to remove
- **parent** (*QModelIndex*) – an index to the parent model

Returns

True if the operation was successful.

Return type

bool

reset(*value*)

Resets the model with new time series data.

setData(*index*, *value*, *role*=*Qt.ItemDataRole.EditRole*)

Sets a given time stamp or value in the series.

Column index 0 refers to time stamps while index 1 to values.

Parameters

- **index** (*QModelIndex*) – an index to the model
- **value** (*numpy.datetime64*, *float*) – a new stamp or value
- **role** (*int*) – a role

Returns

True if the operation was successful

Return type

bool

batch_set_data(*indexes*, *values*)

Sets data for several indexes at once.

Parameters

- **indexes** (*Sequence*) – a sequence of model indexes
- **values** (*Sequence*) – a sequence of datetimes/floats corresponding to the indexes

set_ignore_year(*ignore_year*)

Sets the ignore_year option of the time series.

set_repeat(*repeat*)

Sets the repeat option of the time series.

spinetoolbox.project_item

This subpackage contains base classes for project items.

Submodules**spinetoolbox.project_item.logging_connection**

Contains logging connection and jump classes.

Module Contents**Classes**

<i>HeadlessConnection</i>	A project item connection that is compatible with headless mode.
<i>LoggingConnection</i>	A project item connection that is compatible with headless mode.
<i>LoggingJump</i>	Represents a conditional jump between two project items.

```
class spinetoolbox.project_item.logging_connection.HeadlessConnection(source_name,
                                                                    source_position,
                                                                    destination_name,
                                                                    destination_position,
                                                                    options=None,
                                                                    filter_settings=None,
                                                                    legacy_resource_filter_ids=None)
```

Bases: spine_engine.project_item.connection.ResourceConvertingConnection

A project item connection that is compatible with headless mode.

Parameters

- **source_name** (*str*) – source project item's name

- **source_position** (*str*) – source anchor’s position
- **destination_name** (*str*) – destination project item’s name
- **destination_position** (*str*) – destination anchor’s position
- **options** (*dict*, *optional*) – any additional options
- **filter_settings** (*FilterSettings*, *optional*) – filter settings

property database_resources

Connection’s database resources

set_filter_enabled(*resource_label*, *filter_type*, *filter_name*, *enabled*)

Enables or disables a filter.

Parameters

- **resource_label** (*str*) – database resource name
- **filter_type** (*str*) – filter type
- **filter_name** (*str*) – filter name
- **enabled** (*bool*) – True to enable the filter, False to disable it

_convert_legacy_resource_filter_ids_to_filter_settings()

Converts legacy resource filter ids to filter settings.

This method should be called once after constructing the connection from potentially legacy dict using `from_dict()`.

static _constructor_args_from_dict(*connection_dict*)

See base class.

classmethod from_dict(*connection_dict*, ***kwargs*)

Deserializes a connection from dict.

Parameters

- **connection_dict** (*dict*) – serialized `LoggingConnection`
- ****kwargs** – additional keyword arguments to be forwarded to class constructor

receive_resources_from_source(*resources*)

See base class.

replace_resources_from_source(*old*, *new*)

Replaces existing resources by new ones.

Parameters

- **old** (*list of ProjectItemResource*) – old resources
- **new** (*list of ProjectItemResource*) – new resources

class `spinetoolbox.project_item.logging_connection.LoggingConnection`(*args, *toolbox*, ***kwargs*)

Bases: `spinetoolbox.log_mixin.LogMixin`, `HeadlessConnection`

A project item connection that is compatible with headless mode.

Parameters

- **source_name** (*str*) – source project item’s name
- **source_position** (*str*) – source anchor’s position

- **destination_name** (*str*) – destination project item’s name
- **destination_position** (*str*) – destination anchor’s position
- **options** (*dict*, *optional*) – any additional options
- **filter_settings** (*FilterSettings*, *optional*) – filter settings

property graphics_item

__hash__()

Return hash(self).

static item_type()

has_filters()

Returns True if connection has scenario or tool filters.

Returns

True if connection has filters, False otherwise

Return type

bool

_get_db_map(*url*, *ignore_version_error=False*)

_pop_unused_db_maps()

Removes unused database maps and unregisters from listening the DB manager.

_make_fetch_parent(*db_map*, *item_type*)

_fetch_more_if_possible()

_receive_data_changed()

receive_session_committed(*db_maps*, *cookie*)

receive_session_rolled_back(*db_map*)

receive_error_msg(*_db_map_error_log*)

get_scenario_names(*url*)

get_tool_names(*url*)

may_have_filters()

Returns whether this connection may have filters.

Returns

True if it is possible for the connection to have filters, False otherwise

Return type

bool

may_have_write_index()

Returns whether this connection may have write index.

Returns

True if it is possible for the connection to have write index, False otherwise

Return type

bool

may_use_memory_db()

Returns whether this connection may use memory DB.

Returns

True if it is possible for the connection to use memory DB, False otherwise

Return type

bool

may_use_datapackage()

Returns whether this connection may use datapackage.

Returns

True if it is possible for the connection to use datapackage, False otherwise

Return type

bool

may_purge_before_writing()

Returns whether this connection may purge before writing.

Returns

True if it is possible for the connection to purge before writing, False otherwise

Return type

bool

online_filters(resource_label, filter_type)

Returns filter online states for given resource and filter type.

Parameters

- **resource_label** (*str*) – resource label
- **filter_type** (*str*) – filter type

Returns

mapping from filter names to online states

Return type

dict

set_online(resource, filter_type, online)

Sets the given filters online or offline.

Parameters

- **resource** (*str*) – Resource label
- **filter_type** (*str*) – Either SCENARIO_FILTER_TYPE or TOOL_FILTER_TYPE, for now.
- **online** (*dict*) – mapping from scenario/tool name to online flag

set_filter_default_online_status(auto_online)

Sets the auto_online flag.

Parameters

auto_online (*bool*) – If True, unknown filters are online by default

refresh_resource_filter_model()

Makes resource filter mode fetch filter data from database.

receive_resources_from_source(*resources*)

See base class.

replace_resources_from_source(*old*, *new*)

See base class.

set_connection_options(*options*)

Overwrites connections options.

Parameters

options (*dict*) – new options

_check_available_filters()

Cross-checks filter settings with source databases.

Returns

filter settings containing only filters that exist in source databases

Return type

FilterSettings

_resource_filters_online(*resource*, *filter_type*)

to_dict()

See base class.

tear_down()

Releases system resources held by the connection.

class spinetoolbox.project_item.logging_connection.**LoggingJump**(*args, toolbox=None, **kwargs)

Bases: [spinetoolbox.log_mixin.LogMixin](#), spine_engine.project_item.connection.Jump

Represents a conditional jump between two project items.

Parameters

- **source_name** (*str*) – source project item's name
- **source_position** (*str*) – source anchor's position
- **destination_name** (*str*) – destination project item's name
- **destination_position** (*str*) – destination anchor's position
- **condition** (*dict*) – jump condition

property graphics_item

static item_type()

spinetoolbox.project_item.project_item

Contains base classes for project items and item factories.

Module Contents

Classes

<i>ProjectItem</i>	Class for project items that are not category nor root.
--------------------	---

class `spinetoolbox.project_item.project_item.ProjectItem(name, description, x, y, project)`

Bases: `spinetoolbox.log_mixin.LogMixin`, `spinetoolbox.metaobject.MetaObject`

Class for project items that are not category nor root. These items can be executed, refreshed, and so on.

x

horizontal position in the screen

Type

float

y

vertical position in the screen

Type

float

Parameters

- **name** (*str*) – item name
- **description** (*str*) – item description
- **x** (*float*) – horizontal position on the scene
- **y** (*float*) – vertical position on the scene
- **project** (`SpineToolboxProject`) – project item's project

property logger

abstract property executable_class

create_data_dir()

data_files()

Returns a list of files that are in the data directory.

abstract static item_type()

Item's type identifier string.

Returns

type string

Return type

str

abstract static item_category()

Item's category.

Returns

category name

Return type

str

make_signal_handler_dict()

Returns a dictionary of all shared signals and their handlers. This is to enable simpler connecting and disconnecting. Must be implemented in subclasses.

activate()

Restore selections and connect signals.

deactivate()

Save selections and disconnect signals.

restore_selections()

Restore selections into shared widgets when this project item is selected.

save_selections()

Save selections in shared widgets for this project item into instance variables.

_connect_signals()

Connect signals to handlers.

_disconnect_signals()

Disconnect signals from handlers and check for errors.

set_properties_ui(*properties_ui*)

Sets the properties tab widget for the item.

Note that this method expects the widget that is generated from the .ui files and initialized with the `setUpUi()` method rather than the entire properties tab widget.

Parameters

properties_ui (*QWidget*) – item’s properties UI

specification()

Returns the specification for this item.

undo_specification()**set_specification(*specification*)**

Pushes a new `SetItemSpecificationCommand` to the toolbox’ undo stack.

do_set_specification(*specification*)

Sets specification for this item. Removes specification if `None` given as argument.

Parameters

specification (*ProjectItemSpecification*) – specification of this item. `None` removes the specification.

set_icon(*icon*)

Sets the icon for the item.

Parameters

icon (*ProjectItemIcon*) – item’s icon

get_icon()

Returns the graphics item representing this item in the scene.

_check_notifications()

Checks if exclamation icon notifications need to be set or cleared.

clear_notifications()

Clear all notifications from the exclamation icon.

add_notification(*text*)

Add a notification to the exclamation icon.

remove_notification(*text*)

set_rank(*rank*)

Set rank of this item for displaying in the design view.

handle_execution_successful(*execution_direction*, *engine_state*)

Performs item dependent actions after the execution item has finished successfully.

Parameters

- **execution_direction** (*str*) – “FORWARD” or “BACKWARD”
- **engine_state** – engine state after item’s execution

resources_for_direct_successors()

Returns resources for direct successors.

These resources can include transient files that don’t exist yet, or filename patterns. The default implementation returns an empty list.

Returns

a list of `ProjectItemResources`

Return type

list

resources_for_direct_predecessors()

Returns resources for direct predecessors.

These resources can include transient files that don’t exist yet, or filename patterns. The default implementation returns an empty list.

Returns

a list of `ProjectItemResources`

Return type

list

_resources_to_predecessors_changed()

Notifies direct predecessors that item’s resources have changed.

_resources_to_predecessors_replaced(*old*, *new*)

Notifies direct predecessors that item’s resources have been replaced.

Parameters

- **old** (*list of ProjectItemResource*) – old resources
- **new** (*list of ProjectItemResource*) – new resources

upstream_resources_updated(*resources*)

Notifies item that resources from direct predecessors have changed.

Parameters

resources (*list of ProjectItemResource*) – new resources from upstream

replace_resources_from_upstream(*old, new*)

Replaces existing resources from direct predecessor by a new ones.

Parameters

- **old** (*list of ProjectItemResource*) – old resources
- **new** (*list of ProjectItemResource*) – new resources

_resources_to_successors_changed()

Notifies direct successors that item's resources have changed.

_resources_to_successors_replaced(*old, new*)

Notifies direct successors that one of item's resources has been replaced.

Parameters

- **old** (*list of ProjectItemResource*) – old resources
- **new** (*list of ProjectItemResource*) – new resources

downstream_resources_updated(*resources*)

Notifies item that resources from direct successors have changed.

Parameters

resources (*list of ProjectItemResource*) – new resources from downstream

replace_resources_from_downstream(*old, new*)

Replaces existing resources from direct successor by a new ones.

Parameters

- **old** (*list of ProjectItemResource*) – old resources
- **new** (*list of ProjectItemResource*) – new resources

invalidate_workflow(*edges*)

Notifies that this item's workflow is not acyclic.

Parameters

edges (*list*) – A list of edges that make the graph acyclic after removing them.

revalidate_workflow()

item_dict()

Returns a dictionary corresponding to this item.

Returns

serialized project item

Return type

dict

static item_dict_local_entries()

Returns entries or 'paths' in item dict that should be stored in project's local data directory.

Returns

local data item dict entries

Return type

list of tuple of str

static `parse_item_dict(item_dict)`

Reads the information needed to construct the base `ProjectItem` class from an item dict.

Parameters

item_dict (*dict*) – an item dict

Returns

item's name, description as well as x and y coordinates

Return type

tuple

copy_local_data(*item_dict*)

Copies local data linked to a duplicated project item.

Parameters

item_dict (*dict*) – serialized item

abstract static `from_dict(name, item_dict, toolbox, project)`

Deserialized an item from item dict.

Parameters

- **name** (*str*) – item's name
- **item_dict** (*dict*) – serialized item
- **toolbox** (`ToolboxUI`) – the main window
- **project** (`SpineToolboxProject`) – a project

Returns

deserialized item

Return type

ProjectItem

actions()

Item specific actions.

Returns

item's actions

Return type

list of `QAction`

rename(*new_name, rename_data_dir_message*)

Renames this item.

If the project item needs any additional steps in renaming, override this method in subclass. See e.g. `rename()` method in `DataStore` class.

Parameters

- **new_name** (*str*) – New name
- **rename_data_dir_message** (*str*) – Message to show when renaming item's data directory

Returns

True if item was renamed successfully, False otherwise

Return type

bool

open_directory(*checked=False*)

Open this item's data directory in file explorer.

tear_down()

Tears down this item. Called both before closing the app and when removing the item from the project. Implement in subclasses to eg close all QMainWindow's opened by this item.

set_up()

Sets up this item. Called when adding the item to the project. Implement in subclasses to eg recreate attributes destroyed by `tear_down`.

update_name_label()

Updates the name label on the properties widget, used when selecting an item and renaming the selected one.

notify_destination(*source_item*)

Informs an item that it has become the destination of a connection between two items.

The default implementation logs a warning message. Subclasses should reimplement this if they need more specific behavior.

Parameters

source_item ([ProjectItem](#)) – connection source item

static upgrade_v1_to_v2(*item_name, item_dict*)

Upgrades item's dictionary from v1 to v2.

Subclasses should reimplement this method if there are changes between version 1 and version 2.

Parameters

- **item_name** (*str*) – item's name
- **item_dict** (*dict*) – Version 1 item dictionary

Returns

Version 2 item dictionary

Return type

dict

static upgrade_v2_to_v3(*item_name, item_dict, project_upgrader*)

Upgrades item's dictionary from v2 to v3.

Subclasses should reimplement this method if there are changes between version 2 and version 3.

Parameters

- **item_name** (*str*) – item's name
- **item_dict** (*dict*) – Version 2 item dictionary
- **project_upgrader** ([ProjectUpgrader](#)) – Project upgrader class instance

Returns

Version 3 item dictionary

Return type

dict

`spinetoolbox.project_item.project_item_factory`

Contains base classes for project items and item factories.

Module Contents

Classes

`ProjectItemFactory`

Class for project item factories.

class `spinetoolbox.project_item.project_item_factory.ProjectItemFactory`

Class for project item factories.

abstract static `item_class()`

Returns the project item's class.

Returns

item's class

Return type

type

static `is_deprecated()`

Queries if item is deprecated.

Returns

True if item is deprecated, False otherwise

Return type

bool

abstract static `icon()`

Returns the icon resource path.

Returns

str

abstract static `icon_color()`

Returns the icon color.

Returns

icon's color

Return type

QColor

abstract static `make_add_item_widget(toolbox, x, y, specification)`

Returns an appropriate Add project item widget.

Parameters

- **toolbox** (`ToolboxUI`) – the main window
- **x** (`int`) – Icon coordinates
- **y** (`int`) – Icon coordinates
- **specification** (`ProjectItemSpecification`) – item's specification

Returns

QWidget

abstract static make_icon(toolbox)

Returns a ProjectItemIcon to use with given toolbox, for given project item.

Parameters**toolbox** (ToolboxUI) –**Returns**

item's icon

Return type*ProjectItemIcon***abstract static make_item(name, item_dict, toolbox, project)**

Returns a project item constructed from the given item_dict.

Parameters

- **name** (*str*) – item's name
- **item_dict** (*dict*) – serialized project item
- **toolbox** (ToolboxUI) – Toolbox main window
- **project** (SpineToolboxProject) – the project the item belongs to

Returns

ProjectItem

abstract static make_properties_widget(toolbox)

Creates the item's properties tab widget.

Returns

item's properties tab widget

Return type

QWidget

abstract static make_specification_menu(parent, index)

Creates item specification's context menu.

Subclasses that do not support specifications can still raise `NotImplementedError`.**Parameters**

- **parent** (QWidget) – menu's parent widget
- **index** (QModelIndex) – an index from specification model

Returns

specification's context menu

Return type*ItemSpecificationMenu***abstract static make_specification_editor(toolbox, specification=None, item=None, **kwargs)**

Creates the item's specification widget.

Subclasses that do not support specifications can still raise `NotImplementedError`.**Parameters**

- **toolbox** (ToolboxUI) – Toolbox main window

- **specification** (*ProjectItemSpecification*, *optional*) – a specification to show in the widget or None for a fresh start
- **item** (*ProjectItem*, *optional*) – a project item. If the specification is accepted, it is also set for this item
- ****kwargs** – parameters passed to the specification widget

Returns

item's specification widget

Return type

QWidget

static repair_specification(*toolbox*, *specification*)

Called right after a spec is added to the project. Finds if there's something wrong with the spec and proposes actions to fix it with help from toolbox.

Parameters

- **toolbox** (*ToolboxUI*) – Toolbox main window
- **specification** (*ProjectItemSpecification*) – a specification to check

spinetoolbox.project_item.specification_editor_window

Contains SpecificationEditorWindowBase and ChangeSpecPropertyCommand

Module Contents**Classes**

<i>ChangeSpecPropertyCommand</i>	Command to set specification properties.
<i>SpecificationEditorWindowBase</i>	Base class for spec editors.
<i>_SpecNameDescriptionToolBar</i>	QToolBar for line edits and a hamburger menu.

Functions

<i>prompt_to_save_changes</i> (parent, settings, save_callback)	Prompts to save changes.
---	--------------------------

```
class spinetoolbox.project_item.specification_editor_window.ChangeSpecPropertyCommand(callback,
                                                                                       new_value,
                                                                                       old_value,
                                                                                       cmd_name)
```

Bases: PySide6.QtGui.QUndoCommand

Command to set specification properties.

Parameters

- **callback** (*function*) – Function to call to set the spec property.
- **new_value** (*any*) – new value

- **old_value** (*any*) – old value
- **cmd_name** (*str*) – command name

redo()

undo()

```
class spinetoolbox.project_item.specification_editor_window.SpecificationEditorWindowBase(toolbox,
                                                                                          spec-
                                                                                          i-
                                                                                          fi-
                                                                                          ca-
                                                                                          tion=None,
                                                                                          item=None)
```

Bases: PySide6.QtWidgets.QMainWindow

Base class for spec editors.

Parameters

- **toolbox** (*ToolboxUI*) – QMainWindow instance
- **specification** (*ProjectItemSpecification*, *optional*) – If given, the form is pre-filled with this specification
- **item** (*ProjectItem*, *optional*) – Sets the spec for this item if accepted

abstract property settings_group

Returns the settings group for this spec type.

Returns

str

property _duplicate_kwargs

abstract _make_ui()

Returns the ui object from Qt designer.

Returns

object

_restore_dock_widgets()

Restores dockWidgets to some default state. Called in the constructor, before restoring the ui from settings. Reimplement in subclasses if needed.

abstract _make_new_specification(spec_name, exiting=None)

Returns a ProjectItemSpecification from current form settings.

Parameters

- **spec_name** (*str*) – Name of the spec
- **exiting** (*bool*, *optional*) – Set as True if called when trying to exit the editor window

Returns

ProjectItemSpecification

spec_toolbar()

Returns spec editor window's toolbar, which contains e.g. the hamburger menu.

show_error(message)

_show_status_bar_msg(*msg*)

_populate_main_menu()

_update_window_modified(*clean*)

_set_window_title(*title*)

Sets window title.

Parameters

title (*str*) – new window title

_save(*exiting=None*)

Saves spec.

Parameters

exiting (*bool*, *optional*) – Set as True if called when trying to exit the editor window

Returns

True if operation was successful, False otherwise

Return type

bool

prompt_exit_without_saving()

Prompts whether the user wants to exit without saving or cancel the exit.

Returns

False if the user chooses to cancel, in which case we don't close the form.

Return type

bool

_duplicate()

tear_down()

closeEvent(*event*)

class spinetoolbox.project_item.specification_editor_window._SpecNameDescriptionToolBar(*parent*,
spec,
undo_stack)

Bases: PySide6.QtWidgets.QToolBar

QToolBar for line edits and a hamburger menu.

Parameters

- **parent** (*QMainWindow*) – QMainWindow instance
- **spec** (*ProjectItemSpecification*) – specification that is being edited
- **undo_stack** (*QUndoStack*) – an undo stack

name_changed

_make_main_menu()

_set_name()

_set_description()

do_set_name(*name*)

do_set_description(*description*)

name()

description()

spinetoolbox.project_item.specification_editor_window.prompt_to_save_changes(*parent*, *settings*,
save_callback,
exiting=None)

Prompts to save changes.

Parameters

- **parent** (*QWidget*) – Spec editor widget
- **settings** (*QSettings*) – Toolbox settings
- **save_callback** (*Callable*) – A function to call if the user chooses Save. It must return True or False depending on the outcome of the ‘saving’.
- **exiting** (*bool*, *optional*) – Set as True if called when trying to exit the editor window

Returns

False if the user chooses to cancel, in which case we don’t close the form.

Return type

bool

spinetoolbox.server

Package for handling the client part of executing projects on Spine Engine Server.

Submodules

spinetoolbox.server.engine_client

Client for exchanging messages between the toolbox and the Spine Engine Server.

Module Contents

Classes

ClientSecurityModel

Generic enumeration.

EngineClient

param host

IP address of the Spine Engine Server

```
class spinetoolbox.server.engine_client.ClientSecurityModel
```

Bases: `enum.Enum`

Generic enumeration.

Derive from this class to define new enumerations.

NONE = 0

STONEHOUSE = 1

```
class spinetoolbox.server.engine_client.EngineClient(host, port, sec_model, sec_folder, ping=True)
```

Parameters

- **host** (*str*) – IP address of the Spine Engine Server
- **port** (*int*) – Port of the client facing (frontend) socket on Spine Engine Server
- **sec_model** (`ClientSecurityModel`) – Client security scheme
- **sec_folder** (*str*) – Path to security file directory
- **ping** (*bool*) – Whether to check connectivity at instance creation

```
connect_pull_socket(port)
```

Connects a PULL socket for receiving engine execution events and files from server.

Parameters

port (*str*) – Port of the PUSH socket on server

```
rcv_next(dealer_or_pull)
```

Polls all sockets and returns a new reply based on given socket ‘name’.

Parameters

dealer_or_pull (*str*) – “dealer” to wait reply from DEALER socket, “pull” to wait reply from PULL socket

```
_check_connectivity(timeout)
```

Pings server, waits for the response, and acts accordingly.

Parameters

timeout (*int*) – Time to wait for a response before giving up [ms]

Returns

void

Raises

RemoteEngineInitFailed if the server is not responding. –

```
set_start_time()
```

Sets a start time for an operation. Call `get_elapsed_time()` after an operation has finished to get the elapsed time string.

```
upload_project(project_dir_name, fpath)
```

Uploads the zipped project file to server. Project zip file must be ready and the server available before calling this method.

Parameters

- **project_dir_name** (*str*) – Project directory name
- **fpath** (*str*) – Absolute path to zipped project file.

Returns

Project execution job Id

Return type

str

start_execution(*engine_data*, *job_id*)

Sends the start execution request along with job Id and engine (dag) data to the server. Response message data contains the push/pull socket port if execution starts successfully.

Parameters

- **engine_data** (*str*) – Input for SpineEngine as JSON str. Includes most of project.json, settings, etc.
- **job_id** (*str*) – Project execution job Id on server

Returns

Response tuple (event_type, data). Event_type is “server_init_failed”, “remote_execution_init_failed” or “remote_execution_started”. data is an error message or the publish and push sockets ports concatenated with ‘:’.

Return type

tuple

stop_execution(*job_id*)

Sends a request to stop executing the DAG that is managed by this client.

Parameters

job_id (*str*) – Job Id on server to stop

answer_prompt(*job_id*, *item_name*, *accepted*)

Sends a request to answer a prompt from the DAG that is managed by this client.

Parameters

- **job_id** (*str*) – Job Id on server to stop
- **item_name** (*str*) –
- **accepted** (*Bool*) –

download_files(*q*)

Pulls files from server until b’END’ is received.

save_downloaded_file(*b_rel_path*, *file_data*)

Saves downloaded file to project directory.

Parameters

- **b_rel_path** (*bytes*) – Relative path (to project dir) where the file should be saved
- **file_data** (*bytes*) – File as bytes object

retrieve_project(*job_id*)

Retrieves a zipped project file from server.

Parameters

job_id (*str*) – Job Id for finding the project directory on server

Returns

Zipped project file

Return type

bytes

remove_project_from_server(*job_id*)

Sends a request to remove a project directory from server.

Parameters

job_id (*str*) – Job Id for finding the project directory on server

Returns

Message from server

Return type

str

send_is_complete(*persistent_key, cmd*)

Sends a request to process is_complete(cmd) in persistent manager on server and returns the response.

send_issue_persistent_command(*persistent_key, cmd*)

Sends a request to process given command in persistent manager identified by given key. Yields the response string(s) as they arrive from server.

send_get_persistent_completions(*persistent_key, text*)

Requests completions to given text from persistent execution backend.

send_get_persistent_history_item(*persistent_key, text, prefix, backwards*)

Requests the former or latter history item from persistent execution backend.

send_restart_persistent(*persistent_key*)

Sends restart persistent cmd to persistent execution manager backend on server. Yields the messages resulting from this operation to persistent console client.

send_interrupt_persistent(*persistent_key*)

Sends interrupt persistent cmd to persistent execution manager backend on server.

send_kill_persistent(*persistent_key*)

Sends kill persistent cmd to persistent execution manager backend on server.

Parameters

persistent_key (*tuple*) – persistent manager identifier

send_request_to_persistent(*data*)

Sends given data containing persistent_key, command, cmd_to_persistent to Spine Engine Server to be processed by a persistent execution manager backend. Makes a request using REQ socket, parses the response into a ServerMessage, and returns the second part of the data field.

send_request_to_persistent_generator(*data*)

Pulls all messages from server, that were the result of sending given data to Spine Engine Server.

get_elapsed_time()

Returns the elapsed time between now and when self.start_time was set.

Returns

Time string with unit(s)

Return type

str

close()

Closes client sockets, context and thread.

spinetoolbox.spine_db_editor

This subpackage contains GUI files for the Spine db editor.

Subpackages

spinetoolbox.spine_db_editor.mvcmodels

Modules in this package contain classes that represent Spine Toolbox's models (internal data structures) in the Model-View-Controller design pattern. The model classes define an interface that is used by views and delegates to access data in the application.

Submodules

spinetoolbox.spine_db_editor.mvcmodels.alternative_item

Classes to represent items in an alternative tree.

Module Contents

Classes

<i>DBItem</i>	A root item representing a db.
<i>AlternativeItem</i>	An alternative leaf item.

Attributes

<i>_ALTERNATIVE_ICON</i>

```
spinetoolbox.spine_db_editor.mvcmodels.alternative_item._ALTERNATIVE_ICON = '\uf277'
```

```
class spinetoolbox.spine_db_editor.mvcmodels.alternative_item.DBItem(*args, **kwargs)
    Bases:      spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildMixin,
                spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.FetchMoreMixin,
                spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.StandardDBItem
    A root item representing a db.
    property item_type
    property fetch_item_type
    empty_child()
    _make_child(id_)
```

```
class spinetoolbox.spine_db_editor.mvcmodels.alternative_item.AlternativeItem(identifier=None)
```

Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.GrayIfLastMixin`,
`spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EditableMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem`

An alternative leaf item.

Parameters

identifier (*int*, *optional*) – item’s database id

property **item_type**

property **icon_code**

property **tool_tip**

add_item_to_db(*db_item*)

update_item_in_db(*db_item*)

flags(*column*)

Makes items editable.

```
spinetoolbox.spine_db_editor.mvcmodels.alternative_model
```

Contains alternative tree model.

Module Contents

Classes

AlternativeModel

A model to display alternatives in a tree view.

```
class spinetoolbox.spine_db_editor.mvcmodels.alternative_model.AlternativeModel(db_editor,  
                                                                                   db_mgr,  
                                                                                   *db_maps)
```

Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_model_base.TreeModelBase`

A model to display alternatives in a tree view.

Parameters

- **db_editor** (`SpineDBEditor`) –
- **db_mgr** (`SpineDBManager`) –
- ***db_maps** – `DiffDatabaseMapping` instances

static **_make_db_item**(*db_map*)

static **_top_children**()

mimeData(*indexes*)

Stores selected indexes into MIME data.

The MIME data structure contains two distinct data:

- Text representation of the selection
- A pickled dict mapping db identifier to list of alternative ids

Parameters

indexes (*Sequence of QModelIndex*) – selected indexes

Returns

MIME data

Return type

QMimeData

paste_alternative_mime_data(*mime_data, database_item*)

Pastes alternatives from mime data into model.

Parameters

- **mime_data** (*QMimeData*) – mime data
- **database_item** (*alternative_item.DBItem*) – target database item

spinetoolbox.spine_db_editor.mvcmodels.colors

Color constants for models.

Module Contents

spinetoolbox.spine_db_editor.mvcmodels.colors.PIVOT_TABLE_HEADER_COLOR

spinetoolbox.spine_db_editor.mvcmodels.colors.FIXED_FIELD_COLOR

spinetoolbox.spine_db_editor.mvcmodels.colors.SELECTED_COLOR

spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models

Compound models for object parameter definitions and values. These models concatenate several ‘single’ models and one ‘empty’ model.

Module Contents

Classes

<i>CompoundParameterModel</i>	A model that concatenates several single parameter models
<i>CompoundObjectParameterMixin</i>	Implements the interface for populating and filtering a compound object parameter model.
<i>CompoundRelationshipParameterMixin</i>	Implements the interface for populating and filtering a compound relationship parameter model.
<i>CompoundParameterDefinitionMixin</i>	Handles signals from db mngr for parameter_definition models.
<i>CompoundParameterValueMixin</i>	Handles signals from db mngr for parameter_value models.
<i>CompoundObjectParameterDefinitionModel</i>	A model that concatenates several single object parameter_definition models
<i>CompoundRelationshipParameterDefinitionModel</i>	A model that concatenates several single relationship parameter_definition models
<i>CompoundObjectParameterValueModel</i>	A model that concatenates several single object parameter_value models
<i>CompoundRelationshipParameterValueModel</i>	A model that concatenates several single relationship parameter_value models

class `spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel` (*parent*, *db_mngr*, **db_maps*)

Bases: `spinetoolbox.mvcmodels.compound_table_model.CompoundWithEmptyTableModel`

A model that concatenates several single parameter models and one empty parameter model.

Parameters

- **parent** (`SpineDBEditor`) – the parent object
- **db_mngr** (`SpineDBManager`) – the database manager
- ***db_maps** (`DiffDatabaseMapping`) – the database maps included in the model

abstract property entity_class_type

Returns the entity_class type, either ‘object_class’ or ‘relationship_class’.

Returns

str

abstract property item_type

Returns the parameter item type, either ‘parameter_definition’ or ‘parameter_value’.

Returns

str

property _single_model_type

Returns a constructor for the single models.

Returns

SingleParameterModel

property _empty_model_type

Returns a constructor for the empty model.

Returns

EmptyParameterModel

property entity_class_id_key

Returns the key corresponding to the entity_class id (either “object_class_id” or “relationship_class_id”)

Returns

str

property parameter_definition_id_key**canFetchMore**(*_parent*)

Returns True if any of the submodels that haven’t been fetched yet can fetch more.

fetchMore(*_parent*)

Fetches the next sub model and increments the fetched counter.

accepts_item(*item*, *db_map*)**shows_item**(*item*, *db_map*)**abstract _make_header**()**init_model**()

Initializes the model.

get_auto_filter_menu(*logical_index*)

Returns auto filter menu for given logical index from header view.

Parameters**logical_index** (*int*) –**Returns**

ParameterViewFilterMenu

_make_auto_filter_menu(*field*)**headerData**(*section*, *orientation*=*Qt.Orientation.Horizontal*, *role*=*Qt.ItemDataRole.DisplayRole*)

Returns an italic font in case the given column has an autofilter installed.

_create_empty_model()

Returns the empty model for this compound model.

Returns

EmptyParameterModel

filter_accepts_model(*model*)

Returns a boolean indicating whether the given model passes the filter for compound model.

Parameters**model** (*SingleParameterModel*, *EmptyParameterModel*) –**Returns**

bool

_class_filter_accepts_model(*model*)**_auto_filter_accepts_model**(*model*)

accepted_single_models()

Returns a list of accepted single models by calling `filter_accepts_model` on each of them, just for convenience.

Returns

list

_invalidate_filter()

Sets the filter invalid.

stop_invalidating_filter()

Stops invalidating the filter.

set_filter_class_ids(*class_ids*)**clear_auto_filter()****set_auto_filter(*field*, *values*)**

Updates and applies the auto filter.

Parameters

- **field** (*str*) – the field name
- **values** (*dict*) – mapping (db_map, entity_class_id) to set of valid values

_set_compound_auto_filter(*field*, *values*)

Sets the auto filter for given column in the compound model.

Parameters

- **field** (*str*) – the field name
- **values** (*set*) – set of valid (db_map, item_type, id) tuples

_set_single_auto_filter(*model*, *field*)

Sets the auto filter for given column in the given single model.

Parameters

- **model** ([SingleParameterModel](#)) – the model
- **field** (*str*) – the field name

Returns

True if the auto-filtered values were updated, None otherwise

Return type

bool

_row_map_iterator_for_model(*model*)

Yields row map for the given model. Reimplemented to take filter status into account.

Parameters

model ([SingleParameterModel](#), [EmptyParameterModel](#)) –

Yields

tuple – (model, row number) for each accepted row

_models_with_db_map(*db_map*)

Returns a collection of single models with given db_map.

Parameters

db_map ([DiffDatabaseMapping](#)) –

Returns

list

_items_per_class(*items*)

Returns a dict mapping entity_class ids to a set of items.

Parameters**items** (*list*) –**Returns**

dict

handle_items_added(*db_map_data*)

Runs when either parameter definitions or values are added to the dbs. Adds necessary sub-models and initializes them with data. Also notifies the empty model so it can remove rows that are already in.

Parameters**db_map_data** (*dict*) – list of added dict-items keyed by DiffDatabaseMapping**_get_insert_position**(*model*)**_create_single_model**(*db_map*, *entity_class_id*, *committed*)**_add_parameter_data**(*db_map*, *entity_class_id*, *ids*, *committed*)

Creates new single model and resets it with the given parameter ids.

Parameters

- **db_map** (*DiffDatabaseMapping*) – database map
- **entity_class_id** (*int*) – parameter's entity class id
- **ids** (*list of int*) – parameter ids
- **committed** (*bool*) – True if the ids have been committed, False otherwise

handle_items_updated(*db_map_data*)

Runs when either parameter definitions or values are updated in the dbs. Emits dataChanged so the parameter_name column is refreshed.

Parameters**db_map_data** (*dict*) – list of updated dict-items keyed by DiffDatabaseMapping**handle_items_removed**(*db_map_data*)

Runs when either parameter definitions or values are removed from the dbs. Removes the affected rows from the corresponding single models.

Parameters**db_map_data** (*dict*) – list of removed dict-items keyed by DiffDatabaseMapping**_emit_data_changed_for_column**(*field*)

Lazily emits data changed for an entire column.

Parameters**field** (*str*) – the column header**db_item**(*index*)**db_map_id**(*index*)

index_name(*index*)

Generates a name for data at given index.

Parameters

index (*QModelIndex*) – index to model

Returns

label identifying the data

Return type

str

get_set_data_delayed(*index*)

Returns a function that ParameterValueEditor can call to set data for the given index at any later time, even if the model changes.

Parameters

index (*QModelIndex*) –

Returns

function

get_entity_class_id(*index*, *db_map*)

filter_by(*rows_per_column*)

filter_excluding(*rows_per_column*)

class spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.
CompoundObjectParameterMixin

Implements the interface for populating and filtering a compound object parameter model.

property **entity_class_type**

class spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.
CompoundRelationshipParameterMixin

Implements the interface for populating and filtering a compound relationship parameter model.

property **entity_class_type**

class spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.
CompoundParameterDefinitionMixin

Handles signals from db mngr for parameter_definition models.

property **item_type**

class spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.**CompoundParameterValueMixin**(*args, **kwargs)

Handles signals from db mngr for parameter_value models.

property **item_type**

abstract **property** **entity_type**

Returns the entity type, either 'object' or 'relationship' Used by update_single_main_filter.

Returns

str

init_model()

```
set_filter_entity_ids(entity_ids)
```

```
set_filter_alternative_ids(alternative_ids)
```

```
_create_single_model(db_map, entity_class_id, committed)
```

```
class spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundObjectParameterDefinition
```

Bases: [*CompoundObjectParameterMixin*](#), [*CompoundParameterDefinitionMixin*](#),
[*CompoundParameterModel*](#)

A model that concatenates several single object parameter_definition models and one empty object parameter_definition model.

Parameters

- **parent** ([*SpineDBEditor*](#)) – the parent object
- **db_mgr** ([*SpineDBManager*](#)) – the database manager
- ***db_maps** ([*DiffDatabaseMapping*](#)) – the database maps included in the model

```
_make_header()
```

```
class spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundRelationshipParameterDef
```

Bases: [*CompoundRelationshipParameterMixin*](#), [*CompoundParameterDefinitionMixin*](#),
[*CompoundParameterModel*](#)

A model that concatenates several single relationship parameter_definition models and one empty relationship parameter_definition model.

Parameters

- **parent** ([*SpineDBEditor*](#)) – the parent object
- **db_mgr** ([*SpineDBManager*](#)) – the database manager
- ***db_maps** ([*DiffDatabaseMapping*](#)) – the database maps included in the model

```
_make_header()
```

```
class spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundObjectParameterValueModel
```

Bases: [*CompoundObjectParameterMixin*](#), [*CompoundParameterValueMixin*](#), [*CompoundParameterModel*](#)

A model that concatenates several single object parameter_value models and one empty object parameter_value model.

property entity_type

Returns the entity type, either ‘object’ or ‘relationship’ Used by update_single_main_filter.

Returns

str

```
_make_header()
```

class `spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundRelationshipParameterVal`

Bases: `CompoundRelationshipParameterMixin`, `CompoundParameterValueMixin`,
`CompoundParameterModel`

A model that concatenates several single relationship parameter_value models and one empty relationship parameter_value model.

property entity_type

Returns the entity type, either ‘object’ or ‘relationship’ Used by `update_single_main_filter`.

Returns

str

_make_header()

spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_models

Empty models for parameter definitions and values.

Module Contents

Classes

<code>EmptyParameterModel</code>	An empty parameter model.
<code>EmptyParameterDefinitionModel</code>	An empty parameter_definition model.
<code>EmptyObjectParameterDefinitionModel</code>	An empty object parameter_definition model.
<code>EmptyRelationshipParameterDefinitionModel</code>	An empty relationship parameter_definition model.
<code>EmptyParameterValueModel</code>	An empty parameter_value model.
<code>EmptyObjectParameterValueModel</code>	An empty object parameter_value model.
<code>EmptyRelationshipParameterValueModel</code>	An empty relationship parameter_value model.

class `spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel` (*parent*,
header,
db_mgr)

Bases: `spinetoolbox.mvcmodels.empty_row_model.EmptyRowModel`

An empty parameter model.

Initialize class.

Parameters

- **parent** (*Object*) – the parent object, typically a `CompoundParameterModel`
- **header** (*list*) – list of field names for the header
- **db_mgr** (*SpineDBManager*) –

abstract property item_type

The item type, either ‘parameter_value’ or ‘parameter_definition’, required by the `value_field` property.

abstract property entity_class_type

Either ‘object_class’ or ‘relationship_class’.

`property entity_class_id_key`

`property entity_class_name_key`

`property can_be_filtered`

`property value_field`

`accepted_rows()`

`db_item(_index)`

`item_id(_row)`

`data(index, role=Qt.ItemDataRole.DisplayRole)`

Returns the data stored under the given role for the item referred to by the index.

Parameters

- **index** (*QModelIndex*) – Index of item
- **role** (*int*) – Data role

Returns

Item data for given role.

`_make_unique_id(item)`

Returns a unique id for the given model item (name-based). Used by `handle_items_added`.

`handle_items_added(db_map_data)`

Runs when parameter definitions or values are added. Finds and removes model items that were successfully added to the db.

`batch_set_data(indexes, data)`

Sets data for indexes in batch. If successful, add items to db.

`abstract add_items_to_db(db_map_data)`

Add items to db.

Parameters

db_map_data (*dict*) – mapping DiffDatabaseMapping instance to list of items

`_make_item(row)`

`_make_db_map_data(rows)`

Returns model data grouped by database map.

Parameters

rows (*set*) – group data from these rows

Returns

mapping DiffDatabaseMapping instance to list of items

Return type

dict

`class spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterDefinitionModel(*args, **kwargs)`

Bases: `spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInValueListIdMixin`,
`spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInEntityClassIdMixin`,

spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInParameterNameMixin, EmptyParameterModel

An empty parameter_definition model.

Initializes lookup dicts.

property item_type

The item type, either 'parameter_value' or 'parameter_definition', required by the value_field property.

abstract property entity_class_type

See base class.

add_items_to_db(db_map_data)

See base class.

_check_item(item)

Checks if a db item is ready to be inserted.

class spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_models.**EmptyObjectParameterDefinitionModel**

Bases: *EmptyParameterDefinitionModel*

An empty object parameter_definition model.

Initializes lookup dicts.

property entity_class_type

See base class.

class spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_models.**EmptyRelationshipParameterDefinitionModel**

Bases: *EmptyParameterDefinitionModel*

An empty relationship parameter_definition model.

Initializes lookup dicts.

property entity_class_type

See base class.

flags(index)

Additional hack to make the object_class_name_list column non-editable.

class spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_models.**EmptyParameterValueModel**(*args, **kwargs)

Bases: *spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.InferEntityClassIdMixin, spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInAlternativeIdMixin, spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInParameterDefinitionIdsMixin, spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInEntityIdsMixin, spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInEntityClassIdMixin, EmptyParameterModel*

An empty parameter_value model.

Initializes lookup dicts.

property item_type

The item type, either 'parameter_value' or 'parameter_definition', required by the value_field property.

abstract property entity_type

Either ‘object’ or ‘relationship’.

property entity_id_key

property entity_name_key

property entity_name_key_in_cache

_make_unique_id(item)

Returns a unique id for the given model item (name-based). Used by handle_items_added.

add_items_to_db(db_map_data)

See base class.

_check_item(db_map, item)

Checks if a db item is ready to be inserted.

class spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_models.**EmptyObjectParameterValueModel**(*args, **kwargs)

Bases: [EmptyParameterValueModel](#)

An empty object parameter_value model.

Initializes lookup dicts.

property entity_class_type

Either ‘object_class’ or ‘relationship_class’.

property entity_type

Either ‘object’ or ‘relationship’.

_make_unique_id(item)

Returns a unique id for the given model item (name-based). Used by handle_items_added.

class spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_models.**EmptyRelationshipParameterValueModel**

Bases: [spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.MakeRelationshipOnTheFlyMixin](#), [EmptyParameterValueModel](#)

An empty relationship parameter_value model.

Initializes lookup dicts.

property entity_class_type

Either ‘object_class’ or ‘relationship_class’.

property entity_type

Either ‘object’ or ‘relationship’.

_add_entities_on_the_fly = True

_make_unique_id(item)

Returns a unique id for the given model item (name-based). Used by handle_items_added.

_make_item(row)

add_items_to_db(db_map_data)

See base class.

spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item

Classes to represent entities in a tree.

Module Contents**Classes**

<i>EntityRootItem</i>	A tree item that may belong in multiple databases.
<i>ObjectTreeRootItem</i>	An object tree root item.
<i>RelationshipTreeRootItem</i>	A relationship tree root item.
<i>EntityClassItem</i>	An entity_class item.
<i>ObjectClassItem</i>	An object_class item.
<i>RelationshipClassItem</i>	A relationship_class item.
<i>ObjectRelationshipClassItem</i>	A relationship_class item.
<i>MembersItem</i>	An item to hold members of a group.
<i>EntityItem</i>	An entity item.
<i>ObjectItem</i>	An object item.
<i>MemberObjectItem</i>	A member object item.
<i>RelationshipItem</i>	A relationship item.

```
class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.EntityRootItem(model=None,  
                                                                           db_map_ids=None)
```

Bases: [*spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem*](#)

A tree item that may belong in multiple databases.

Init class.

Parameters

- **model** ([*MinimalTreeModel*](#), *optional*) – item’s model
- **db_map_ids** (*dict*, *optional*) – maps instances of *DiffDatabaseMapping* to the id of the item in that db

property display_id

See super class.

property display_icon

Returns an icon to display next to the name. Reimplement in subclasses to return something nice.

property display_data

See super class.

item_type = 'root'

set_data(*column*, *value*, *role*)

See base class.

```
class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.ObjectTreeRootItem(model=None,  
                                                                           db_map_ids=None)
```

Bases: [*EntityRootItem*](#)

An object tree root item.

Init class.

Parameters

- **model** ([MinimalTreeModel](#), *optional*) – item’s model
- **db_map_ids** (*dict*, *optional*) – maps instances of DiffDatabaseMapping to the id of the item in that db

property child_item_class

Returns ObjectClassItem.

item_type = 'root'

```
class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.RelationshipTreeRootItem(model=None,
                                                                                       db_map_ids=None)
```

Bases: [EntityRootItem](#)

A relationship tree root item.

Init class.

Parameters

- **model** ([MinimalTreeModel](#), *optional*) – item’s model
- **db_map_ids** (*dict*, *optional*) – maps instances of DiffDatabaseMapping to the id of the item in that db

property child_item_class

Returns RelationshipClassItem.

item_type = 'root'

```
class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.EntityClassItem(model=None,
                                                                                db_map_ids=None)
```

Bases: [spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem](#)

An entity_class item.

Init class.

Parameters

- **model** ([MinimalTreeModel](#), *optional*) – item’s model
- **db_map_ids** (*dict*, *optional*) – maps instances of DiffDatabaseMapping to the id of the item in that db

property display_icon

Returns class icon.

_display_icon(*for_group=False*)

data(*column*, *role=Qt.ItemDataRole.DisplayRole*)

Returns data for given column and role.

accepts_item(*item*, *db_map*)

set_data(*column*, *value*, *role*)

See base class.

```
class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.ObjectClassItem(*args,  
                                                                              **kwargs)
```

Bases: [EntityClassItem](#)

An object_class item.

Init class.

Parameters

- **model** ([MinimalTreeModel](#), *optional*) – item's model
- **db_map_ids** (*dict*, *optional*) – maps instances of DiffDatabaseMapping to the id of the item in that db

property child_item_class

Returns ObjectItem.

property _children_sort_key

Reimplemented so groups are above non-groups.

item_type = 'object_class'

default_parameter_data()

Return data to put as default in a parameter table when this item is selected.

_can_fetch_more_entity_groups()

can_fetch_more()

Returns whether this item can fetch more.

_fetch_more_entity_groups()

fetch_more()

Fetches children from all associated databases.

_accepts_entity_group_item(item, db_map)

_handle_entity_group_items_added(db_map_data)

_handle_entity_group_items_updated(db_map_data)

tear_down()

Do stuff after the item has been removed.

revitalize()

See base class

```
class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.RelationshipClassItem(model=None,  
                                                                              db_map_ids=None)
```

Bases: [EntityClassItem](#)

A relationship_class item.

Init class.

Parameters

- **model** ([MinimalTreeModel](#), *optional*) – item's model
- **db_map_ids** (*dict*, *optional*) – maps instances of DiffDatabaseMapping to the id of the item in that db

property child_item_class

Returns RelationshipItem.

visual_key = ['name', 'object_class_name_list']

item_type = 'relationship_class'

default_parameter_data()

Return data to put as default in a parameter table when this item is selected.

class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.**ObjectRelationshipClassItem**(*model=None, db_map_ids=None*)

Bases: [RelationshipClassItem](#)

A relationship_class item.

Init class.

Parameters

- **model** ([MinimalTreeModel](#), *optional*) – item’s model
- **db_map_ids** (*dict*, *optional*) – maps instances of DiffDatabaseMapping to the id of the item in that db

set_data(*column, value, role*)

See base class.

accepts_item(*item, db_map*)

class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.**MembersItem**(*model=None, db_map_ids=None*)

Bases: [EntityClassItem](#)

An item to hold members of a group.

Init class.

Parameters

- **model** ([MinimalTreeModel](#), *optional*) – item’s model
- **db_map_ids** (*dict*, *optional*) – maps instances of DiffDatabaseMapping to the id of the item in that db

property display_id

Returns an id for display based on the display key. This id must be the same across all db_maps. If it’s not, this property becomes None and measures need to be taken (see `update_children_by_id`).

property display_data

Returns the name for display.

property child_item_class

Returns MemberObjectItem.

item_type = 'members'

db_map_data(*db_map*)

Returns data for this item as if it was indeed an object class.

_display_icon(*for_group=False*)

Returns icon for this item as if it was indeed an object class.

accepts_item(*item*, *db_map*)

default_parameter_data()

Return data to put as default in a parameter table when this item is selected.

data(*column*, *role*=*Qt.ItemDataRole.DisplayRole*)

Returns data for given column and role.

```
class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.EntityItem(*args,
                                                                           is_group=False,
                                                                           **kwargs)
```

Bases: *spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem*

An entity item.

Init class.

Parameters

- **model** (*MinimalTreeModel*, *optional*) – item’s model
- **db_map_ids** (*dict*, *optional*) – maps instances of DiffDatabaseMapping to the id of the item in that db

property display_icon

Returns corresponding class icon.

update(*is_group*=*False*)

should_be_merged()

data(*column*, *role*=*Qt.ItemDataRole.DisplayRole*)

Returns data for given column and role.

set_data(*column*, *value*, *role*)

See base class.

_can_fetch_members_item()

_fetch_members_item()

can_fetch_more()

Returns whether this item can fetch more.

fetch_more()

Fetches children from all associated databases.

```
class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.ObjectItem(*args,
                                                                           is_group=False,
                                                                           **kwargs)
```

Bases: *EntityItem*

An object item.

Init class.

Parameters

- **model** (*MinimalTreeModel*, *optional*) – item’s model
- **db_map_ids** (*dict*, *optional*) – maps instances of DiffDatabaseMapping to the id of the item in that db

property child_item_class

Child class is always *ObjectRelationshipClassItem*.

item_type = 'object'

default_parameter_data()

Return data to put as default in a parameter table when this item is selected.

accepts_item(*item*, *db_map*)

```
class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.MemberObjectItem(*args,
                                                                              is_group=False,
                                                                              **kwargs)
```

Bases: *ObjectItem*

A member object item.

Init class.

Parameters

- **model** (*MinimalTreeModel*, *optional*) – item's model
- **db_map_ids** (*dict*, *optional*) – maps instances of DiffDatabaseMapping to the id of the item in that db

property display_icon

Returns corresponding class icon.

property display_data

“Returns the name for display.

item_type = 'entity_group'

visual_key = ['member_name']

has_children()

Returns whether this item has or could have children.

can_fetch_more()

Returns whether this item can fetch more.

```
class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.RelationshipItem(*args,
                                                                              **kwargs)
```

Bases: *EntityItem*

A relationship item.

Overridden method to make sure we never try to fetch this item.

property object_name_list**property display_data**

“Returns the name for display.

property edit_data

visual_key = ['name', 'object_name_list']

item_type = 'relationship'

default_parameter_data()

Return data to put as default in a parameter table when this item is selected.

has_children()

Returns whether this item has or could have children.

can_fetch_more()

Returns whether this item can fetch more.

is_valid()

Checks that the grand parent object is still in the relationship.

spinetoolbox.spine_db_editor.mvcmodels.entity_tree_models

Models to represent entities in a tree.

Module Contents

Classes

<i>ObjectTreeModel</i>	An 'object-oriented' tree model.
<i>RelationshipTreeModel</i>	A relationship-oriented tree model.

class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_models.**ObjectTreeModel**(*db_editor*,
db_mgr,
**db_maps*)

Bases: *spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_model.MultiDBTreeModel*

An 'object-oriented' tree model.

Init class.

Parameters

- **db_editor** (*SpineDBEditor*) –
- **db_mgr** (*SpineDBManager*) – A manager for the given db_maps
- **db_maps** (*iter*) – DiffDatabaseMapping instances

property root_item_type

Implement in subclasses to create a model specific to any entity type.

find_next_relationship_index(*index*)

Find and return next occurrence of relationship item.

class spinetoolbox.spine_db_editor.mvcmodels.entity_tree_models.**RelationshipTreeModel**(*db_editor*,
db_mgr,
**db_maps*)

Bases: *spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_model.MultiDBTreeModel*

A relationship-oriented tree model.

Init class.

Parameters

- **db_editor** ([SpineDBEditor](#)) –
- **db_mgr** ([SpineDBManager](#)) – A manager for the given db_maps
- **db_maps** (*iter*) – DiffDatabaseMapping instances

property root_item_type

Implement in subclasses to create a model specific to any entity type.

spinetoolbox.spine_db_editor.mvcmodels.frozen_table_model

Contains FrozenTableModel class.

Module Contents**Classes***FrozenTableModel*

Used by custom_qtableview.FrozenTableView

```
class spinetoolbox.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel(db_mgr,
                                                                              par-
                                                                              ent=None)
```

Bases: PySide6.QtCore.QAbstractTableModel

Used by custom_qtableview.FrozenTableView

Parameters

- **db_mgr** ([SpineDBManager](#)) – database manager
- **parent** (*QObject*, *optional*) – parent object

property headers**selected_row_changed****set_headers**(*headers*)

Sets headers for the header row wiping data.

This method does nothing if the new headers are equal to existing ones.

Parameters

headers (*Iterable of str*) – headers

clear_model()**add_values**(*data*)

Adds more frozen values that aren't in the table already.

Parameters

data (*set of tuple*) – frozen values

remove_values(*data*)

Removes frozen values from the table.

Parameters

data (*set of tuple*) – frozen values

clear_selected()

Clears selected row.

set_selected(*row*)

Changes selected row.

Parameters

row (*int*) – row index

get_frozen_value()

Return currently selected frozen value.

Returns

frozen value

Return type

tuple

rowCount(*parent=QModelIndex()*)

columnCount(*parent=QModelIndex()*)

row(*index*)

insert_column_data(*header, values, column*)

Inserts new column with given header.

Parameters

- **header** (*str*) – frozen header
- **values** (*set of tuple*) – column's values
- **column** (*int*) – position

remove_column(*column*)

Removes column and makes rows unique.

Parameters

column (*int*) – column to remove

moveColumns(*sourceParent, sourceColumn, count, destinationParent, destinationChild*)

_keep_sorted()

Sorts the data table.

_unique_values()

Turns non-header data into sets of unique values on each column.

Returns

each column's unique values

Return type

list of set

_find_first(*row_data, mask_column=None*)

Finds first row that matches given row data.

Parameters

- **row_data** (*tuple*) – row data to search for
- **mask_column** (*int, optional*) – ignored column

Returns

row index

Return type

int

data(*index*, *role*=*Qt.ItemDataRole.DisplayRole*)**_tooltip_from_data**(*row*, *column*)

Resolves item tooltip which is usually its description.

Parameters

- **row** (*int*) – row
- **column** (*int*) – column

Returns

value's tooltip

Return type

str

_name_from_data(*value*, *header*)

Resolves item name.

Parameters

- **value** (*tuple* or *DatabaseMappingBase*) – cell value
- **header** (*str*) – column header

Returns

value's name

Return type

str

spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_modelContains *ItemMetadataTableModel* and associated functionality.**Module Contents****Classes**

<i>ExtraColumn</i>	Identifiers for hidden table columns.
<i>ItemType</i>	Allowed item types.
<i>ItemMetadataTableModel</i>	Model for entity and parameter value metadata.

class spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model.**ExtraColumn**

Bases: enum.IntEnum

Identifiers for hidden table columns.

Initialize self. See help(type(self)) for accurate signature.

ITEM_METADATA_ID

METADATA_ID

class spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model.ItemType

Bases: enum.Enum

Allowed item types.

ENTITY

VALUE

class spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel(*db_mgr*,
db_maps,
db_editor)

Bases: [spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase](#)

Model for entity and parameter value metadata.

Parameters

- **db_mgr** ([SpineDBManager](#)) – database manager
- **db_maps** (*Iterable of DatabaseMappingBase*) – database maps
- **db_editor** ([SpineDBEditor](#)) – DB editor

_ITEM_NAME_KEY = 'metadata_name'

_ITEM_VALUE_KEY = 'metadata_value'

_fetch_parents()

Yields fetch parents for this model.

Yields

FetchParent

clear()

Clears the model.

static _make_hidden_adder_columns()

See base class.

_accepts_entity_metadata_item(*item*, *db_map*)

_accepts_parameter_value_metadata_item(*item*, *db_map*)

set_entity_ids(*db_map_ids*)

Sets the model to show metadata from given entity.

Parameters

db_map_ids (*dict*) – mapping from database mapping to entity's id in that database

set_parameter_value_ids(*db_map_ids*)

Sets the model to show metadata from given parameter value.

Parameters

db_map_ids (*dict*) – mapping from database mapping to value's id in that database

`_reset_metadata(item_type, db_map_ids)`

Resets model.

Parameters

- **`item_type`** (*ItemType*) – current item type
- **`db_map_ids`** (*dict*) – mapping from database mapping to value's id in that database

`_reset_fetch_parents()`

`_add_data_to_db_mgr(name, value, db_map)`

See base class.

`_update_data_in_db_mgr(id_, name, value, db_map)`

See base class

`rollback(_db_maps)`

Rolls back changes in database.

Parameters

`db_maps` (*Iterable of DiffDatabaseMapping*) – database mappings that have been rolled back

`flags(index)`

`static _ids_from_added_item(item)`

See base class.

`static _extra_cells_from_added_item(item)`

See base class.

`_set_extra_columns(row, ids)`

See base class.

`_database_table_name()`

See base class

`_row_id(row)`

See base class.

`add_item_metadata(db_map_data)`

Adds new item metadata from database manager to the model.

Parameters

`db_map_data` (*dict*) – added items keyed by database mapping

`update_item_metadata(db_map_data)`

Updates item metadata in model after it has been updated in databases.

Parameters

`db_map_data` (*dict*) – updated metadata records

`remove_item_metadata(db_map_data)`

Removes item metadata from model after it has been removed from databases.

Parameters

`db_map_data` (*dict*) – removed items keyed by database mapping

`spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model`

Contains *MetadataTableModel* and associated functionality.

Module Contents

Classes

<i>ExtraColumn</i>	Identifiers for hidden table columns.
<i>MetadataTableModel</i>	Model for metadata.

class `spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.ExtraColumn`

Bases: `enum.IntEnum`

Identifiers for hidden table columns.

Initialize self. See `help(type(self))` for accurate signature.

ID

class `spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel(db_mgr,
db_maps,
db_editor)`

Bases: `spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase`

Model for metadata.

Parameters

- **db_mgr** (`SpineDBManager`) – database manager
- **db_maps** (*Iterable of DatabaseMappingBase*) – database maps
- **db_editor** (`SpineDBEditor`) – DB editor

_ITEM_NAME_KEY = 'name'

_ITEM_VALUE_KEY = 'value'

static `_make_hidden_adder_columns()`

See base class.

`_add_data_to_db_mgr(name, value, db_map)`

See base class.

`_update_data_in_db_mgr(id_, name, value, db_map)`

See base class

rollback(*db_maps*)

Rolls back changes in database.

Parameters

db_maps (*Iterable of DiffDatabaseMapping*) – database mappings that have been rolled back

`_database_table_name()`

See base class

`_row_id(row)`

See base class.

`flags(index)`

`_fetch_parents()`

Yields fetch parents for this model.

Yields

FetchParent

`static _ids_from_added_item(item)`

See base class.

`static _extra_cells_from_added_item(item)`

See base class.

`_set_extra_columns(row, ids)`

See base class.

`add_metadata(db_map_data)`

Adds new metadata from database manager to the model.

Parameters

`db_map_data` (*dict*) – added metadata items keyed by database mapping

`update_metadata(db_map_data)`

Updates model according to data received from database manager.

Parameters

`db_map_data` (*dict*) – updated metadata items keyed by database mapping

`remove_metadata(db_map_data)`

Removes metadata from model after it has been removed from databases.

Parameters

`db_map_data` (*dict*) – removed items keyed by database mapping

`spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base`

Contains base class for metadata table models associated functionality.

Module Contents

Classes

<i>Column</i>	Identifiers for visible table columns.
<i>MetadataTableModelBase</i>	Base for metadata table models

Attributes

FLAGS_FIXED

FLAGS_EDITABLE

class spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.**Column**

Bases: `enum.IntEnum`

Identifiers for visible table columns.

Initialize self. See `help(type(self))` for accurate signature.

NAME = 0

VALUE = 1

DB_MAP = 2

static max()

spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.**FLAGS_FIXED**

spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.**FLAGS_EDITABLE**

class spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.**MetadataTableModelBase**(*db_mgr*,
db_maps,
db_editor)

Bases: `PySide6.QtCore.QAbstractTableModel`

Base for metadata table models

Parameters

- **db_mgr** (*SpineDBManager*) – database manager
- **db_maps** (*Iterable of DatabaseMappingBase*) – database maps
- **db_editor** (*SpineDBEditor*) – DB editor

msg_error

Emitted when an error occurs.

_HEADER = ('name', 'value', 'database')

_ITEM_NAME_KEY

_ITEM_VALUE_KEY

classmethod **_make_adder_row**(*default_db_map*)

Generates a new empty last row.

Parameters

default_db_map (*DiffDatabaseMapping*) – initial database mapping

Returns

empty row

Return type

list

abstract static _make_hidden_adder_columns()

Creates hidden extra columns for adder row.

Returns

extra columns

Return type

list

set_db_maps(*db_maps*)

Changes current database mappings.

Parameters

db_maps (*Iterable of DiffDatabaseMapping*) – database mappings

abstract _fetch_parents()

Yields fetch parents for this model.

Yields

FetchParent

canFetchMore()

fetchMore()

rowCount(*parent=QModelIndex()*)

columnCount(*parent=QModelIndex()*)

data(*index, role=Qt.ItemDataRole.DisplayRole*)

abstract _add_data_to_db_mgr(*name, value, db_map*)

Tells database manager to start adding data.

Parameters

- **name** (*str*) – metadata name
- **value** (*str*) – metadata value
- **db_map** (*DiffDatabaseMapping*) – database mapping

abstract _update_data_in_db_mgr(*id_, name, value, db_map*)

Tells database manager to start updating data.

Parameters

- **id** (*int*) – database id
- **name** (*str*) – metadata name
- **value** (*str*) – metadata value
- **db_map** (*DiffDatabaseMapping*) – database mapping

setData(*index, value, role=Qt.ItemDataRole.EditRole*)

batch_set_data(*indexes, values*)

Sets data in multiple indexes simultaneously.

Parameters

- **indexes** (*Iterable of QModelIndex*) – indexes to set
- **values** (*Iterable of str*) – values corresponding to indexes

headerData(*section, orientation, role=Qt.ItemDataRole.DisplayRole*)

insertRows(*row, count, parent=QModelIndex()*)

abstract _database_table_name()

Returns primary database table name.

Returns

table name

Return type

str

abstract _row_id(*row*)

Returns a unique row id.

Parameters

row (*list*) – data table row

Returns

id or None

Return type

int

removeRows(*first, count, parent=QModelIndex()*)

abstract static _ids_from_added_item(*item*)

Returns ids that uniquely identify an added database item.

Parameters

item (*dict*) – added item

Returns

unique identifier

Return type

Any

abstract static _extra_cells_from_added_item(*item*)

Constructs extra cells for data row from added database item.

Parameters

item (*dict*) – added item

Returns

extra cells

Return type

list

abstract _set_extra_columns(*row, ids*)

Sets extra columns for data row.

Parameters

- **row** (*list*) – data row
- **ids** (*Any*) –

`_add_data(db_map_data)`

Adds new data from database manager to the model.

Parameters

`db_map_data` (*dict*) – added items keyed by database mapping

`_update_data(db_map_data, id_column)`

Update data table after database update.

Parameters

- **`db_map_data`** (*dict*) – updated items keyed by database mapping
- **`id_column`** (*int*) – column that contains item ids

`_remove_data(db_map_data, id_column)`

Removes data from model after it has been removed from databases.

Parameters

- **`db_map_data`** (*dict*) – removed items keyed by database mapping
- **`id_column`** (*int*) – column that contains item ids

`sort(column, order=Qt.AscendingOrder)`

`_find_db_map(codename)`

Finds database mapping with given codename.

Parameters

`codename` (*str*) – database mapping's code name

Returns

database mapping or None if not found

Return type

DiffDatabaseMapping

`_reserved_metadata()`

Collects metadata names and values that are already in database.

Returns

mapping from database mapping to metadata name and value

Return type

dict

`spinetoolbox.spine_db_editor.mvcmodels.mime_types`

Contains mime types used by the models.

Module Contents

```
spinetoolbox.spine_db_editor.mvcmodels.mime_types.ALTERNATIVE_DATA =  
'application/vnd.spinetoolbox.alternative'
```

```
spinetoolbox.spine_db_editor.mvcmodels.mime_types.SCENARIO_DATA =  
'application/vnd.spinetoolbox.scenario'
```

```
spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item
```

Base classes to represent items from multiple databases in a tree.

Module Contents

Classes

<i>MultiDBTreeItem</i>	A tree item that may belong in multiple databases.
------------------------	--

```
class spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem(model=None,  
                                         db_map_ids=None)
```

Bases: *spinetoolbox.mvcmodels.minimal_tree_model.TreeItem*

A tree item that may belong in multiple databases.

Init class.

Parameters

- **model** (*MinimalTreeModel*, optional) – item’s model
- **db_map_ids** (*dict*, optional) – maps instances of *DiffDatabaseMapping* to the id of the item in that db

property db_mgr

property child_item_class

Returns the type of child items. Reimplement in subclasses to return something more meaningful.

property display_id

Returns an id for display based on the display key. This id must be the same across all db_maps. If it’s not, this property becomes None and measures need to be taken (see *update_children_by_id*).

property display_data

Returns the name for display.

property display_database

Returns the database for display.

property display_icon

Returns an icon to display next to the name. Reimplement in subclasses to return something nice.

property first_db_map

Returns the first associated db_map.

property last_db_map

Returns the last associated db_map.

property db_maps

Returns a list of all associated db_maps.

property db_map_ids

Returns dict with db_map as key and id as value

property _children_sort_key**property fetch_item_type****item_type**

Item type identifier string. Should be set to a meaningful value by subclasses.

visual_key = ['name']**child_number()**

Returns the rank of this item within its parent or -1 if it's an orphan.

abstract set_data(*column, value, role*)

Sets data for this item.

Parameters

- **column** (*int*) – column index
- **value** (*object*) – a new value
- **role** (*int*) – role of the new value

Returns

True if data was set successfully, False otherwise

Return type

bool

add_db_map_id(*db_map, id_*)

Adds id for this item in the given db_map.

take_db_map(*db_map*)

Removes the mapping for given db_map and returns it.

_deep_refresh_children()

Refreshes children after taking db_maps from them. Called after removing and updating children for this item.

deep_remove_db_map(*db_map*)

Removes given db_map from this item and all its descendants.

deep_take_db_map(*db_map*)

Removes given db_map from this item and all its descendants, and returns a new item from the db_map's data.

Returns

MultiDBTreeItem, NoneType

deep_merge(*other*)

Merges another item and all its descendants into this one.

db_map_id(*db_map*)

Returns the id for this item in given *db_map* or None if not present.

db_map_data(*db_map*)

Returns data for this item in given *db_map* or None if not present.

db_map_data_field(*db_map*, *field*, *default=None*)

Returns field from data for this item in given *db_map* or None if not found.

_create_new_children(*db_map*, *children_ids*, ***kwargs*)

Creates new items from ids associated to a db map.

Parameters

- **db_map** (*DiffDatabaseMapping*) – create children for this *db_map*
- **children_ids** (*iter*) – create children from these ids

Returns

new children

Return type

list of *MultiDBTreeItem*

_merge_children(*new_children*)

Merges new children into this item. Ensures that each child has a valid display id afterwards.

_insert_children_sorted(*new_children*)

Inserts and sorts children.

will_have_children_change()

Notifies the view that the model's layout has changed. This triggers a repaint so this item will be painted gray if no children.

can_fetch_more()

Returns whether this item can fetch more.

fetch_more()

Fetches children from all associated databases.

fetch_more_if_possible()

accepts_item(*item*, *db_map*)

handle_items_added(*db_map_data*)

handle_items_removed(*db_map_data*)

handle_items_updated(*db_map_data*)

append_children_by_id(*db_map_ids*, ***kwargs*)

Appends children by id.

Parameters

db_map_ids (*dict*) – maps *DiffDatabaseMapping* instances to list of ids

remove_children_by_id(*db_map_ids*)

Removes children by id.

Parameters

db_map_ids (*dict*) – maps *DiffDatabaseMapping* instances to list of ids

is_valid()

Checks if the item is still valid after an update operation.

update_children_by_id(*db_map_ids*, *kwargs*)**

Updates children by id. Essentially makes sure all children have a valid display id after updating the underlying data. These may require ‘splitting’ a child into several for different dbs or merging two or more children from different dbs.

Examples of problems:

- The user renames an object_class in one db but not in the others → we need to split
- The user renames an object_class and the new name is already ‘taken’ by another object_class in another db_map → we need to merge

Parameters

db_map_ids (*dict*) – maps DiffDatabaseMapping instances to list of ids

update(*kwargs*)****should_be_merged()****insert_children(*position*, *children*)**

Insert new children at given position. Returns a boolean depending on how it went.

Parameters

- **position** (*int*) – insert new items here
- **children** (*iter*) – insert items from this iterable

remove_children(*position*, *count*)

Removes count children starting from the given position.

clear_children()

Clear children list.

_refresh_child_map()

Recomputes the child map.

find_row(*db_map*, *id_*)**find_children_by_id(*db_map*, **ids*, *reverse=True*)**

Generates children with the given ids in the given db_map. If the first id is None, then generates *all* children with the given db_map.

find_rows_by_id(*db_map*, **ids*, *reverse=True*)**_find_unsorted_rows_by_id(*db_map*, **ids*)**

Generates rows corresponding to children with the given ids in the given db_map. If the only id given is None, then generates rows corresponding to *all* children with the given db_map.

data(*column*, *role=Qt.ItemDataRole.DisplayRole*)

Returns data for given column and role.

default_parameter_data()

Returns data to set as default in a parameter table when this item is selected.

tear_down()

Do stuff after the item has been removed.

revitalize()

Reverts tear down operation

spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_model

A base model class to represent items from multiple databases in a tree.

Module Contents

Classes

<i>MultiDBTreeModel</i>	Base class for all tree models in Spine db editor.
-------------------------	--

```
class spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_model.MultiDBTreeModel(db_editor,
                                                                                    db_mgr,
                                                                                    *db_maps)
```

Bases: *spinetoolbox.mvcmodels.minimal_tree_model.MinimalTreeModel*

Base class for all tree models in Spine db editor.

Init class.

Parameters

- **db_editor** (*SpineDBEditor*) –
- **db_mgr** (*SpineDBManager*) – A manager for the given db_maps
- **db_maps** (*iter*) – DiffDatabaseMapping instances

abstract property root_item_type

Implement in subclasses to create a model specific to any entity type.

property root_item

property root_index

build_tree()

Builds tree.

columnCount (*parent=QModelIndex()*)

headerData (*section, orientation, role=Qt.ItemDataRole.DisplayRole*)

find_items (*db_map, path_prefix, fetch=False*)

Returns items at given path prefix.

spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins

Miscellaneous mixins for parameter models

Module Contents**Classes**

<i>ConvertToDBMixin</i>	Base class for all mixins that convert model items (name-based) into database items (id-based).
<i>FillInAlternativeIdMixin</i>	Fills in alternative names.
<i>FillInParameterNameMixin</i>	Fills in parameter names.
<i>FillInValueListIdMixin</i>	Fills in value list ids.
<i>FillInEntityClassIdMixin</i>	Fills in entity_class ids.
<i>FillInEntityIdsMixin</i>	Fills in entity ids.
<i>FillInParameterDefinitionIdsMixin</i>	Fills in parameter_definition ids.
<i>InferEntityClassIdMixin</i>	Infers entity class ids.
<i>ImposeEntityClassIdMixin</i>	Imposes entity class ids.
<i>MakeRelationshipOnTheFlyMixin</i>	Makes relationships on the fly.

class spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.**ConvertToDBMixin**

Base class for all mixins that convert model items (name-based) into database items (id-based).

build_lookup_dictionary(*db_map_data*)

Begins an operation to convert items.

_convert_to_db(*item*, *db_map*)

Returns a db item (id-based) from the given model item (name-based).

Parameters

- **item** (*dict*) – the model item
- **db_map** (*DiffDatabaseMapping*) – the database where the resulting item belongs

Returns

the db item list: error log

Return type

dict

class spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.**FillInAlternativeIdMixin**(*args, **kwargs)

Bases: *ConvertToDBMixin*

Fills in alternative names.

Initializes lookup dicts.

build_lookup_dictionary(*db_map_data*)

Builds a name lookup dictionary for the given data.

Parameters

db_map_data (*dict*) – lists of model items keyed by DiffDatabaseMapping

_convert_to_db(*item*, *db_map*)

Returns a db item (id-based) from the given model item (name-based).

Parameters

- **item** (*dict*) – the model item
- **db_map** (*DiffDatabaseMapping*) – the database where the resulting item belongs

Returns

the db item list: error log

Return type

dict

class spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.**FillInParameterNameMixin**

Bases: [ConvertToDBMixin](#)

Fills in parameter names.

_convert_to_db(*item*, *db_map*)

Returns a db item (id-based) from the given model item (name-based).

Parameters

- **item** (*dict*) – the model item
- **db_map** (*DiffDatabaseMapping*) – the database where the resulting item belongs

Returns

the db item list: error log

Return type

dict

class spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.**FillInValueListIdMixin**(*args,
**kwargs)

Bases: [ConvertToDBMixin](#)

Fills in value list ids.

Initializes lookup dicts.

build_lookup_dictionary(*db_map_data*)

Builds a name lookup dictionary for the given data.

Parameters

db_map_data (*dict*) – lists of model items keyed by DiffDatabaseMapping

_convert_to_db(*item*, *db_map*)

Returns a db item (id-based) from the given model item (name-based).

Parameters

- **item** (*dict*) – the model item
- **db_map** (*DiffDatabaseMapping*) – the database where the resulting item belongs

Returns

the db item list: error log

Return type

dict

`_fill_in_value_list_id(item, db_map)`

Fills in the value list id in the given db item.

Parameters

- **item** (*dict*) – the db item
- **db_map** (*DiffDatabaseMapping*) – the database where the given item belongs

Returns

error log

Return type

list

```
class spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInEntityClassIdMixin(*args,
                                                                                       **kwargs)
```

Bases: [ConvertToDBMixin](#)

Fills in entity_class ids.

Initializes lookup dicts.

`build_lookup_dictionary(db_map_data)`

Builds a name lookup dictionary for the given data.

Parameters

- **db_map_data** (*dict*) – lists of model items keyed by DiffDatabaseMapping

`_fill_in_entity_class_id(item, db_map)`

Fills in the entity_class id in the given db item.

Parameters

- **item** (*dict*) – the db item
- **db_map** (*DiffDatabaseMapping*) – the database where the given item belongs

Returns

error log

Return type

list

`_convert_to_db(item, db_map)`

Returns a db item (id-based) from the given model item (name-based).

Parameters

- **item** (*dict*) – the model item
- **db_map** (*DiffDatabaseMapping*) – the database where the resulting item belongs

Returns

the db item list: error log

Return type

dict

```
class spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInEntityIdsMixin(*args,
                                                                                     **kwargs)
```

Bases: [ConvertToDBMixin](#)

Fills in entity ids.

Initializes lookup dicts.

_add_entities_on_the_fly = False

build_lookup_dictionary(*db_map_data*)

Builds a name lookup dictionary for the given data.

Parameters

db_map_data (*dict*) – lists of model items keyed by DiffDatabaseMapping

_fill_in_entity_ids(*item*, *db_map*)

Fills in all possible entity ids keyed by entity_class id in the given db item (as there can be more than one entity for the same name).

Parameters

- **item** (*dict*) – the db item
- **db_map** (*DiffDatabaseMapping*) – the database where the given item belongs

Returns

error log

Return type

list

_convert_to_db(*item*, *db_map*)

Returns a db item (id-based) from the given model item (name-based).

Parameters

- **item** (*dict*) – the model item
- **db_map** (*DiffDatabaseMapping*) – the database where the resulting item belongs

Returns

the db item list: error log

Return type

dict

class spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.**FillInParameterDefinitionIdsMixin**(*args, **kwargs)

Bases: [ConvertToDBMixin](#)

Fills in parameter_definition ids.

Initializes lookup dicts.

build_lookup_dictionary(*db_map_data*)

Builds a name lookup dictionary for the given data.

Parameters

db_map_data (*dict*) – lists of model items keyed by DiffDatabaseMapping

_fill_in_parameter_ids(*item*, *db_map*)

Fills in all possible parameter_definition ids keyed by entity_class id in the given db item (as there can be more than one parameter_definition for the same name).

Parameters

- **item** (*dict*) – the db item
- **db_map** (*DiffDatabaseMapping*) – the database where the given item belongs

Returns

error log

Return type

list

_convert_to_db(*item*, *db_map*)

Returns a db item (id-based) from the given model item (name-based).

Parameters

- **item** (*dict*) – the model item
- **db_map** (*DiffDatabaseMapping*) – the database where the resulting item belongs

Returns

the db item list: error log

Return type

dict

class spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.**InferEntityClassIdMixin**Bases: [ConvertToDBMixin](#)

Infers entity class ids.

_convert_to_db(*item*, *db_map*)

Returns a db item (id-based) from the given model item (name-based).

Parameters

- **item** (*dict*) – the model item
- **db_map** (*DiffDatabaseMapping*) – the database where the resulting item belongs

Returns

the db item list: error log

Return type

dict

_infer_and_fill_in_entity_class_id(*item*, *db_map*)

Fills the entity_class id in the given db item, by intersecting entity ids and parameter ids. Then picks the correct entity id and parameter_definition id. Also sets the inferred entity_class name in the model.

Parameters

- **item** (*dict*) – the db item
- **db_map** (*DiffDatabaseMapping*) – the database where the given item belongs

Returns

error log

Return type

list

class spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.**ImposeEntityClassIdMixin**Bases: [ConvertToDBMixin](#)

Imposes entity class ids.

_convert_to_db(*item*, *db_map*)

Returns a db item (id-based) from the given model item (name-based).

Parameters

- **item** (*dict*) – the model item
- **db_map** (*DiffDatabaseMapping*) – the database where the resulting item belongs

Returns

the db item list: error log

Return type

dict

_impose_entity_class_id(*item*, *db_map*)

Imposes the entity_class id from the model, to pick the correct entity id and parameter_definition id.

Parameters

- **item** (*dict*) – the db item
- **db_map** (*DiffDatabaseMapping*) – the database where the given item belongs

Returns

error log

Return type

list

class spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.**MakeRelationshipOnTheFlyMixin**(*args,
**kwargs)

Makes relationships on the fly.

Initializes lookup dicts.

static **_make_unique_relationship_id**(*item*)

Returns a unique name-based identifier for db relationships.

build_lookup_dictionaries(*db_map_data*)

Builds a name lookup dictionary for the given data.

Parameters

db_map_data (*dict*) – lists of model items keyed by DiffDatabaseMapping.

_make_relationship_on_the_fly(*item*, *db_map*)

Returns a database relationship item (id-based) from the given model parameter_value item (name-based).

Parameters

- **item** (*dict*) – the model parameter_value item
- **db_map** (*DiffDatabaseMapping*) – the database where the resulting item belongs

Returns

the db relationship item list: error log

Return type

dict

`spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item`

Tree items for parameter_value lists.

Module Contents

Classes

<i>DBItem</i>	An item representing a db.
<i>ListItem</i>	A list item.
<i>ValueItem</i>	Paints the item gray if it's the last.

```
class spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item.DBItem(*args,
                                                                              **kwargs)
```

Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildMixin`,
`spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.FetchMoreMixin`,
`spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.StandardDBItem`

An item representing a db.

property `item_type`

property `fetch_item_type`

empty_child()

_make_child(id_)

```
class spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item.ListItem(identifier=None,
                                                                              name=None)
```

Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.GrayIfLastMixin`,
`spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EditableMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.SortChildrenMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.BoldTextMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.FetchMoreMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem`

A list item.

Parameters

identifier (*int*, *optional*) – item's database id

property `item_type`

property `fetch_item_type`

_make_item_data()

_do_set_up()

Do stuff after the item has been inserted.

empty_child()

_make_child(id_)

accepts_item(*item*, *db_map*)

_children_sort_key(*child*)

data(*column*, *role*=*Qt.ItemDataRole.DisplayRole*)

Returns data for given column and role.

_make_item_to_add(*value*)

add_item_to_db(*db_item*)

update_item_in_db(*db_item*)

class `spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item.ValueItem`(*identifier*=*None*)

Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.GrayIfLastMixin`,
`spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EditableMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem`

Paints the item gray if it's the last.

Parameters

identifier (*int*, *optional*) – item's database id

property *item_type*

data(*column*, *role*=*Qt.ItemDataRole.DisplayRole*)

Returns data for given column and role.

list_index()

_make_item_to_add(*value*)

_make_item_to_update(*_column*, *value*)

add_item_to_db(*db_item*)

update_item_in_db(*db_item*)

`spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_model`

A tree model for `parameter_value` lists.

Module Contents

Classes

ParameterValueListModel

A model to display `parameter_value_list` data in a tree view.

class `spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_model.ParameterValueListModel`(*db_editor*,
db_mgr,
**db_map*)

Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_model_base.TreeModelBase`

A model to display `parameter_value_list` data in a tree view.

Parameters

- **db_editor** ([SpineDBEditor](#)) –
- **db_mgr** ([SpineDBManager](#)) –
- ***db_maps** – DiffDatabaseMapping instances

static `_make_db_item(db_map)`

static `_top_children()`

columnCount(*parent=QModelIndex()*)

Returns the number of columns under the given parent. Always 1.

index_name(*index*)

get_set_data_delayed(*index*)

Returns a function that ParameterValueEditor can call to set data for the given index at any later time, even if the model changes.

Parameters

index (*QModelIndex*) –

Returns

Callable

`spinetoolbox.spine_db_editor.mvcmodels.pivot_model`

Provides PivotModel.

Module Contents

Classes

PivotModel

class `spinetoolbox.spine_db_editor.mvcmodels.pivot_model.PivotModel`

property `rows`

property `columns`

reset_model(*data, top_left_headers=(), rows=(), columns=(), frozen=(), frozen_value=()*)

Resets the model.

clear_model()

update_model(*data*)

add_to_model(*data*)

Adds data to model.

Parameters

data (*dict*) – pivot model data

Returns

added row count and added column count

Return type

tuple

remove_from_model(*data*)

frozen_values(*data*)

Collects frozen values from data.

Parameters

data (*dict*) – pivot model data

Returns

frozen values

Return type

set of tuple

_check_pivot(*rows*, *columns*, *frozen*, *frozen_value*)

Checks if given pivot is valid.

Returns

error message or None if no error

Return type

str, NoneType

_index_key_getter(*indexes*)

Returns an itemgetter that always returns tuples from list of indexes

Parameters

indexes (*tuple*) –

Returns

an itemgetter

Return type

Callable

_get_unique_index_values(*indexes*)

Returns unique indexes that match the frozen condition.

Parameters

indexes (*tuple*) – indexes to match

Returns

unique indexes

Return type

list

set_pivot(*rows*, *columns*, *frozen*, *frozen_value*)

Sets pivot.

set_frozen_value(*value*)

Sets values for the frozen indexes.

Parameters

value (*list of str*) –

set_frozen(*frozen*)

Sets the frozen names without resetting the pivot.

Parameters

frozen (*Iterable of str*) –

get_pivoted_data(*row_mask, column_mask*)

Returns data for indexes in *row_mask* and *column_mask*.

Parameters

- **row_mask** (*list*) –
- **column_mask** (*list*) –

Returns

list(list)

row_key(*row*)

column_key(*column*)

`spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models`

Provides pivot table models for the Tabular View.

Module Contents

Classes

<i>TopLeftHeaderItem</i>	Base class for all 'top left pivot headers'.
<i>TopLeftObjectHeaderItem</i>	A top left header for <i>object_class</i> .
<i>TopLeftParameterHeaderItem</i>	A top left header for <i>parameter_definition</i> .
<i>TopLeftParameterIndexHeaderItem</i>	A top left header for <i>parameter index</i> .
<i>TopLeftAlternativeHeaderItem</i>	A top left header for <i>alternative</i> .
<i>TopLeftScenarioHeaderItem</i>	A top left header for <i>scenario</i> .
<i>TopLeftDatabaseHeaderItem</i>	A top left header for <i>database</i> .
<i>PivotTableModelBase</i>	
param db_editor	
<i>ParameterValuePivotTableModel</i>	A model for the pivot table in <i>parameter_value</i> input type.
<i>IndexExpansionPivotTableModel</i>	A model for the pivot table in <i>parameter index expansion</i> input type.
<i>RelationshipPivotTableModel</i>	A model for the pivot table in <i>relationship</i> input type.
<i>ScenarioAlternativePivotTableModel</i>	A model for the pivot table in <i>scenario alternative</i> input type.
<i>PivotTableSortFilterProxy</i>	Initialize class.

class `spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftHeaderItem`(*model*)

Base class for all 'top left pivot headers'. Represents a header located in the top left area of the pivot table.

Parameters

model (*PivotTableModelBase*) –

property `model`

property `db_mgr`

_get_header_data_from_db(*item_type*, *header_id*, *field_name*, *role*)

abstract **header_data**(*header_id*, *role*=*Qt.ItemDataRole.DisplayRole*)

Returns header data for given id.

Parameters

- **header_id** (*Any*) – header id
- **role** (*Qt.ItemDataRole*) – data role

Returns

data corresponding to role

Return type

Any

abstract **update_data**(*db_map_data*)

Updates database data.

Parameters

db_map_data (*dict*) – update data

Returns

True if data was successfully updated, False otherwise

Return type

bool

abstract **add_data**(*names*, *db_map*)

Adds more data to database.

Parameters

- **names** (*set of str*) – header names
- **db_map** (*DatabaseMapping*) – database to add the data to

Returns

True if data was added successfully, False otherwise

Return type

bool

class `spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftObjectHeaderItem`(*model*,
rank,
class_name,
class_id)

Bases: *TopLeftHeaderItem*

A top left header for object_class.

Parameters

model (*PivotTableModelBase*) –

property `header_type`

property `name`

header_data(*header_id*, *role=Qt.ItemDataRole.DisplayRole*)

See base class.

update_data(*db_map_data*)

See base class.

add_data(*names*, *db_map*)

See base class.

class `spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftParameterHeaderItem(model)`

Bases: [TopLeftHeaderItem](#)

A top left header for parameter_definition.

Parameters

model ([PivotTableModelBase](#)) –

property `header_type`

property `name`

header_data(*header_id*, *role=Qt.ItemDataRole.DisplayRole*)

See base class.

update_data(*db_map_data*)

See base class.

add_data(*names*, *db_map*)

See base class.

class `spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftParameterIndexHeaderItem(model)`

Bases: [TopLeftHeaderItem](#)

A top left header for parameter index.

Parameters

model ([PivotTableModelBase](#)) –

property `header_type`

property `name`

header_data(*header_id*, *role=Qt.ItemDataRole.DisplayRole*)

See base class.

update_data(*db_map_data*)

See base class.

add_data(*names*, *db_map*)

See base class.

class `spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftAlternativeHeaderItem(model)`

Bases: [TopLeftHeaderItem](#)

A top left header for alternative.

Parameters

model ([PivotTableModelBase](#)) –

property `header_type`

property name

header_data(*header_id*, *role*=*Qt.ItemDataRole.DisplayRole*)

See base class.

update_data(*db_map_data*)

See base class.

add_data(*names*, *db_map*)

See base class.

class `spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftScenarioHeaderItem(model)`

Bases: [*TopLeftHeaderItem*](#)

A top left header for scenario.

Parameters

model ([*PivotTableModelBase*](#)) –

property header_type

property name

header_data(*header_id*, *role*=*Qt.ItemDataRole.DisplayRole*)

See base class.

update_data(*db_map_data*)

See base class.

add_data(*names*, *db_map*)

See base class.

class `spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftDatabaseHeaderItem(model)`

Bases: [*TopLeftHeaderItem*](#)

A top left header for database.

Parameters

model ([*PivotTableModelBase*](#)) –

property header_type

property name

header_data(*header_id*, *role*=*Qt.ItemDataRole.DisplayRole*)

See base class.

update_data(*db_map_data*)

See base class.

add_data(*names*, *db_map*)

See base class.

set_data(*codename*)

Sets database mapping's codename.

Parameters

codename (*str*) – database codename

Returns

True if codename was acceptable, False otherwise

Return type

bool

take_suggested_db_map()

Suggests database mapping resetting the suggestion afterwards.

Returns

database mapping

Return type

DatabaseMapping

suggest_db_map_codename()

Suggests a database mapping codename.

Returns

codename

Return type

str

class spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.**PivotTableModelBase**(*db_editor*)

Bases: PySide6.QtCore.QAbstractTableModel

Parameters

db_editor ([SpineDBEditor](#)) –

property db_maps

abstract property item_type

Returns the item type.

property plot_x_column

Returns the index of the column designated as Y values for plotting or None.

_CHUNK_SIZE = 1000

model_data_changed

frozen_values_added

frozen_values_removed

reset_fetch_parents()

abstract _fetch_parents()

Yields fetch parents for this model.

Yields

FetchParent

canFetchMore(_)

fetchMore(_)

_reset_data_count()

_collect_more_data()

_collect_more_rows()

_collect_more_columns()

abstract call_reset_model(*pivot=None*)

Parameters

pivot (*tuple*, *optional*) – list of rows, list of columns, list of frozen indexes, frozen value

abstract static make_delegate(*parent*)

reset_model(*data*, *index_ids*, *rows=()*, *columns=()*, *frozen=()*, *frozen_value=()*)

clear_model()

update_model(*data*)

Update model with new data, but doesn't grow the model.

Parameters

data (*dict*) –

add_to_model(*db_map_data*)

remove_from_model(*data*)

_emit_all_data_changed()

set_pivot(*rows*, *columns*, *frozen*, *frozen_value*)

set_frozen(*frozen*)

Sets the order of frozen headers without changing model data.

Parameters

frozen (*list of str*) – new frozen

set_frozen_value(*frozen_value*)

Sets frozen value resetting the model.

Parameters

frozen_value (*tuple*) – frozen value

Returns

True if value was set, False otherwise

Return type

bool

set_plot_x_column(*column*, *is_x*)

Sets or clears the X flag on a column

x_value(*index*)

Returns x value for given model index.

Parameters

index (*QModelIndex*) – model index

Returns

x value

Return type

Any

x_parameter_name()

Returns x column's parameter name.

Returns

parameter name

Return type

str

headerRowCount()

Returns number of rows occupied by header.

headerColumnCount()

Returns number of columns occupied by header.

dataRowCount()

Returns number of rows that contain actual data.

dataColumnCount()

Returns number of columns that contain actual data.

emptyRowCount()**emptyColumnCount()****rowCount**(*parent=QModelIndex()*)

Number of rows in table, number of header rows + datarows + 1 empty row

columnCount(*parent=QModelIndex()*)

Number of columns in table, number of header columns + datacolumns + 1 empty columns

flags(*index*)

Roles for data

top_left_indexes()

Returns indexes in the top left area.

Returns

list(QModelIndex): top indexes (horizontal headers, associated to rows) list(QModelIndex): left indexes (vertical headers, associated to columns)

index_within_top_left(*index*)**index_in_top**(*index*)**index_in_left**(*index*)**index_in_top_left**(*index*)

Returns whether the given index is in top left corner, where pivot names are displayed.

index_in_column_headers(*index*)

Returns whether the given index is in column headers (horizontal) area.

index_in_row_headers(*index*)

Returns whether the given index is in row headers (vertical) area.

index_in_headers(*index*)**index_in_empty_column_headers**(*index*)

Returns whether the given index is in empty column headers (vertical) area.

index_in_empty_row_headers(*index*)

Returns whether the given index is in empty row headers (vertical) area.

index_in_data(*index*)

Returns whether the given index is in data area.

column_is_index_column(*column*)

Returns True if column is the column containing expanded parameter_value indexes.

headerData(*section, orientation, role=Qt.ItemDataRole.DisplayRole*)

map_to_pivot(*index*)

Returns a tuple of row and column in the pivot model that corresponds to the given model index.

Parameters

index (*QModelIndex*) –

Returns

row int: column

Return type

int

top_left_id(*index*)

Returns the id of the top left header corresponding to the given header index.

Parameters

index (*QModelIndex*) –

Returns

int, NoneType

_header_id(*index*)

Returns the id of the given row or column header index.

Parameters

index (*QModelIndex*) –

Returns

tuple or DatabaseMapping or NoneType

_header_ids(*row, column*)

Returns the ids for the headers at given row *and* column.

Parameters

- **row** (*int*) –
- **column** (*int*) –

Returns

tuple(int)

header_name(*index*)

Returns the name corresponding to the given header index. Used by PivotTableView.

Parameters

index (*QModelIndex*) –

Returns

str

```

_color_data(index)
_text_alignment_data(index)
_header_data(index, role=Qt.ItemDataRole.DisplayRole)
_header_name(top_left_id, header_id)
abstract _data(index, role)
data(index, role=Qt.ItemDataRole.DisplayRole)
setData(index, value, role=Qt.ItemDataRole.EditRole)
batch_set_data(indexes, values)
_batch_set_inner_data(inner_data)
abstract _do_batch_set_inner_data(row_map, column_map, data, values)
_batch_set_header_data(header_data)

```

Sets header data for multiple indexes at once.

Parameters

header_data (*list of tuple*) – mapping from index to data

Returns

True if data was set successfully, False otherwise

Return type

bool

```
_batch_set_empty_header_data(header_data, get_top_left_id)
```

```
class spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ParameterValuePivotTableModel(parent)
```

Bases: [*PivotTableModelBase*](#)

A model for the pivot table in parameter_value input type.

Parameters

parent ([*SpineDBEditor*](#)) –

property item_type

Returns the item type.

```

_handle_entities_added(db_map_data)
_handle_entities_removed(db_map_data)
_handle_parameter_definitions_added(db_map_data)
_handle_parameter_definitions_removed(db_map_data)
_handle_parameter_values_added(db_map_data)
_handle_parameter_values_removed(db_map_data)
_handle_alternatives_added(db_map_data)
_handle_alternatives_removed(db_map_data)

```

`_load_empty_parameter_value_data(*args, **kwargs)`

`_load_full_parameter_value_data(*args, **kwargs)`

`_fetch_parents()`

Yields fetch parents for this model.

Yields

FetchParent

`reset_fetch_parents()`

`db_map_object_ids(index)`

Returns db_map and object ids for given index. Used by PivotTableView.

Returns

DatabaseMapping, list

`_db_map_object_ids(header_ids)`

`all_header_names(index)`

Returns the object, parameter, alternative, and db names corresponding to the given data index.

Parameters

`index` (*QModelIndex*) –

Returns

object names str: parameter name str: alternative name str: db name

Return type

list(str)

`index_name(index)`

Returns a string that concatenates the object and parameter names corresponding to the given data index. Used by plotting and ParameterValueEditor.

Parameters

`index` (*QModelIndex*) –

Returns

str

`column_name(column)`

Returns a string that concatenates the object and parameter names corresponding to the given column. Used by plotting.

Parameters

`column` (*int*) –

Returns

str

`call_reset_model(pivot=None)`

See base class.

`static make_delegate(parent)`

`_default_pivot(data)`

`_data(index, role)`

`_do_batch_set_inner_data(row_map, column_map, data, values)`

`_object_parameter_value_to_add(db_map, header_ids, value_and_type)`

`_relationship_parameter_value_to_add(db_map, header_ids, value_and_type, rel_id_lookup)`

`_make_parameter_value_to_add()`

`static _parameter_value_to_update(id_, header_ids, value_and_type)`

`_batch_set_parameter_value_data(row_map, column_map, data, values)`

Sets parameter values in batch.

`_add_parameter_values(db_map_data)`

`_update_parameter_values(db_map_data)`

`get_set_data_delayed(index)`

Returns a function that ParameterValueEditor can call to set data for the given index at any later time, even if the model changes.

Parameters

index (*QModelIndex*) –

Returns

function

class `spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.IndexExpansionPivotTableModel(parent)`

Bases: [*ParameterValuePivotTableModel*](#)

A model for the pivot table in parameter index expansion input type.

Parameters

parent (*SpineDBEditor*) –

`call_reset_model(pivot=None)`

See base class.

`flags(index)`

Roles for data

`column_is_index_column(column)`

Returns True if column is the column containing expanded parameter_value indexes.

`_load_empty_parameter_value_data(*args, **kwargs)`

`_load_full_parameter_value_data(*args, **kwargs)`

`_data(index, role)`

`static _parameter_value_to_update(id_, header_ids, value_and_type)`

`_update_parameter_values(db_map_data)`

class `spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.RelationshipPivotTableModel(parent)`

Bases: [*PivotTableModelBase*](#)

A model for the pivot table in relationship input type.

Parameters

parent (*SpineDBEditor*) –

property item_type

Returns the item type.

_handle_relationships_added(*db_map_data*)

_handle_relationships_removed(*db_map_data*)

_load_empty_relationship_data(*db_map_data*)

_handle_objects_added(*db_map_data*)

_handle_objects_removed(*db_map_data*)

_fetch_parents()

Yields fetch parents for this model.

Yields

FetchParent

call_reset_model(*pivot=None*)

See base class.

static make_delegate(*parent*)

_default_pivot(*data*)

_data(*index, role*)

_do_batch_set_inner_data(*row_map, column_map, data, values*)

_batch_set_relationship_data(*row_map, column_map, data, values*)

class spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ScenarioAlternativePivotTableModel(*parent*)

Bases: [PivotTableModelBase](#)

A model for the pivot table in scenario alternative input type.

Parameters

parent ([SpineDBEditor](#)) –

property item_type

Returns the item type.

_handle_scenarios_added(*db_map_data*)

_handle_scenarios_removed(*db_map_data*)

_handle_alternatives_added(*db_map_data*)

_handle_alternatives_removed(*db_map_data*)

_handle_scenario_alternatives_changed(*db_map_data*)

_fetch_parents()

Yields fetch parents for this model.

Yields

FetchParent

call_reset_model(*pivot=None*)

See base class.

static `make_delegate(parent)`

`_default_pivot(data)`

`_data(index, role)`

`_do_batch_set_inner_data(row_map, column_map, data, values)`

`_batch_set_scenario_alternative_data(row_map, column_map, data, values)`

class `spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableSortFilterProxy`(*parent=None*)

Bases: `PySide6.QtCore.QSortFilterProxyModel`

Initialize class.

model_data_changed

setSourceModel(*model*)

set_filter(*identifier, filter_value*)

Sets filter for a given index (object_class) name.

Parameters

- **identifier** (*int*) – index identifier
- **filter_value** (*set, None*) – A set of accepted values, or None if no filter (all pass)

clear_filter()

accept_index(*index, index_ids*)

filterAcceptsRow(*source_row, source_parent*)

Returns true if the item in the row indicated by the given *source_row* and *source_parent* should be included in the model; otherwise returns false.

filterAcceptsColumn(*source_column, source_parent*)

Returns true if the item in the column indicated by the given *source_column* and *source_parent* should be included in the model; otherwise returns false.

batch_set_data(*indexes, values*)

spinetoolbox.spine_db_editor.mvcmodels.scenario_item

Classes to represent items in scenario tree.

Module Contents

Classes

<i>ScenarioDBItem</i>	A root item representing a db.
<i>ScenarioItem</i>	A scenario leaf item.
<i>ScenarioAlternativeItem</i>	A scenario alternative leaf item.

Attributes

_SCENARIO_ICON

`spinetoolbox.spine_db_editor.mvcmodels.scenario_item._SCENARIO_ICON = '\uf008'`

class `spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioDBItem(*args, **kwargs)`
Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildMixin`,
`spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.FetchMoreMixin`,
`spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.StandardDBItem`

A root item representing a db.

property `item_type`

property `fetch_item_type`

empty_child()

_make_child(id_)

class `spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioItem(*args, **kwargs)`
Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.GrayIfLastMixin`,
`spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EditableMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.FetchMoreMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.BoldTextMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem`

A scenario leaf item.

property `item_type`

property `fetch_item_type`

property `icon_code`

property `tool_tip`

property `alternative_id_list`

_do_set_up()

Doesn't add children to the last row.

add_item_to_db(db_item)

update_item_in_db(db_item)

handle_updated_in_db()

flags(column)

Makes items editable.

update_alternative_id_list()

handle_items_added(*_db_map_data*)

Inserts items at right positions. Items with `commit_id` are kept sorted. Items without a `commit_id` are put at the end.

Parameters

db_map_data (*dict*) – mapping `db_map` to list of dict corresponding to db items

handle_items_removed(*_db_map_data*)

handle_items_updated(*_db_map_data*)

empty_child()

See base class.

_make_child(*id_*)

Not needed - we don't quite add children here, but rather update them in `update_alternative_id_list`.

class `spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioAlternativeItem`(*identifier=None*)

Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.GrayIfLastMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EditableMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem`

A scenario alternative leaf item.

Parameters

identifier (*int*, *optional*) – item's database id

property `item_type`

property `tool_tip`

property `item_data`

property `alternative_id`

_make_item_data()

abstract `add_item_to_db`(*db_item*)

abstract `update_item_in_db`(*db_item*)

flags(*column*)

Makes items editable.

set_data(*column*, *value*, *role=Qt.ItemDataRole.EditRole*)

Sets data for this item.

Parameters

- **column** (*int*) – column index
- **value** (*object*) – a new value
- **role** (*int*) – role of the new value

Returns

True if data was set successfully, False otherwise

Return type

bool

`spinetoolbox.spine_db_editor.mvcmodels.scenario_model`

Contains scenario tree model.

Module Contents

Classes

ScenarioModel

A model to display scenarios in a tree view.

```
class spinetoolbox.spine_db_editor.mvcmodels.scenario_model.ScenarioModel(db_editor,
                                                                           db_mgr,
                                                                           *db_maps)
```

Bases: *spinetoolbox.spine_db_editor.mvcmodels.tree_model_base.TreeModelBase*

A model to display scenarios in a tree view.

Parameters

- **db_editor** (*SpineDBEditor*) –
- **db_mgr** (*SpineDBManager*) –
- ***db_maps** – DiffDatabaseMapping instances

static `_make_db_item(db_map)`

static `_top_children()`

supportedDropActions()

mimeData(*indexes*)

Stores selected indexes into MIME data.

If indexes contains scenario indexes, only those indexes will be kept. Otherwise, only scenario alternative indexes are kept.

The MIME data contains distinct data: - Text representation of the selection - A pickled dict mapping db identifier to list of alternative ids - A pickled dict mapping db identifier to list of scenario ids

Parameters

indexes (*Sequence of QModelIndex*) – selected indexes

Returns

MIME data or None if selection was bad

Return type

QMimeData

canDropMimeData(*data, drop_action, row, column, parent*)

dropMimeData(*data, drop_action, row, column, parent*)

paste_alternative_mime_data(*mime_data, row, scenario_item*)

Adds alternatives from MIME data to the model.

Parameters

- **mime_data** (*QMimeData*) – mime data that must contain ALTERNATIVE_DATA format

- **row** (*int*) – where to paste within scenario item, -1 lets the model choose
- **scenario_item** (*ScenarioItem*) – parent item

paste_scenario_mime_data(*mime_data*, *db_item*)

Adds scenarios and their alternatives from MIME data to the model.

Parameters

- **mime_data** (*QMimeData*) – mime data that must contain ALTERNATIVE_DATA format
- **db_item** (*ScenarioDBItem*) – parent item

duplicate_scenario(*scenario_item*)

Duplicates scenario within database.

Parameters

- **scenario_item** (*ScenarioItem*) – scenario item to duplicate

spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models

Single models for parameter definitions and values (as ‘for a single entity’).

Module Contents

Classes

<i>HalfSortedTableModel</i>	Table model for outlining simple tabular data.
<i>SingleParameterModel</i>	A parameter model for a single entity_class to go in a CompoundParameterModel.
<i>SingleObjectParameterMixin</i>	Associates a parameter model with a single object_class.
<i>SingleRelationshipParameterMixin</i>	Associates a parameter model with a single relationship_class.
<i>SingleParameterDefinitionMixin</i>	A parameter_definition model for a single entity_class.
<i>SingleParameterValueMixin</i>	A parameter_value model for a single entity_class.
<i>SingleObjectParameterDefinitionModel</i>	An object parameter_definition model for a single object_class.
<i>SingleRelationshipParameterDefinitionModel</i>	A relationship parameter_definition model for a single relationship_class.
<i>SingleObjectParameterValueModel</i>	An object parameter_value model for a single object_class.
<i>SingleRelationshipParameterValueModel</i>	A relationship parameter_value model for a single relationship_class.

class spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.**HalfSortedTableModel**(*parent=None*,
header=None,
lazy=True)

Bases: *spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel*

Table model for outlining simple tabular data.

Parameters

- **parent** (*QObject*, *optional*) – the parent object

- **header** (*list of str*) – header labels
- **lazy** (*boolean*) – if True, fetches data lazily

reset_model(*main_data=None*)

Reset model.

add_rows(*data*)

_sort_key(*element*)

```
class spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel(header,
                                                                                          db_mgr,
                                                                                          db_map,
                                                                                          en-
                                                                                          tity_class_id,
                                                                                          com-
                                                                                          mit-
                                                                                          ted,
                                                                                          lazy=False)
```

Bases: [*HalfSortedTableModel*](#)

A parameter model for a single entity_class to go in a CompoundParameterModel. Provides methods to associate the model to an entity_class as well as to filter entities within the class.

Init class.

Parameters

header (*list*) – list of field names for the header

abstract property item_type

The item type, either 'parameter_value' or 'parameter_definition', required by the data method.

abstract property entity_class_type

The entity_class type, either 'object_class' or 'relationship_class'.

property entity_class_name_field

property entity_class_name

property entity_class_id_key

property value_field

property fixed_fields

property group_fields

property parameter_definition_id_key

property can_be_filtered

__lt__(*other*)

item_id(*row*)

Returns parameter id for row.

Parameters

row (*int*) – row index

Returns

parameter id

Return type

int

item_ids()

Returns model's parameter ids.

Returns

ids

Return type

set of int

db_item(index)**_db_item(row)****db_item_from_id(id_)****db_items()****flags(index)**

Make fixed indexes non-editable.

get_field_item_data(field)

Returns item data for given field.

Parameters**field** (*str*) – A field from the header**Returns***str*, *str***get_id_key(field)****get_field_item(field, db_item)**

Returns a db item corresponding to the given field from the table header, or an empty dict if the field doesn't contain db items.

data(index, role=*Qt.ItemDataRole.DisplayRole*)Gets the id and database for the row, and reads data from the db manager using the *item_type* property. Paint the *object_class* icon next to the name. Also paint background of fixed indexes gray and apply custom format to JSON fields.**batch_set_data(indexes, data)**

Sets data for indexes in batch. Sets data directly in database using db mngr. If successful, updated data will be automatically seen by the data method.

abstract update_items_in_db(items)Update items in db. Required by *batch_set_data***_filter_accepts_row(row)****filter_accepts_item(item)****set_auto_filter(field, values)**

_auto_filter_accepts_item(*item*)

Returns the result of the auto filter.

accepted_rows()

Yields accepted rows, for convenience.

_get_field_item(*field*, *id_*)

Returns a item from the `db_mgr.get_item` depending on the field. If a field doesn't correspond to a item in the database then an empty dict is returned.

class

`spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleObjectParameterMixin`

Associates a parameter model with a single `object_class`.

property `entity_class_type`

class `spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.`

SingleRelationshipParameterMixin

Associates a parameter model with a single `relationship_class`.

property `entity_class_type`

class `spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterDefinitionMixin(*args, **kwargs)`

Bases: `spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInParameterNameMixin`, `spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInValueListIdMixin`

A `parameter_definition` model for a single `entity_class`.

Initializes lookup dicts.

property `item_type`

_sort_key(*element*)

update_items_in_db(*items*)

Update items in db.

Parameters

items (*list*) – dictionary-items

class `spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterValueMixin(*args, **kwargs)`

Bases: `spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInAlternativeIdMixin`, `spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.ImposeEntityClassIdMixin`, `spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInParameterDefinitionIdsMixin`, `spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInEntityIdsMixin`

A `parameter_value` model for a single `entity_class`.

Initializes lookup dicts.

property `item_type`

abstract property `entity_type`

Either 'object' or 'relationship'.


```

property entity_id_key
property entity_name_key
property entity_name_key_in_cache
_filter_db_map_class_entity_ids
_filter_alternative_ids
_filter_entity_ids
_sort_key(element)
set_filter_entity_ids(db_map_class_entity_ids)
set_filter_alternative_ids(db_map_alternative_ids)
filter_accepts_item(item)
    Reimplemented to also account for the entity and alternative filter.
_entity_filter_accepts_item(item)
    Returns the result of the entity filter.
_alternative_filter_accepts_item(item)
    Returns the result of the alternative filter.
update_items_in_db(items)
    Update items in db.

```

Parameters

items (*list*) – dictionary-items

```
class spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleObjectParameterDefinitionModel
```

Bases: *SingleObjectParameterMixin*, *SingleParameterDefinitionMixin*, *SingleParameterModel*

An object parameter_definition model for a single object_class.

Initializes lookup dicts.

```
class spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleRelationshipParameterDefinitionModel
```

Bases: *SingleRelationshipParameterMixin*, *SingleParameterDefinitionMixin*, *SingleParameterModel*

A relationship parameter_definition model for a single relationship_class.

Initializes lookup dicts.

```
class spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleObjectParameterValueModel(*args, **kwargs)
```

Bases: *SingleObjectParameterMixin*, *SingleParameterValueMixin*, *SingleParameterModel*

An object parameter_value model for a single object_class.

Initializes lookup dicts.

property entity_type

Either 'object' or 'relationship'.

class `spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleRelationshipParameterValueModel`

Bases: `SingleRelationshipParameterMixin`, `spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.MakeRelationshipOnTheFlyMixin`, `SingleParameterValueMixin`, `SingleParameterModel`

A relationship parameter_value model for a single relationship_class.

Initializes lookup dicts.

property `entity_type`

Either 'object' or 'relationship'.

update_items_in_db(*items*)

Update items in db.

Parameters

items (*list*) – dictionary-items

spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item

Classes to represent tool and feature items in a tree.

Module Contents

Classes

<code>FeatureRootItem</code>	A feature root item.
<code>ToolRootItem</code>	A tool root item.
<code>FeatureLeafItem</code>	A feature leaf item.
<code>ToolLeafItem</code>	A tool leaf item.
<code>ToolFeatureRootItem</code>	A tool_feature root item.
<code>ToolFeatureLeafItem</code>	A tool feature leaf item.
<code>ToolFeatureRequiredItem</code>	A tool feature required item.
<code>ToolFeatureMethodRootItem</code>	A tool_feature_method root item.
<code>ToolFeatureMethodLeafItem</code>	A tool_feature_method leaf item.

Attributes

<code>_FEATURE_ICON</code>
<code>_TOOL_ICON</code>
<code>_METHOD_ICON</code>

`spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item._FEATURE_ICON = '\uf5bc'`

`spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item._TOOL_ICON = '\uf6e3'`

```
spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item._METHOD_ICON = '\uf1de'
```

class spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.**FeatureRootItem**(*args,
**kwargs)

Bases: *spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildRootItem*

A feature root item.

property item_type

property display_data

property icon_code

empty_child()

_make_child(id_)

class spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.**ToolRootItem**(*args,
**kwargs)

Bases: *spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildRootItem*

A tool root item.

property item_type

property display_data

property icon_code

empty_child()

_make_child(id_)

class spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.**FeatureLeafItem**(identifier=None)

Bases: *spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.GrayIfLastMixin, spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EditableMixin, spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem*

A feature leaf item.

Parameters

identifier(int, optional) – item’s database id

property item_type

property item_data

property tool_tip

_make_item_data()

add_item_to_db(db_item)

update_item_in_db(db_item)

flags(column)

Makes items editable.

_make_item_to_add(value)

`_make_item_to_update(column, value)`

`_get_ids_from_feat_name(feature_name)`

class `spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolLeafItem(identifier=None)`

Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.GrayIfLastMixin`,
`spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EditableMixin`, `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem`

A tool leaf item.

Parameters

identifier (*int*, *optional*) – item’s database id

property `item_type`

`add_item_to_db(db_item)`

`update_item_in_db(db_item)`

`_do_set_up()`

Do stuff after the item has been inserted.

class `spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureRootItem(*args, **kwargs)`

Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildRootItem`

A tool_feature root item.

property `item_type`

property `display_data`

property `tool_tip`

property `icon_code`

property `feature_id_list`

flags (*column*)

Enables the item and makes it selectable.

`empty_child()`

`_make_child(id_)`

`accepts_item(item, db_map)`

class `spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureLeafItem(identifier=None)`

Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.GrayIfLastMixin`,
`spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem`

A tool feature leaf item.

Parameters

identifier (*int*, *optional*) – item’s database id

property `item_type`

property `item_data`

_do_set_up()

Do stuff after the item has been inserted.

_make_item_to_add(*value*)

add_item_to_db(*db_item*)

update_item_in_db(*db_item*)

flags(*column*)

Enables the item and makes it selectable.

class spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.**ToolFeatureRequiredItem**(*model=None*)

Bases: *spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeItem*

A tool feature required item.

Parameters

model (*MinimalTreeModel*, *optional*) – The model where the item belongs.

property item_type

flags(*column*)

Enables the item and makes it selectable.

data(*column*, *role=Qt.ItemDataRole.DisplayRole*)

Returns data for given column and role.

set_data(*column*, *value*, *role=Qt.ItemDataRole.EditRole*)

Sets data for this item.

Parameters

- **column** (*int*) – column index
- **value** (*object*) – a new value
- **role** (*int*) – role of the new value

Returns

True if data was set successfully, False otherwise

Return type

bool

has_children()

Returns whether this item has or could have children.

class spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.**ToolFeatureMethodRootItem**(**args*,
***kwargs*)

Bases: *spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildRootItem*

A tool_feature_method root item.

property item_type

property display_data

property icon_code

empty_child()

`_make_child(id_)`

`accepts_item(item, db_map)`

class `spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureMethodLeafItem`(*identifier=None*)

Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.GrayIfLastMixin`,
`spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem`

A tool_feature_method leaf item.

Parameters

identifier (*int*, *optional*) – item’s database id

property **item_type**

property **tool_feature_item**

property **item_data**

`_make_item_data()`

flags (*column*)

Enables the item and makes it selectable.

`_make_item_to_add(value)`

`_make_item_to_update(column, value)`

`_get_method_index(parameter_value_list_id, method)`

`add_item_to_db(db_item)`

`update_item_in_db(db_item)`

`spinetoolbox.spine_db_editor.mvcmodels.tool_feature_model`

Models to represent tools and features in a tree.

Module Contents

Classes

ToolFeatureModel

A model to display tools and features in a tree view.

class `spinetoolbox.spine_db_editor.mvcmodels.tool_feature_model.ToolFeatureModel`(*parent*,
db_mgr,
**db_maps*)

Bases: `spinetoolbox.spine_db_editor.mvcmodels.tree_model_base.TreeModelBase`

A model to display tools and features in a tree view.

Parameters

- **parent** (`SpineDBEditor`) –
- **db_mgr** (`SpineDBManager`) –

- **db_maps** (*iter*) – DiffDatabaseMapping instances

Initialize class

```
static _make_db_item(db_map)

static _top_children()

static make_feature_name(entity_class_name, parameter_definition_name)

_begin_set_features(db_map)

get_all_feature_names(db_map)

get_feature_data(db_map, feature_name)

_begin_set_feature_method(db_map, parameter_value_list_id)

get_all_feature_methods(db_map, parameter_value_list_id)

get_method_index(db_map, parameter_value_list_id, method)

supportedDropActions()

mimeData(indexes)
    Builds a dict mapping db name to item type to a list of ids.

Returns
    QMimeData

canDropMimeData(data, drop_action, row, column, parent)

dropMimeData(data, drop_action, row, column, parent)
```

spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility

A tree model for parameter_value lists.

Module Contents

Classes

<i>StandardTreeItem</i>	A tree item that fetches their children as they are inserted.
<i>EditableMixin</i>	
<i>GrayIfLastMixin</i>	Paints the item gray if it's the last.
<i>BoldTextMixin</i>	Bolds text.
<i>EmptyChildMixin</i>	Guarantees there's always an empty child.
<i>SortChildrenMixin</i>	
<i>FetchMoreMixin</i>	
<i>StandardDBItem</i>	An item representing a db.
<i>RootItem</i>	A root item.
<i>EmptyChildRootItem</i>	Guarantees there's always an empty child.
<i>LeafItem</i>	A tree item that fetches their children as they are inserted.

class `spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeItem(model=None)`

Bases: `spinetoolbox.mvcmodels.minimal_tree_model.TreeItem`

A tree item that fetches their children as they are inserted.

Parameters

model (`MinimalTreeModel`, *optional*) – The model where the item belongs.

property `item_type`

property `db_mgr`

property `display_data`

property `icon_code`

property `tool_tip`

property `display_icon`

property `non_empty_children`

property `children_ids`

data(*column*, *role*=`Qt.ItemDataRole.DisplayRole`)

Returns data for given column and role.

set_data(*column*, *value*, *role*=`Qt.ItemDataRole.DisplayRole`)

Sets data for this item.

Parameters

- **column** (*int*) – column index
- **value** (*object*) – a new value
- **role** (*int*) – role of the new value

Returns

True if data was set successfully, False otherwise

Return type

bool

class spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.**EditableMixin**

flags(*column*)

Makes items editable.

class spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.**GrayIfLastMixin**

Paints the item gray if it's the last.

data(*column*, *role*=*Qt.ItemDataRole.DisplayRole*)

class spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.**BoldTextMixin**

Bolds text.

data(*column*, *role*=*Qt.ItemDataRole.DisplayRole*)

class spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.**EmptyChildMixin**

Guarantees there's always an empty child.

property **non_empty_children**

abstract **empty_child()**

_do_set_up()

class spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.**SortChildrenMixin**

_children_sort_key(*child*)

insert_children_sorted(*children*)

_resort()

class spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.**FetchMoreMixin**(*args,
**kwargs)

property **fetch_item_type**

tear_down()

_fetch_parents()

can_fetch_more()

fetch_more()

abstract **_make_child**(*id_*)

accepts_item(*item*, *db_map*)

handle_items_added(*db_map_data*)

Inserts items at right positions. Items with commit_id are kept sorted. Items without a commit_id are put at the end.

Parameters

db_map_data (*dict*) – mapping db_map to list of dict corresponding to db items

handle_items_removed(*db_map_data*)

handle_items_updated(*db_map_data*)

class spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.**StandardDBItem**(*db_map*)

Bases: [SortChildrenMixin](#), [StandardTreeItem](#)

An item representing a db.

Init class.

Args

db_mgr (SpineDBManager) db_map (DiffDatabaseMapping)

property item_type

data(*column*, *role=Qt.ItemDataRole.DisplayRole*)

Shows Spine icon for fun.

class spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.**RootItem**(*args, **kwargs)

Bases: [SortChildrenMixin](#), [BoldTextMixin](#), [FetchMoreMixin](#), [StandardTreeItem](#)

A root item.

abstract property item_type

property db_map

class spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.**EmptyChildRootItem**(*args, **kwargs)

Bases: [EmptyChildMixin](#), [RootItem](#)

Guarantees there's always an empty child.

abstract empty_child()

class spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.**LeafItem**(*identifier=None*)

Bases: [StandardTreeItem](#)

A tree item that fetches their children as they are inserted.

Parameters

identifier(*int*, *optional*) – item's database id

abstract property item_type

property db_map

property id

property item_data

property name

_make_item_data()

abstract add_item_to_db(*db_item*)

abstract update_item_in_db(*db_item*)

header_data(*column*)

data(*column*, *role*=*Qt.ItemDataRole.DisplayRole*)

Returns data for given column and role.

set_data(*column*, *value*, *role*=*Qt.ItemDataRole.EditRole*)

Sets data for this item.

Parameters

- **column** (*int*) – column index
- **value** (*object*) – a new value
- **role** (*int*) – role of the new value

Returns

True if data was set successfully, False otherwise

Return type

bool

_make_item_to_add(*value*)

_make_item_to_update(*column*, *value*)

handle_updated_in_db()

can_fetch_more()

Returns whether this item can fetch more.

spinetoolbox.spine_db_editor.mvcmodels.tree_model_base

Models to represent things in a tree.

Module Contents

Classes

TreeModelBase

A base model to display items in a tree view.

class spinetoolbox.spine_db_editor.mvcmodels.tree_model_base.**TreeModelBase**(*db_editor*,
db_mgr,
**db_maps*)

Bases: *spinetoolbox.mvcmodels.minimal_tree_model.MinimalTreeModel*

A base model to display items in a tree view.

Parameters

- **db_editor** (*SpineDBEditor*) –
- **db_mgr** (*SpineDBManager*) –
- ***db_maps** – DiffDatabaseMapping instances

columnCount(*parent=QModelIndex()*)

Returns the number of columns under the given parent. Always 2.

Returns

column count

Return type

int

headerData(*section, orientation, role=Qt.ItemDataRole.DisplayRole*)

build_tree()

Builds tree.

abstract static _make_db_item(*db_map*)

abstract static _top_children()

static db_item(*item*)

db_row(*item*)

spinetoolbox.spine_db_editor.mvcmodels.utils

General helper functions and classes for DB editor's models.

Module Contents

Functions

<i>two_column_as_csv</i> (indexes)	Writes data in given indexes into a CSV table.
------------------------------------	--

spinetoolbox.spine_db_editor.mvcmodels.utils.two_column_as_csv(*indexes*)

Writes data in given indexes into a CSV table.

Expects the source table to have two columns.

Parameters

indexes (*Sequence of QModelIndex*) – model indexes

Returns

data as CSV table

Return type

str

`spinetoolbox.spine_db_editor.ui`

Automatically generated UI modules for Spine db editor.

Submodules

`spinetoolbox.spine_db_editor.ui.scenario_generator`

Module Contents

Classes

Ui_Form

```
class spinetoolbox.spine_db_editor.ui.scenario_generator.Ui_Form
    Bases: object
    setupUi (Form)
    retranslateUi (Form)
```

`spinetoolbox.spine_db_editor.ui.select_databases`

Module Contents

Classes

Ui_Form

```
class spinetoolbox.spine_db_editor.ui.select_databases.Ui_Form
    Bases: object
    setupUi (Form)
    retranslateUi (Form)
```

`spinetoolbox.spine_db_editor.ui.spine_db_editor_window`

Module Contents

Classes

Ui_MainWindow

class `spinetoolbox.spine_db_editor.ui.spine_db_editor_window.Ui_MainWindow`

Bases: `object`

setupUi (*MainWindow*)

retranslateUi (*MainWindow*)

`spinetoolbox.spine_db_editor.widgets`

Interface logic for Spine db editor.

Submodules

`spinetoolbox.spine_db_editor.widgets.add_items_dialogs`

Classes for custom QDialogs to add items to databases.

Module Contents

Classes

<i>AddReadyRelationshipsDialog</i>	A dialog to let the user add new 'ready' relationships.
<i>AddItemsDialog</i>	A dialog to query user's preferences for new db items.
<i>AddObjectClassesDialog</i>	A dialog to query user's preferences for new object classes.
<i>AddObjectsDialog</i>	A dialog to query user's preferences for new objects.
<i>AddRelationshipClassesDialog</i>	A dialog to query user's preferences for new relationship classes.
<i>AddOrManageRelationshipsDialog</i>	A dialog to query user's preferences for new relationships.
<i>AddRelationshipsDialog</i>	A dialog to query user's preferences for new relationships.
<i>ManageRelationshipsDialog</i>	A dialog to query user's preferences for managing relationships.
<i>ObjectGroupDialogBase</i>	<p>param parent data store widget</p>
<i>AddObjectGroupDialog</i>	<p>param parent data store widget</p>
<i>ManageMembersDialog</i>	<p>param parent data store widget</p>

```
class spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddReadyRelationshipsDialog(parent,
                                                                                       re-
                                                                                       la-
                                                                                       tion-
                                                                                       ships_class,
                                                                                       re-
                                                                                       la-
                                                                                       tion-
                                                                                       ships,
                                                                                       db_mgr,
                                                                                       *db_maps)
```

Bases: `spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ManageItemsDialogBase`

A dialog to let the user add new ‘ready’ relationships.

Parameters

- **parent** (`SpineDBEditor`) –
- **relationships_class** (`dict`) –
- **relationships** (`list(list(str))`) –
- **db_mgr** (`SpineDBManager`) –
- ***db_maps** – DiffDatabaseMapping instances

make_table_view()

populate_table_view()

connect_signals()

Connect signals to slots.

_handle_table_view_cell_clicked(*row*, *column*)

_handle_table_view_current_changed(*current*, *_previous*)

accept()

```
class spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddItemDialog(parent, db_mgr,
                                                                                       *db_maps)
```

Bases: `spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ManageItemsDialog`

A dialog to query user’s preferences for new db items.

Parameters

- **parent** (`SpineDBEditor`) –
- **db_mgr** (`SpineDBManager`) –
- ***db_maps** – DiffDatabaseMapping instances

connect_signals()

Connect signals to slots.

remove_selected_rows(*checked=True*)

all_databases(*row*)

Returns a list of db names available for a given row. Used by delegates.

```
class spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog(parent,
                                                                                   db_mgr,
                                                                                   *db_maps)
```

Bases: *spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ShowIconColorEditorMixin, AddItemsDialog*

A dialog to query user's preferences for new object classes.

Parameters

- **parent** (*SpineDBEditor*) –
- **db_mgr** (*SpineDBManager*) –
- ***db_maps** – DiffDatabaseMapping instances

connect_signals()

Connect signals to slots.

all_db_maps(row)

Returns a list of db maps available for a given row. Used by ShowIconColorEditorMixin.

accept()

Collect info from dialog and try to add items.

```
class spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectsDialog(parent,
                                                                                   parent_item,
                                                                                   db_mgr,
                                                                                   *db_maps,
                                                                                   force_default=False)
```

Bases: *spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetObjectClassesMixin, AddItemsDialog*

A dialog to query user's preferences for new objects.

Parameters

- **parent** (*SpineDBEditor*) –
- **parent_item** (*MultiDBTreeItem*) –
- **db_mgr** (*SpineDBManager*) –
- ***db_maps** – DiffDatabaseMapping instances
- **force_default** (*bool*) – if True, defaults are non-editable

accept()

Collect info from dialog and try to add items.

```
class spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddRelationshipClassesDialog(parent,
                                                                                   par-
                                                                                   ent_item,
                                                                                   db_mgr,
                                                                                   *db_maps,
                                                                                   force_default=False)
```

Bases: *spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ShowIconColorEditorMixin, spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetObjectClassesMixin, AddItemsDialog*

A dialog to query user's preferences for new relationship classes.

Parameters

- **parent** ([SpineDBEditor](#)) –
- **parent_item** ([MultiDBTreeItem](#)) –
- **db_mgr** ([SpineDBManager](#)) –
- ***db_maps** – DiffDatabaseMapping instances
- **force_default** (*bool*) – if True, defaults are non-editable

connect_signals()

Connect signals to slots.

_handle_spin_box_value_changed(*i*)

insert_column()

remove_column()

_handle_model_data_changed(*top_left, bottom_right, roles*)

Reimplement in subclasses to handle changes in model data.

accept()

Collect info from dialog and try to add items.

```
class spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddOrManageRelationshipsDialog(parent,
                                                                                          db_mgr,
                                                                                          *db_maps)
```

Bases: [spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetRelationshipClassesMixin](#), [spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetObjectMixins](#), [AddItemsDialog](#)

A dialog to query user's preferences for new relationships.

Parameters

- **parent** ([SpineDBEditor](#)) –
- **db_mgr** ([SpineDBManager](#)) –
- ***db_maps** – DiffDatabaseMapping instances

abstract make_model()

connect_signals()

Connect signals to slots.

abstract reset_model(*index*)

Called when relationship_class's combobox's index changes. Update relationship_class attribute accordingly and reset model.

```
class spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddRelationshipsDialog(parent,
                                                                                      par-
                                                                                      ent_item,
                                                                                      db_mgr,
                                                                                      *db_maps,
                                                                                      force_default=False)
```

Bases: [AddOrManageRelationshipsDialog](#)

A dialog to query user's preferences for new relationships.

Parameters

- **parent** ([SpineDBEditor](#)) –
- **parent_item** ([MultiDBTreeItem](#)) –
- **db_mgr** ([SpineDBManager](#)) –
- ***db_maps** – DiffDatabaseMapping instances
- **force_default** (*bool*) – if True, defaults are non-editable

make_model()**reset_model**(*index*)

Setup model according to current relationship_class selected in combobox.

_handle_model_data_changed(*top_left, bottom_right, roles*)

Reimplement in subclasses to handle changes in model data.

accept()

Collect info from dialog and try to add items.

```
class spinetoolbox.spine_db_editor.widgets.add_items_dialogs.ManageRelationshipsDialog(parent,  
                                                                                      par-  
                                                                                      ent_item,  
                                                                                      db_mgr,  
                                                                                      *db_maps)
```

Bases: [AddOrManageRelationshipsDialog](#)

A dialog to query user's preferences for managing relationships.

Parameters

- **parent** ([SpineDBEditor](#)) – data store widget
- **parent_item** ([MultiDBTreeItem](#)) –
- **db_mgr** ([SpineDBManager](#)) – the manager to do the removal
- ***db_maps** – DiffDatabaseMapping instances

make_model()**splitter_widgets()****connect_signals()**

Connect signals to slots.

reset_relationship_class_combo_box(*database, relationship_class_key=None*)**add_relationships**(*checked=True*)**reset_model**(*index*)

Setup model according to current relationship_class selected in combobox.

resize_window_to_columns(*height=None*)**accept()**

Collect info from dialog and try to add items.

```
class spinetoolbox.spine_db_editor.widgets.add_items_dialogs.ObjectGroupDialogBase(parent,
                                                                                   ob-
                                                                                   ject_class_item,
                                                                                   db_mngr,
                                                                                   *db_maps)
```

Bases: PySide6.QtWidgets.QDialog

Parameters

- **parent** ([SpineDBEditor](#)) – data store widget
- **object_class_item** ([ObjectClassItem](#)) –
- **db_mngr** ([SpineDBManager](#)) –
- ***db_maps** – database mappings

connect_signals()

Connect signals to slots.

reset_list_widgets(database)

abstract initial_member_ids()

abstract initial_entity_id()

add_members(checked=False)

remove_members(checked=False)

_check_validity()

```
class spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectGroupDialog(parent,
                                                                                   ob-
                                                                                   ject_class_item,
                                                                                   db_mngr,
                                                                                   *db_maps)
```

Bases: [ObjectGroupDialogBase](#)

Parameters

- **parent** ([SpineDBEditor](#)) – data store widget
- **object_class_item** ([ObjectClassItem](#)) –
- **db_mngr** ([SpineDBManager](#)) –
- ***db_maps** – database mappings

initial_member_ids()

initial_entity_id()

_check_validity()

accept()

```
class spinetoolbox.spine_db_editor.widgets.add_items_dialogs.ManageMembersDialog(parent, ob-
                                                                                   ject_item,
                                                                                   db_mngr,
                                                                                   *db_maps)
```

Bases: [ObjectGroupDialogBase](#)

Parameters

- **parent** ([SpineDBEditor](#)) – data store widget
- **object_item** ([entity_tree_item.ObjectItem](#)) –
- **db_mgr** ([SpineDBManager](#)) –
- ***db_maps** – database mappings

_entity_groups()**initial_member_ids()****initial_entity_id()****accept()****spinetoolbox.spine_db_editor.widgets.commit_viewer**

Contains the CommitViewer class.

Module Contents**Classes**

<i>_DBCommitViewer</i>	
<i>_CommitItem</i>	A widget to show commit message, author and data on a QTreeWidgetItem.
<i>_AffectedItemsFromOneTable</i>	A widget to show all the items from one table that are affected by a commit.
<i>CommitViewer</i>	

param	qsettings
--------------	------------------

```
class spinetoolbox.spine_db_editor.widgets.commit_viewer._DBCommitViewer(db_mgr, db_map,  
                                                                    parent=None)
```

Bases: PySide6.QtWidgets.QWidget

_select_commit(*current, previous*)**_do_select_commit**(*current*)

```
class spinetoolbox.spine_db_editor.widgets.commit_viewer._CommitItem(commit, parent=None)
```

Bases: PySide6.QtWidgets.QWidget

A widget to show commit message, author and data on a QTreeWidgetItem.

```
class spinetoolbox.spine_db_editor.widgets.commit_viewer._AffectedItemsFromOneTable(items,  
                                                                    par-  
                                                                    ent=None)
```

Bases: PySide6.QtWidgets.QTreeWidgetItem

A widget to show all the items from one table that are affected by a commit.

moveEvent(*ev*)

sizeHint()

class spinetoolbox.spine_db_editor.widgets.commit_viewer.**CommitViewer**(*qsettings*, *db_mgr*,
**db_maps*,
parent=None)

Bases: PySide6.QtWidgets.QMainWindow

Parameters

- **qsettings** (*QSettings*) –
- **db_mgr** (*SpineDBManager*) –
- **db_maps** (*DiffDatabaseMapping*) –

_carry_splitter_state(*index*)

closeEvent(*ev*)

spinetoolbox.spine_db_editor.widgets.custom_delegates

Custom item delegates.

Module Contents

Classes

<i>PivotTableDelegateMixin</i>	A mixin that fixes Pivot table's header table editor position.
<i>RelationshipPivotTableDelegate</i>	A mixin that fixes Pivot table's header table editor position.
<i>ScenarioAlternativeTableDelegate</i>	A mixin that fixes Pivot table's header table editor position.
<i>ParameterPivotTableDelegate</i>	A mixin that fixes Pivot table's header table editor position.
<i>ParameterValueElementDelegate</i>	Delegate for Array and Map editors' table cells.
<i>ParameterDelegate</i>	Base class for all custom parameter delegates.
<i>DatabaseNameDelegate</i>	A delegate for the database name.
<i>ParameterValueOrDefaultValueDelegate</i>	A delegate for either the value or the default value.
<i>ParameterDefaultValueDelegate</i>	A delegate for the default value.
<i>ParameterValueDelegate</i>	A delegate for the parameter_value.
<i>ValueListDelegate</i>	A delegate for the parameter value list.
<i>ObjectClassNameDelegate</i>	A delegate for the object_class name.
<i>RelationshipClassNameDelegate</i>	A delegate for the relationship_class name.
<i>ParameterNameDelegate</i>	A delegate for the object parameter name.
<i>ObjectNameDelegate</i>	A delegate for the object name.
<i>AlternativeNameDelegate</i>	A delegate for the object name.
<i>ObjectNameListDelegate</i>	A delegate for the object name list.
<i>ToolFeatureDelegate</i>	A delegate for the tool feature tree.
<i>AlternativeDelegate</i>	A delegate for the alternative tree.
<i>ScenarioDelegate</i>	A delegate for the scenario tree.
<i>ParameterValueListDelegate</i>	A delegate for the parameter value list tree.
<i>ManageItemsDelegate</i>	A custom delegate for the model in {Add/Edit}ItemDialogs.
<i>ManageEntityClassesDelegate</i>	A custom delegate for the model in {Add/Edit}ItemDialogs.
<i>ManageObjectClassesDelegate</i>	A delegate for the model and view in {Add/Edit}ObjectClassesDialog.
<i>ManageObjectsDelegate</i>	A delegate for the model and view in {Add/Edit}ObjectsDialog.
<i>ManageRelationshipClassesDelegate</i>	A delegate for the model and view in {Add/Edit}RelationshipClassesDialog.
<i>ManageRelationshipsDelegate</i>	A delegate for the model and view in {Add/Edit}RelationshipsDialog.
<i>RemoveEntitiesDelegate</i>	A delegate for the model and view in RemoveEntitiesDialog.
<i>ItemMetadataDelegate</i>	A delegate for name and value columns in item metadata editor.

class `spinetoolbox.spine_db_editor.widgets.custom_delegates.PivotTableDelegateMixin`

A mixin that fixes Pivot table's header table editor position.

updateEditorGeometry(*editor, option, index*)

Fixes position of header table editors.

class `spinetoolbox.spine_db_editor.widgets.custom_delegates.RelationshipPivotTableDelegate(parent)`

Bases: *PivotTableDelegateMixin*, *spinetoolbox.widgets.custom_delegates.CheckBoxDelegate*

A mixin that fixes Pivot table's header table editor position.

Parameters

parent ([SpineDBEditor](#)) – parent widget, i.e. the database editor

data_committed

static `_is_relationship_index(index)`

Checks whether the given index corresponds to a relationship, in which case we need to use the check box delegate.

Parameters

index ([QModelIndex](#)) – index to check

Returns

True if index corresponds to relationship, False otherwise

Return type

bool

setModelData(*editor, model, index*)

Send signal.

setEditorData(*editor, index*)

Do nothing. We're setting editor data right away in createEditor.

paint(*painter, option, index*)

Paint a checkbox without the label.

editorEvent(*event, model, option, index*)

Change the data in the model and the state of the checkbox when user presses left mouse button and this cell is editable. Otherwise do nothing.

createEditor(*parent, option, index*)

Important, otherwise an editor is created if the user clicks in this cell. ** Need to hook up a signal to the model.

class `spinetoolbox.spine_db_editor.widgets.custom_delegates.ScenarioAlternativeTableDelegate(parent)`

Bases: [PivotTableDelegateMixin](#), [spinetoolbox.widgets.custom_delegates.RankDelegate](#)

A mixin that fixes Pivot table's header table editor position.

Parameters

parent ([SpineDBEditor](#)) – database editor

data_committed

static `_is_scenario_alternative_index(index)`

Checks whether or not the given index corresponds to a scenario alternative, in which case we need to use the rank delegate.

Returns

bool

setModelData(*editor, model, index*)

Send signal.

setEditorData(*editor, index*)

Do nothing. We're setting editor data right away in createEditor.

paint(*painter, option, index*)

Paint a checkbox without the label.

editorEvent(*event, model, option, index*)

Change the data in the model and the state of the checkbox when user presses left mouse button and this cell is editable. Otherwise do nothing.

createEditor(*parent, option, index*)

Important, otherwise an editor is created if the user clicks in this cell. ** Need to hook up a signal to the model.

class `spinetoolbox.spine_db_editor.widgets.custom_delegates.ParameterPivotTableDelegate`(*parent*)

Bases: `PivotTableDelegateMixin`, `PySide6.QtWidgets.QStyledItemDelegate`

A mixin that fixes Pivot table's header table editor position.

Parameters

parent (`SpineDBEditor`) – parent widget, i.e. database editor

parameter_value_editor_requested

data_committed

setModelData(*editor, model, index*)

Send signal.

setEditorData(*editor, index*)

Do nothing. We're setting editor data right away in createEditor.

createEditor(*parent, option, index*)

class `spinetoolbox.spine_db_editor.widgets.custom_delegates.ParameterValueElementDelegate`

Bases: `PySide6.QtWidgets.QStyledItemDelegate`

Delegate for Array and Map editors' table cells.

value_editor_requested

Emitted when editing the value requires the full blown editor dialog.

setModelData(*editor, model, index*)

Sets data in the model.

editor (`CustomLineEditor`): editor widget model (`QAbstractItemModel`): model index (`QModelIndex`): target index

createEditor(*parent, option, index*)

Creates an editor widget or emits `value_editor_requested` for complex values.

Parameters

- **parent** (`QWidget`) – parent widget
- **option** (`QStyleOptionViewItem`) – unused
- **index** (`QModelIndex`) – element's model index

Returns

editor widget

Return type

`ParameterValueLineEditor`


```
class spinetoolbox.spine_db_editor.widgets.custom_delegates.ParameterDelegate(parent,  
                                                                           db_mgr)
```

Bases: PySide6.QtWidgets.QStyledItemDelegate

Base class for all custom parameter delegates.

db_mgr

database manager

Type

SpineDBManager

Parameters

- **parent** (*QWidget*) – parent widget
- **db_mgr** (*SpineDBManager*) – database manager

data_committed

setModelData(*editor*, *model*, *index*)

Send signal.

setEditorData(*editor*, *index*)

Do nothing. We're setting editor data right away in createEditor.

updateEditorGeometry(*editor*, *option*, *index*)

_close_editor(*editor*, *index*)

Closes editor. Needed by SearchBarEditor.

_get_db_map(*index*)

Returns the db_map for the database at given index or None if not set yet.

```
class spinetoolbox.spine_db_editor.widgets.custom_delegates.DatabaseNameDelegate(parent,  
                                                                           db_mgr)
```

Bases: *ParameterDelegate*

A delegate for the database name.

Parameters

- **parent** (*QWidget*) – parent widget
- **db_mgr** (*SpineDBManager*) – database manager

createEditor(*parent*, *option*, *index*)

Returns editor.

```
class spinetoolbox.spine_db_editor.widgets.custom_delegates.ParameterValueOrDefaultValueDelegate(parent,  
                                                                           db_mgr)
```

Bases: *ParameterDelegate*

A delegate for either the value or the default value.

Parameters

- **parent** (*QWidget*) – parent widget
- **db_mgr** (*SpineDatabaseManager*) – database manager

parameter_value_editor_requested

setModelData(*editor, model, index*)

Send signal.

_create_or_request_parameter_value_editor(*parent, index*)

Emits the signal to request a standalone *ParameterValueEditor* from parent widget.

Parameters

- **parent** (*QWidget*) – editor’s parent widget
- **index** (*QModelIndex*) – index to parameter value model

Returns

editor or None if `parameter_value_editor_request` signal was emitted

Return type

ParameterValueLineEdit

abstract _get_value_list_id(*index, db_map*)

Returns a value list id for the given index and `db_map`.

Parameters

- **index** (*QModelIndex*) – value list’s index
- **db_map** (*DiffDatabaseMapping*) – database mapping

Returns

value list id

Return type

int

createEditor(*parent, option, index*)

If the parameter has associated a value list, returns a *SearchBarEditor*. Otherwise returns or requests a dedicated `parameter_value` editor.

class `spinetoolbox.spine_db_editor.widgets.custom_delegates.ParameterDefaultValueDelegate`(*parent, db_mgr*)

Bases: *ParameterValueOrDefaultValueDelegate*

A delegate for the default value.

Parameters

- **parent** (*QWidget*) – parent widget
- **db_mgr** (*SpineDatabaseManager*) – database manager

_get_value_list_id(*index, db_map*)

See base class

class `spinetoolbox.spine_db_editor.widgets.custom_delegates.ParameterValueDelegate`(*parent, db_mgr*)

Bases: *ParameterValueOrDefaultValueDelegate*

A delegate for the `parameter_value`.

Parameters

- **parent** (*QWidget*) – parent widget

- **db_mgr** (*SpineDatabaseManager*) – database manager

_get_value_list_id(*index, db_map*)

See base class.

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**ValueListDelegate**(*parent, db_mgr*)

Bases: *ParameterDelegate*

A delegate for the parameter value list.

Parameters

- **parent** (*QWidget*) – parent widget
- **db_mgr** (*SpineDBManager*) – database manager

createEditor(*parent, option, index*)

Returns editor.

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**ObjectClassNameDelegate**(*parent, db_mgr*)

Bases: *ParameterDelegate*

A delegate for the object_class name.

Parameters

- **parent** (*QWidget*) – parent widget
- **db_mgr** (*SpineDBManager*) – database manager

createEditor(*parent, option, index*)

Returns editor.

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**RelationshipClassNameDelegate**(*parent, db_mgr*)

Bases: *ParameterDelegate*

A delegate for the relationship_class name.

Parameters

- **parent** (*QWidget*) – parent widget
- **db_mgr** (*SpineDBManager*) – database manager

createEditor(*parent, option, index*)

Returns editor.

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**ParameterNameDelegate**(*parent, db_mgr*)

Bases: *ParameterDelegate*

A delegate for the object parameter name.

Parameters

- **parent** (*QWidget*) – parent widget
- **db_mgr** (*SpineDBManager*) – database manager

createEditor(*parent, option, index*)

Returns editor.

```
class spinetoolbox.spine_db_editor.widgets.custom_delegates.ObjectNameDelegate(parent,
                                                                              db_mgr)
```

Bases: *ParameterDelegate*

A delegate for the object name.

Parameters

- **parent** (*QWidget*) – parent widget
- **db_mgr** (*SpineDBManager*) – database manager

```
createEditor(parent, option, index)
```

Returns editor.

```
class spinetoolbox.spine_db_editor.widgets.custom_delegates.AlternativeNameDelegate(parent,
                                                                              db_mgr)
```

Bases: *ParameterDelegate*

A delegate for the object name.

Parameters

- **parent** (*QWidget*) – parent widget
- **db_mgr** (*SpineDBManager*) – database manager

```
createEditor(parent, option, index)
```

Returns editor.

```
class spinetoolbox.spine_db_editor.widgets.custom_delegates.ObjectNameListDelegate(parent,
                                                                              db_mgr)
```

Bases: *ParameterDelegate*

A delegate for the object name list.

Parameters

- **parent** (*QWidget*) – parent widget
- **db_mgr** (*SpineDBManager*) – database manager

```
object_name_list_editor_requested
```

```
createEditor(parent, option, index)
```

Returns editor.

```
class spinetoolbox.spine_db_editor.widgets.custom_delegates.ToolFeatureDelegate(*args,
                                                                              **kwargs)
```

Bases: *PySide6.QtWidgets.QStyledItemDelegate*

A delegate for the tool feature tree.

Parameters

- ***args** – arguments passed to *QStyledItemDelegate*
- ****kwargs** – keyword arguments passed to *QStyledItemDelegate*

```
data_committed
```

_get_names(*item*, *model*)

Collects names under given tree item.

Parameters

- **item** (*Standard tree item*) – A non-leaf item
- **model** (*ToolFeatureModel*) – model

Returns

names or None if nothing was found

Return type

list of str

static _get_index_data(*item*, *index*)

Returns formatted model data from given index.

Parameters

- **item** (*StandardTreeItem*) – item corresponding to index
- **index** (*QModelIndex*) – index

Returns

index data

Return type

str

setModelData(*editor*, *model*, *index*)

Send signal.

setEditorData(*editor*, *index*)

Do nothing. We're setting editor data right away in createEditor.

createEditor(*parent*, *option*, *index*)

Returns editor.

updateEditorGeometry(*editor*, *option*, *index*)

_close_editor(*editor*, *index*)

Closes editor.

Needed by SearchBarEditor.

Parameters

- **editor** (*QWidget*) – editor widget
- **index** (*QModelIndex*) – index that is being edited

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**AlternativeDelegate**(*args,
**kwargs)

Bases: PySide6.QtWidgets.QStyledItemDelegate

A delegate for the alternative tree.

Parameters

- ***args** – arguments passed to QStyledItemDelegate
- ****kwargs** – keyword arguments passed to QStyledItemDelegate

data_committed

setModelData(*editor, model, index*)

Send signal.

setEditorData(*editor, index*)

Do nothing. We're setting editor data right away in createEditor.

createEditor(*parent, option, index*)

Returns editor.

updateEditorGeometry(*editor, option, index*)

_close_editor(*editor, index*)

Closes editor.

Needed by SearchBarEditor.

Parameters

- **editor** (*QWidget*) – editor widget
- **index** (*QModelIndex*) – index that is being edited

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**ScenarioDelegate**(*args,
**kwargs)

Bases: PySide6.QtWidgets.QStyledItemDelegate

A delegate for the scenario tree.

Parameters

- ***args** – arguments passed to QStyledItemDelegate
- ****kwargs** – keyword arguments passed to QStyledItemDelegate

data_committed

setModelData(*editor, model, index*)

Send signal.

setEditorData(*editor, index*)

Do nothing. We're setting editor data right away in createEditor.

_get_names(*item*)

Collects available alternative names avoiding duplicates in a scenario.

Excludes alternatives that are already in the scenario

Parameters

item (*ScenarioAlternativeItem*) – one of scenario's scenario alternatives

Returns

available alternative names

Return type

list of str

createEditor(*parent, option, index*)

Returns editor.

updateEditorGeometry(*editor, option, index*)

_close_editor(*editor, index*)

Closes editor.

Needed by SearchBarEditor.

Parameters

- **editor** (*QWidget*) – editor widget
- **index** (*QModelIndex*) – index that is being edited

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**ParameterValueListDelegate**

Bases: PySide6.QtWidgets.QStyledItemDelegate

A delegate for the parameter value list tree.

data_committed

parameter_value_editor_requested

setModelData(*editor, model, index*)

Send signal.

setEditorData(*editor, index*)

Do nothing. We're setting editor data right away in createEditor.

createEditor(*parent, option, index*)

Returns editor.

_close_editor(*editor, index*)

Closes editor.

Needed by SearchBarEditor.

Parameters

- **editor** (*QWidget*) – editor widget
- **index** (*QModelIndex*) – index that is being edited

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**ManageItemsDelegate**

Bases: PySide6.QtWidgets.QStyledItemDelegate

A custom delegate for the model in {Add/Edit}ItemDialogs.

data_committed

setModelData(*editor, model, index*)

Send signal.

close_editor(*editor, index*)

Closes editor.

Needed by SearchBarEditor.

Parameters

- **editor** (*QWidget*) – editor widget
- **index** (*QModelIndex*) – index that is being edited

updateEditorGeometry(*editor, option, index*)

connect_editor_signals(*editor, index*)

Connect editor signals if necessary.

Parameters

- **editor** (*QWidget*) – editor widget
- **index** (*QModelIndex*) – index being edited

_create_database_editor(*parent, index*)

Creates an editor.

Parameters

- **parent** (*QWidget*) – parent widget
- **index** (*QModelIndex*) – index being edited

Returns

editor

Return type

QWidget

createEditor(*parent, option, index*)

Returns an editor.

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**ManageEntityClassesDelegate**

Bases: [*ManageItemsDelegate*](#)

A custom delegate for the model in {Add/Edit}ItemDialogs.

paint(*painter, option, index*)

Get a pixmap from the index data and paint it in the middle of the cell.

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**ManageObjectClassesDelegate**

Bases: [*ManageEntityClassesDelegate*](#)

A delegate for the model and view in {Add/Edit}ObjectClassesDialog.

icon_color_editor_requested

createEditor(*parent, option, index*)

Return editor.

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**ManageObjectsDelegate**

Bases: [*ManageItemsDelegate*](#)

A delegate for the model and view in {Add/Edit}ObjectsDialog.

createEditor(*parent, option, index*)

Return editor.

class

spinetoolbox.spine_db_editor.widgets.custom_delegates.**ManageRelationshipClassesDelegate**

Bases: [*ManageEntityClassesDelegate*](#)

A delegate for the model and view in {Add/Edit}RelationshipClassesDialog.

icon_color_editor_requested

createEditor(*parent, option, index*)

Return editor.

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**ManageRelationshipsDelegate**

Bases: [ManageItemsDelegate](#)

A delegate for the model and view in {Add/Edit}RelationshipsDialog.

createEditor(*parent, option, index*)

Return editor.

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**RemoveEntitiesDelegate**

Bases: [ManageItemsDelegate](#)

A delegate for the model and view in RemoveEntitiesDialog.

createEditor(*parent, option, index*)

Return editor.

class spinetoolbox.spine_db_editor.widgets.custom_delegates.**ItemMetadataDelegate**(*item_metadata_model, meta-data_model, column, parent*)

Bases: PySide6.QtWidgets.QStyledItemDelegate

A delegate for name and value columns in item metadata editor.

Parameters

- **item_metadata_model** (*ItemMetadataModel*) – item metadata model
- **metadata_model** (*MetadataTableModel*) – metadata model
- **column** (*int*) – item metadata table column column
- **parent** (*QObject, optional*) – parent object

createEditor(*parent, option, index*)

spinetoolbox.spine_db_editor.widgets.custom_menus

Classes for custom context menus and pop-up menus.

Module Contents

Classes

[*MainMenu*](#)

[*ParameterViewFilterMenu*](#)

Filter menu.

[*TabularViewFilterMenu*](#)

Filter menu to use together with FilterWidget in TabularViewMixin.

```
class spinetoolbox.spine_db_editor.widgets.custom_menus.MainMenu
```

Bases: PySide6.QtWidgets.QMenu

event(*ev*)

Intercepts shortcuts and instead sends an equivalent event with the ‘Alt’ modifier, so that mnemonics works with just the key. Also sends a key press event with the ‘Alt’ key when this menu shows, so that mnemonics are underlined on Windows.

```
class spinetoolbox.spine_db_editor.widgets.custom_menus.ParameterViewFilterMenu(parent,
                                                                              db_mgr,
                                                                              db_maps,
                                                                              item_type,
                                                                              en-
                                                                              tity_class_id_key,
                                                                              field,
                                                                              show_empty=True)
```

Bases: *spinetoolbox.widgets.custom_menus.FilterMenuBase*

Filter menu.

Parameters

- **parent** (*SpineDBEditor*) –
- **db_mgr** (*SpineDBManager*) –
- **db_maps** (*Sequence of DatabaseMapping*) –
- **item_type** (*str*) –
- **entity_class_id_key** (*str*) –
- **field** (*str*) – the field name

filterChanged

set_filter_accepted_values(*accepted_values*)

set_filter_rejected_values(*rejected_values*)

_get_value(*item, db_map*)

_get_display_value(*item, db_map*)

_accepts_item(*item, db_map*)

_handle_items_added(*db_map_data*)

_handle_items_removed(*db_map_data*)

_build_auto_filter(*valid_values*)

Builds the auto filter given valid values.

Parameters

valid_values (*Sequence*) – Values accepted by the filter.

Returns

mapping (db_map, entity_class_id) to set of valid values

Return type

dict

emit_filter_changed(*valid_values*)

Builds auto filter and emits signal.

Parameters

valid_values (*Sequence*) – Values accepted by the filter.

```
class spinetoolbox.spine_db_editor.widgets.custom_menus.TabularViewFilterMenu(parent,
                                                                              db_mgr,
                                                                              db_maps,
                                                                              item_type,
                                                                              accepts_item,
                                                                              identifier,
                                                                              show_empty=True)
```

Bases: `spinetoolbox.widgets.custom_menus.FilterMenuBase`

Filter menu to use together with FilterWidget in TabularViewMixin.

Parameters

- **parent** (`SpineDBEditor`) –
- **db_mgr** (`SpineDBManager`) –
- **db_maps** (*Sequence of DatabaseMapping*) –
- **item_type** (*str*) –
- **accepts_item** (*function*) –
- **identifier** (*int*) – index identifier

filterChanged

_handle_items_added(*db_map_data*)

_get_values(*db_map, item*)

_handle_items_removed(*db_map_data*)

emit_filter_changed(*valid_values*)

event(*event*)

spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews

Classes for custom QGraphicsViews for the Entity graph view.

Module Contents

Classes

EntityQGraphicsView

QGraphicsView for the Entity Graph View.

class `spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView`(*parent*)

Bases: `spinetoolbox.widgets.custom_qgraphicsviews.CustomQGraphicsView`

QGraphicsView for the Entity Graph View.

Parameters

parent (*QWidget*) – Graph View Form’s (QMainWindow) central widget (self.centralwidget)

property `_qsettings`

property `db_mgr`

property `entity_items`

graph_selection_changed

handle_scene_selection_changed()

Filters parameters by selected objects in the graph.

connect_spine_db_editor(*spine_db_editor*)

populate_context_menu()

increase_arc_length()

decrease_arc_length()

_update_actions_visibility()

Enables or disables actions according to current selection in the graph.

make_items_menu()

_set_auto_expand_objects(*_checked=False, save_setting=True*)

set_auto_expand_objects(*checked*)

_set_merge_dbs(*_checked=False, save_setting=True*)

set_merge_dbs(*checked*)

_set_disable_max_relationship_dimension(*_checked=False, save_setting=True*)

set_disable_max_relationship_dimension(*checked*)

_set_max_relationship_dimension(*_value=None, save_setting=True*)

add_objects_at_position(*checked=False*)

edit_selected(*_=False*)

Edits selected items.

remove_selected(*_=False*)

Removes selected items.

_get_selected_entity_names()

hide_selected_items(*checked=False*)

Hides selected items.

_hide_class(*action*)

Hides some class.

show_all_hidden_items(*checked=False*)

Shows all hidden items.

show_hidden_items(*action*)

Shows some hidden items.

prune_selected_items(*checked=False*)

Prunes selected items.

_prune_class(*action*)

Prunes some class.

restore_all_pruned_items(*checked=False*)

Reinstates all pruned items.

restore_pruned_items(*action*)

Reinstates some pruned items.

select_position_parameters(*checked=False*)

_set_position_parameters(*parameter_pos_x, parameter_pos_y*)

save_positions(*checked=False*)

clear_saved_positions(*checked=False*)

export_as_pdf(*_=False*)

_populate_add_heat_map_menu()

Populates the menu ‘Add heat map’ with parameters for currently shown items in the graph.

add_heat_map(*action*)

Adds heat map for the parameter in the action text.

_clean_up_heat_map_items()

set_cross_hairs_items(*relationship_class, cross_hairs_items*)

Sets ‘cross_hairs’ items for relationship creation.

Parameters

- **relationship_class** (*dict*) –
- **cross_hairs_items** (*list(QGraphicsItems)*) –

clear_cross_hairs_items()

_cross_hairs_has_valid_target()

mousePressEvent(*event*)

Handles relationship creation if one it’s in process.

mouseMoveEvent(*event*)

Updates the hovered object item if we’re in relationship creation mode.

_update_cross_hairs_pos(*pos*)

Updates the hovered object item and sets the ‘cross_hairs’ icon accordingly.

Parameters

- pos** (*QPoint*) – the desired position in view coordinates

mouseReleaseEvent(*event*)

Reestablish scroll hand drag mode.

_scroll_scene_by(*dx, dy*)

keyPressEvent(*event*)

Aborts relationship creation if user presses ESC.

contextMenuEvent(*e*)

Shows context menu.

Parameters

e (*QContextMenuEvent*) – Context menu event

_compute_max_zoom()

_use_smooth_zoom()

_zoom(*factor*)

apply_zoom()

wheelEvent(*event*)

Zooms in/out. If user has pressed the shift key, rotates instead.

Parameters

event (*QWheelEvent*) – Mouse wheel event

_handle_rotation_time_line_advanced(*pos*)

Performs rotation whenever the smooth rotation time line advances.

_rotate(*angle*)

rotate_clockwise()

Performs a rotate clockwise with fixed angle.

rotate_anticlockwise()

Performs a rotate anticlockwise with fixed angle.

spinetoolbox.spine_db_editor.widgets.custom_qtableview

Custom QTableView classes that support copy-paste and the like.

Module Contents

Classes

<i>ParameterTableView</i>	Custom QTableView class with autofilter functionality.
<i>ObjectParameterTableMixin</i>	
<i>RelationshipParameterTableMixin</i>	
<i>ParameterDefinitionTableView</i>	Custom QTableView class with autofilter functionality.
<i>ParameterValueTableView</i>	Custom QTableView class with autofilter functionality.
<i>ObjectParameterDefinitionTableView</i>	A custom QTableView for the object parameter_definition pane in Spine db editor.
<i>RelationshipParameterDefinitionTableView</i>	A custom QTableView for the relationship parameter_definition pane in Spine db editor.
<i>ObjectParameterValueTableView</i>	A custom QTableView for the object parameter_value pane in Spine db editor.
<i>RelationshipParameterValueTableView</i>	A custom QTableView for the relationship parameter_value pane in Spine db editor.
<i>PivotTableView</i>	Custom QTableView class with pivot capabilities.
<i>FrozenTableView</i>	
<i>MetadataTableViewBase</i>	Base for metadata and item metadata table views.
<i>MetadataTableView</i>	Table view for metadata.
<i>ItemMetadataTableView</i>	Table view for entity and parameter value metadata.

Functions

<i>_set_parameter_data</i> (index, new_value)	Updates (object or relationship) parameter_definition or value with newly edited data.
---	--

`spinetoolbox.spine_db_editor.widgets.custom_qtableview._set_parameter_data(index, new_value)`

Updates (object or relationship) parameter_definition or value with newly edited data.

class `spinetoolbox.spine_db_editor.widgets.custom_qtableview.ParameterTableView(parent)`

Bases: `spinetoolbox.widgets.custom_qwidgets.ResizingViewMixin`, `spinetoolbox.widgets.custom_qtableview.AutoFilterCopyPasteTableView`

Custom QTableView class with autofilter functionality.

Parameters

parent (*QObject*) – parent object

value_column_header: `str`

Either “default value” or “value”. Used to identify the value column for advanced editing and plotting.

connect_spine_db_editor(*spine_db_editor*)

Connects a Spine db editor to work with this view.

Parameters

spine_db_editor (*SpineDBEditor*) –

_make_delegate(*column_name*, *delegate_class*)

Creates a delegate for the given column and returns it.

Parameters

- **column_name** (*str*) –
- **delegate_class** (*ParameterDelegate*) –

Returns

ParameterDelegate

create_delegates()

Creates delegates for this view

open_in_editor()

Opens the current index in a parameter_value editor using the connected Spine db editor.

plot(*checked=False*)

Plots current index.

abstract _plot_selection(*selection, plot_widget=None*)

Adds selected indexes to existing plot or creates a new plot window.

Parameters

- **selection** (*Iterable of QModelIndex*) – a list of QModelIndex objects for plotting
- **plot_widget** (*PlotWidget, optional*) – an existing plot widget to draw into or None to create a new widget

Returns

a PlotWidget object

Return type

PlotWidget

plot_in_window(*action*)

Plots current index in the window given by action's name.

populate_context_menu()

Creates a context menu for this view.

contextMenuEvent(*event*)

Shows context menu.

Parameters

- **event** (*QContextMenuEvent*) –

_selected_rows_per_column()

Computes selected rows per column.

Returns

Mapping columns to selected rows in that column.

Return type

dict

filter_by_selection(*checked=False*)

filter_excluding_selection(*checked=False*)

remove_selected()

Removes selected indexes.

_do_resize()

_refresh_copy_paste_actions(, __)

Enables or disables copy and paste actions.

class spinetoolbox.spine_db_editor.widgets.custom_qtableview.**ObjectParameterTableMixin**

create_delegates()

class

spinetoolbox.spine_db_editor.widgets.custom_qtableview.**RelationshipParameterTableMixin**

create_delegates()

class spinetoolbox.spine_db_editor.widgets.custom_qtableview.**ParameterDefinitionTableView**(parent)

Bases: [ParameterTableView](#)

Custom QTableView class with autofilter functionality.

Parameters

parent (*QObject*) – parent object

value_column_header = 'default_value'

create_delegates()

Creates delegates for this view

abstract **_plot_selection**(selection, plot_widget=None)

See base class

class spinetoolbox.spine_db_editor.widgets.custom_qtableview.**ParameterValueTableView**(parent)

Bases: [ParameterTableView](#)

Custom QTableView class with autofilter functionality.

Parameters

parent (*QObject*) – parent object

abstract **property** **_pk_fields**

value_column_header = 'value'

connect_spine_db_editor(spine_db_editor)

Connects a Spine db editor to work with this view.

Parameters

spine_db_editor ([SpineDBEditor](#)) –

create_delegates()

Creates delegates for this view

_update_pinned_values(selected, deselected)

_make_pinned_value(index)

abstract **_plot_selection**(selection, plot_widget=None)

See base class

class `spinetoolbox.spine_db_editor.widgets.custom_qtableview.ObjectParameterDefinitionTableView(parent)`

Bases: *ObjectParameterTableMixin*, *ParameterDefinitionTableView*

A custom QTableView for the object parameter_definition pane in Spine db editor.

Parameters

parent (*QObject*) – parent object

_plot_selection(*selection*, *plot_widget=None*)

See base class

class `spinetoolbox.spine_db_editor.widgets.custom_qtableview.RelationshipParameterDefinitionTableView(p`

Bases: *RelationshipParameterTableMixin*, *ParameterDefinitionTableView*

A custom QTableView for the relationship parameter_definition pane in Spine db editor.

Parameters

parent (*QObject*) – parent object

_plot_selection(*selection*, *plot_widget=None*)

See base class

class `spinetoolbox.spine_db_editor.widgets.custom_qtableview.ObjectParameterValueTableView(parent)`

Bases: *ObjectParameterTableMixin*, *ParameterValueTableView*

A custom QTableView for the object parameter_value pane in Spine db editor.

Parameters

parent (*QObject*) – parent object

property **_pk_fields**

create_delegates()

Creates delegates for this view

_plot_selection(*selection*, *plot_widget=None*)

See base class.

class `spinetoolbox.spine_db_editor.widgets.custom_qtableview.RelationshipParameterValueTableView(parent)`

Bases: *RelationshipParameterTableMixin*, *ParameterValueTableView*

A custom QTableView for the relationship parameter_value pane in Spine db editor.

Parameters

parent (*QObject*) – parent object

property **_pk_fields**

create_delegates()

Creates delegates for this view

_plot_selection(*selection*, *plot_widget=None*)

See base class.

class `spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView(parent=None)`

Bases: *spinetoolbox.widgets.custom_qwidgets.ResizingViewMixin*, *spinetoolbox.widgets.custom_qtableview.CopyPasteTableView*

Custom QTableView class with pivot capabilities.

Uses ‘contexts’ to provide different UI elements (table headers, context menus,...) depending on what data the pivot table currently contains.

Parameters

parent (*QWidget*, *optional*) – parent widget

class `_ContextBase`(*view*, *db_editor*, *horizontal_header*, *vertical_header*)

Base class for pivot table view's contexts.

Parameters

- **view** (*PivotTableView*) – parent view
- **db_editor** (*SpineDBEditor*) – database editor
- **horizontal_header** (*QHeaderView*) – horizontal header
- **vertical_header** (*QHeaderView*) – vertical header

`_REMOVE_OBJECT` = 'Remove objects'

`_REMOVE_RELATIONSHIP` = 'Remove relationships'

`_REMOVE_PARAMETER` = 'Remove parameter definitions'

`_REMOVE_ALTERNATIVE` = 'Remove alternatives'

`_REMOVE_SCENARIO` = 'Remove scenarios'

`_DUPLICATE_SCENARIO` = 'Duplicate scenario'

`_clear_selection_lists()`

Clears cached selected index lists.

abstract `populate_context_menu()`

Generates context menu.

`_refresh_selected_indexes()`

Caches selected index lists.

remove_alternatives()

Removes selected alternatives from the database.

show_context_menu(*position*)

Shows the context menu.

`_to_selection_lists`(*index*)

Caches given index to corresponding selected index list.

Parameters

index (*QModelIndex*) – index to cache

abstract `_update_actions_availability()`

Enables/disables context menu entries before the menu is shown.

class `_EntityContextBase`(*view*, *db_editor*, *horizontal_header*, *vertical_header*)

Bases: *PivotTableView._ContextBase*

Base class for contexts that contain entities and entity classes.

Parameters

- **view** (*PivotTableView*) – parent view
- **db_editor** (*SpineDBEditor*) – database editor
- **horizontal_header** (*QHeaderView*) – horizontal header

- **vertical_header** (*QHeaderView*) – vertical header

_can_remove_relationships()

Checks if it makes sense to remove selected relationships from the database.

Returns

True if relationships can be removed, False otherwise

Return type

bool

_clear_selection_lists()

See base class.

abstract populate_context_menu()

See base class.

remove_objects()

Removes selected objects from the database.

remove_relationships()

Removes selected relationships from the database.

abstract _update_actions_availability()

See base class.

class _ParameterValueContext(*view, db_editor*)

Bases: [*PivotTableView._EntityContextBase*](#)

Context for showing parameter values in the pivot table.

Parameters

- **view** ([*PivotTableView*](#)) – parent view
- **db_editor** ([*SpineDBEditor*](#)) – database editor

_clear_selection_lists()

See base class.

populate_context_menu()

See base class.

open_in_editor()

Opens the parameter value editor for the first selected cell.

plot()

Plots the selected cells.

_plot_in_window(*action*)

Plots the selected cells in an existing window.

remove_parameters()

Removes selected parameter definitions from the database.

remove_values()

Removes selected parameter values from the database.

show_context_menu(*position*)

Shows the context menu.

_to_selection_lists(*index*)

See base class.

_update_actions_availability()

See base class.

class _IndexExpansionContext(*view*, *db_editor*)

Bases: [PivotTableView._ParameterValueContext](#)

Context for expanded parameter values

Parameters

- **view** ([PivotTableView](#)) – parent view
- **db_editor** ([SpineDBEditor](#)) – database editor

class _RelationshipContext(*view*, *db_editor*)

Bases: [PivotTableView._EntityContextBase](#)

Context for presenting relationships in the pivot table.

Parameters

- **view** ([PivotTableView](#)) – parent view
- **db_editor** ([SpineDBEditor](#)) – database editor

populate_context_menu()

See base class.

_update_actions_availability()

See base class.

class _ScenarioAlternativeContext(*view*, *db_editor*)

Bases: [PivotTableView._ContextBase](#)

Context for presenting scenarios and alternatives

Parameters

- **view** ([PivotTableView](#)) – parent view
- **db_editor** ([SpineDBEditor](#)) – database editor

_clear_selection_lists()

See base class.

populate_context_menu()

See base class.

remove_scenarios()

Removes selected scenarios from the database.

duplicate_scenario()

Duplicates current scenario in the database.

_to_selection_lists(*index*)

See base class.

_update_actions_availability()

See base class.

`_open_scenario_generator()`

Opens the scenario generator dialog.

`_toggle_checked_state()`

Toggles the checked state of selected alternatives.

property `source_model`

property `db_mgr`

header_changed

`_do_resize()`

`connect_spine_db_editor(spine_db_editor)`

`_change_context()`

Changes the UI engine according to pivot model type.

`contextMenuEvent(event)`

Shows context menu.

Parameters

`event` (*QContextMenuEvent*) –

`setModel(model)`

`_synch_selection_with_header_tables(selected, deselected)`

`setIndexWidget(proxy_index, widget)`

`setHorizontalHeader(horizontal_header)`

`setVerticalHeader(vertical_header)`

`resizeEvent(ev)`

`_fetch_more_visible()`

`_update_header_tables()`

`_update_section_width(logical_index, _old_size, new_size)`

`_update_section_height(logical_index, _old_size, new_size)`

`_update_header_tables_geometry()`

`_refresh_copy_paste_actions(_, _)`

class `spinetoolbox.spine_db_editor.widgets.custom_qtableview.FrozenTableView`

Bases: `PySide6.QtWidgets.QTableView`

property `area`

header_dropped

`dragEnterEvent(event)`

`dragMoveEvent(event)`

dropEvent(*event*)

class spinetoolbox.spine_db_editor.widgets.custom_qtableview.**MetadataTableViewBase**(*parent*)

Bases: [spinetoolbox.widgets.custom_qtableview.CopyPasteTableView](#)

Base for metadata and item metadata table views.

Parameters

parent (*QWidget*, *optional*) – parent widget

connect_spine_db_editor(*db_editor*)

Finishes view's initialization.

Parameters

db_editor ([SpineDBEditor](#)) – database editor instance

contextMenuEvent(*event*)

_remove_selected()

Removes selected rows from view's model.

_enable_delegates(*db_editor*)

Creates delegates for this view

Parameters

db_editor ([SpineDBEditor](#)) – database editor

_populate_context_menu()

Fills context menu with actions.

_set_model_data(*index*, *value*)

Sets model data.

Parameters

- **index** (*QModelIndex*) – model index to set
- **value** (*str*) – value

_refresh_copy_paste_actions()

class spinetoolbox.spine_db_editor.widgets.custom_qtableview.**MetadataTableView**(*parent*)

Bases: [MetadataTableViewBase](#)

Table view for metadata.

Parameters

parent (*QWidget*, *optional*) – parent widget

_enable_delegates(*db_editor*)

See base class.

class spinetoolbox.spine_db_editor.widgets.custom_qtableview.**ItemMetadataTableView**(*parent*)

Bases: [MetadataTableViewBase](#)

Table view for entity and parameter value metadata.

Parameters

parent (*QWidget*) – parent widget

set_models(*item_metadata_model*, *metadata_model*)

Sets models.

Parameters

- **item_metadata_model** (*ItemMetadataModel*) – item metadata model
- **metadata_model** (*MetadataTableModel*) – metadata model

_enable_delegates(*db_editor*)

See base class

spinetoolbox.spine_db_editor.widgets.custom_qtreeview

Classes for custom QTreeView.

Module Contents

Classes

<i>ResizableTreeView</i>	Custom QTreeView class with copy and paste support.
<i>EntityTreeView</i>	Tree view base class for object and relationship tree views.
<i>ObjectTreeView</i>	Custom QTreeView class for the object tree in SpineDBEditor.
<i>RelationshipTreeView</i>	Custom QTreeView class for the relationship tree in SpineDBEditor.
<i>ItemTreeView</i>	Base class for all non-entity tree views.
<i>ToolFeatureTreeView</i>	Custom QTreeView class for tools and features in SpineDBEditor.
<i>AlternativeTreeView</i>	Custom QTreeView for the alternative tree in SpineDBEditor.
<i>ScenarioTreeView</i>	Custom QTreeView for the scenario tree in SpineDBEditor.
<i>ParameterValueListTreeView</i>	Custom QTreeView class for parameter_value_list in SpineDBEditor.

class spinetoolbox.spine_db_editor.widgets.custom_qtreeview.**ResizableTreeView**(*args,
**kwargs)

Bases: *spinetoolbox.widgets.custom_qwidgets.ResizingViewMixin*, *spinetoolbox.widgets.custom_qtreeview.CopyPasteTreeView*

Custom QTreeView class with copy and paste support.

_do_resize()

class spinetoolbox.spine_db_editor.widgets.custom_qtreeview.**EntityTreeView**(*parent*)

Bases: *ResizableTreeView*

Tree view base class for object and relationship tree views.

Parameters

parent (*QWidget*) – parent widget

relationship_selection_changed

object_selection_changed

reset()

connect_spine_db_editor(*spine_db_editor*)

Connects a Spine db editor to work with this view.

Parameters

spine_db_editor ([SpineDBEditor](#)) –

_add_middle_actions()

Adds action at the middle of the context menu. Subclasses can reimplement at will.

_create_context_menu()

Creates a context menu for this view.

edit(*index, trigger, event*)

Edit all selected items.

connect_signals()

Connects signals.

rowsInserted(*parent, start, end*)

rowsRemoved(*parent, start, end*)

setModel(*model*)

_fetch_more_visible()

verticalScrollbarValueChanged(*value*)

_handle_selection_changed(*selected, deselected*)

Classifies selection by item type and emits signal.

_refresh_selected_indexes()

clear_any_selections()

Clears the selection if any.

fully_expand()

Expands selected indexes and all their children.

fully_collapse()

Collapses selected indexes and all their children.

export_selected()

Exports data from selected indexes using the connected Spine db editor.

remove_selected()

Removes selected indexes using the connected Spine db editor.

manage_relationships()

contextMenuEvent(*event*)

Shows context menu.

Parameters

event ([QContextMenuEvent](#)) –

mousePressEvent(*event*)

Overrides selection behaviour if the user has selected sticky selection in Settings. If sticky selection is enabled, multiple-selection is enabled when selecting items in the Object tree. Pressing the Ctrl-button down, enables single selection.

Parameters

event (*QMouseEvent*) –

_add_relationship_actions()**update_actions_availability()**

Updates the visible property of actions according to whether or not they apply to given item.

edit_selected()

Edits all selected indexes using the connected Spine db editor.

class `spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView`(*parent*)

Bases: [*EntityTreeView*](#)

Custom QTreeView class for the object tree in SpineDBEditor.

Parameters

parent (*QWidget*) – parent widget

update_actions_availability()

Updates the visible property of actions according to whether or not they apply to given item.

_add_middle_actions()

Adds action at the middle of the context menu. Subclasses can reimplement at will.

connect_signals()

Connects signals.

rowsInserted(*parent, start, end*)**add_object_classes()****add_objects()****add_relationship_classes()****add_relationships()****find_next_relationship()**

Finds the next occurrence of the relationship at the current index and expands it.

_do_find_next_relationship()**duplicate_object()**

Duplicates the object at the current index using the connected Spine db editor.

add_object_group()**manage_members()**

class `spinetoolbox.spine_db_editor.widgets.custom_qtreeview.RelationshipTreeView`(*parent*)

Bases: [*EntityTreeView*](#)

Custom QTreeView class for the relationship tree in SpineDBEditor.

Parameters**parent** (*QWidget*) – parent widget**_add_middle_actions()**

Adds action at the middle of the context menu. Subclasses can reimplement at will.

update_actions_availability()

Updates the visible property of actions according to whether or not they apply to given item.

add_relationship_classes()**add_relationships()****class** `spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ItemTreeView(parent)`Bases: [*ResizableTreeView*](#)

Base class for all non-entity tree views.

Parameters**parent** (*QWidget*) – parent widget**rowsInserted(parent, start, end)****connect_signals()**

Connects signals.

abstract remove_selected()

Removes items selected in the view.

abstract update_actions_availability(item)

Updates the visible property of actions according to whether or not they apply to given item.

connect_spine_db_editor(spine_db_editor)

Prepares the view to work with the DB editor.

Parameters**spine_db_editor** ([*SpineDBEditor*](#)) – editor instance**populate_context_menu()**

Creates a context menu for this view.

contextMenuEvent(event)

Shows context menu.

Parameters**event** (*QContextMenuEvent*) –**_refresh_copy_paste_actions(, __)**

Refreshes copy and paste actions enabled state.

class `spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ToolFeatureTreeView(parent)`Bases: [*ItemTreeView*](#)

Custom QTreeView class for tools and features in SpineDBEditor.

Parameters**parent** (*QWidget*) – parent widget**connect_spine_db_editor(spine_db_editor)**

see base class

remove_selected()

See base class.

update_actions_availability(*item*)

See base class.

dragMoveEvent(*event*)

dragEnterEvent(*event*)

class spinetoolbox.spine_db_editor.widgets.custom_qtreeview.**AlternativeTreeView(*parent*)**

Bases: *ItemTreeView*

Custom QTreeView for the alternative tree in SpineDBEditor.

Parameters

parent (*QWidget*) – parent widget

property **selected_alternative_ids**

alternative_selection_changed

reset()

connect_signals()

Connects signals.

connect_spine_db_editor(*spine_db_editor*)

see base class

populate_context_menu()

See base class.

_db_map_alt_ids_from_selection(*selection*)

Gather alternative ids per database map from selection.

Parameters

selection (*QItemSelection*) – selection

Returns

mapping from database map to set of alternative ids

Return type

dict

_handle_selection_changed(*selected, deselected*)

Emits alternative_selection_changed with the current selection.

remove_selected()

See base class.

update_actions_availability(*item*)

See base class.

_open_scenario_generator()

Opens the scenario generator dialog.

can_copy()

See base class.

can_paste()

See base class.

copy()

See base class.

paste()

Pastes alternatives from clipboard to the tree.

This makes sense only when pasting alternatives from one database to another.

class `spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ScenarioTreeView(parent)`

Bases: *ItemTreeView*

Custom QTreeView for the scenario tree in SpineDBEditor.

Parameters

parent (*QWidget*) – parent widget

property `selected_alternative_ids`

scenario_selection_changed

reset()

connect_signals()

Connects signals.

connect_spine_db_editor(*spine_db_editor*)

see base class

populate_context_menu()

See base class.

_db_map_alternative_ids_from_selection(*selection*)

Collects database maps and alternative ids within given selection.

Parameters

selection (*Sequence of QModelIndex*) – selection indices

Returns

mapping from database map to set of alternative ids

Return type

dict

_handle_selection_changed(*selected, deselected*)

Emits `scenario_selection_changed` with the current selection.

remove_selected()

See base class.

dragMoveEvent(*event*)

dragEnterEvent(*event*)

update_actions_availability(*item*)

See base class

copy()

See base class.

can_paste()

See base class.

paste()

Pastes alternatives and scenarios from clipboard to the tree.

_duplicate_scenario()

Duplicates selected scenarios.

class spinetoolbox.spine_db_editor.widgets.custom_qtreeview.**ParameterValueListTreeView**(*parent*)

Bases: [ItemTreeView](#)

Custom QTreeView class for parameter_value_list in SpineDBEditor.

Parameters

parent (*QWidget*) – parent widget

connect_spine_db_editor(*spine_db_editor*)

see base class

populate_context_menu()

Creates a context menu for this view.

update_actions_availability(*item*)

See base class.

open_in_editor()

Opens the parameter_value editor for the first selected cell.

remove_selected()

See base class.

spinetoolbox.spine_db_editor.widgets.custom_qwidgets

Custom QWidgets.

Module Contents

Classes

<i>OpenFileButton</i>	A button to open files or show them in the folder.
<i>OpenSQLiteFileButton</i>	A button to open sqlite files, show them in the folder, or add them to the project.
<i>ShootingLabel</i>	

class spinetoolbox.spine_db_editor.widgets.custom_qwidgets.**OpenFileButton**(*file_path*,
db_editor)

Bases: PySide6.QtWidgets.QToolButton

A button to open files or show them in the folder.

open_file(*checked=False*)

open_containing_folder(*checked=False*)

class spinetoolbox.spine_db_editor.widgets.custom_qwidgets.**OpenSQLiteFileButton**(*file_path*,
db_editor)

Bases: *OpenFileButton*

A button to open sqlite files, show them in the folder, or add them to the project.

open_file(*checked=False*)

class spinetoolbox.spine_db_editor.widgets.custom_qwidgets.**ShootingLabel**(*origin*, *destination*,
parent=None,
duration=1200)

Bases: PySide6.QtWidgets.QLabel

_handle_value_changed(*value*)

show()

spinetoolbox.spine_db_editor.widgets.edit_or_remove_items_dialogs

Classes for custom QDialogs to edit items in databases.

Module Contents

Classes

<i>EditOrRemoveItemsDialog</i>	A dialog with a CopyPasteTableView and a QDialogButtonBox. Base class for all
<i>EditObjectClassesDialog</i>	A dialog to query user's preferences for updating object classes.
<i>EditObjectsDialog</i>	A dialog to query user's preferences for updating objects.
<i>EditRelationshipClassesDialog</i>	A dialog to query user's preferences for updating relationship classes.
<i>EditRelationshipsDialog</i>	A dialog to query user's preferences for updating relationships.
<i>RemoveEntitiesDialog</i>	A dialog to query user's preferences for removing tree items.

class spinetoolbox.spine_db_editor.widgets.edit_or_remove_items_dialogs.**EditOrRemoveItemsDialog**(*parent*,
db_mgr:

Bases: *spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ManageItemsDialog*

A dialog with a CopyPasteTableView and a QDialogButtonBox. Base class for all dialogs to query user's preferences for adding/editing/managing data items.

Init class.

Parameters

- **parent** (*SpineDBEditor*) – data store widget
- **db_mgr** (*SpineDBManager*) –

all_databases(*row*)

Returns a list of db names available for a given row. Used by delegates.

class spinetoolbox.spine_db_editor.widgets.edit_or_remove_items_dialogs.**EditObjectClassesDialog**(*parent*,
db_mgr,
se-
lected)

Bases: *spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ShowIconColorEditorMixin*, *EditOrRemoveItemsDialog*

A dialog to query user's preferences for updating object classes.

Init class.

Parameters

- **parent** (*SpineDBEditor*) – data store widget
- **db_mgr** (*SpineDBManager*) – the manager to do the update
- **selected** (*set*) – set of ObjectClassItem instances to edit

connect_signals()

Connect signals to slots.

all_db_maps(*row*)

Returns a list of db maps available for a given row. Used by ShowIconColorEditorMixin.

accept()

Collect info from dialog and try to update items.

class spinetoolbox.spine_db_editor.widgets.edit_or_remove_items_dialogs.**EditObjectsDialog**(*parent*,
db_mgr,
se-
lected)

Bases: *EditOrRemoveItemsDialog*

A dialog to query user's preferences for updating objects.

Init class.

Parameters

- **parent** (*SpineDBEditor*) – data store widget
- **db_mgr** (*SpineDBManager*) – the manager to do the update
- **selected** (*set*) – set of ObjectItem instances to edit

accept()

Collect info from dialog and try to update items.

class spinetoolbox.spine_db_editor.widgets.edit_or_remove_items_dialogs.**EditRelationshipClassesDialog**(*parent*,
db_mgr,
se-
lected)

Bases: *spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ShowIconColorEditorMixin*, *EditOrRemoveItemsDialog*

A dialog to query user's preferences for updating relationship classes.

Init class.

Parameters

- **parent** ([SpineDBEditor](#)) – data store widget
- **db_mgr** ([SpineDBManager](#)) – the manager to do the update
- **selected** (*set*) – set of RelationshipClassItem instances to edit

connect_signals()

Connect signals to slots.

accept()

Collect info from dialog and try to update items.

```
class spinetoolbox.spine_db_editor.widgets.edit_or_remove_items_dialogs.EditRelationshipsDialog(parent,
                                                                                               db_mgr,
                                                                                               se-
                                                                                               lected,
                                                                                               class_key
```

Bases: [spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetRelationshipClassesMixin](#), [spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetObjectsMixin](#), [EditOrRemoveItemsDialog](#)

A dialog to query user's preferences for updating relationships.

Init class.

Parameters

- **parent** ([SpineDBEditor](#)) – data store widget
- **db_mgr** ([SpineDBManager](#)) – the manager to do the update
- **selected** (*set*) – set of RelationshipItem instances to edit
- **class_key** (*tuple*) – (*class_name*, *object_class_name_list*) for identifying the relationship_class

accept()

Collect info from dialog and try to update items.

```
class spinetoolbox.spine_db_editor.widgets.edit_or_remove_items_dialogs.RemoveEntitiesDialog(parent,
                                                                                               db_mgr,
                                                                                               se-
                                                                                               lected)
```

Bases: [EditOrRemoveItemsDialog](#)

A dialog to query user's preferences for removing tree items.

Init class.

Parameters

- **parent** ([SpineDBEditor](#)) – data store widget
- **db_mgr** ([SpineDBManager](#)) – the manager to do the removal
- **selected** (*dict*) – maps item type (class) to instances

accept()

Collect info from dialog and try to remove items.

`spinetoolbox.spine_db_editor.widgets.graph_layout_generator`

Contains the `GraphLayoutGeneratorRunnable` class.

Module Contents

Classes

`ProgressBarWidget`

`GraphLayoutGeneratorRunnable`

Computes the layout for the Entity Graph View.

Functions

`make_heat_map`(`x`, `y`, `values`)

`spinetoolbox.spine_db_editor.widgets.graph_layout_generator.make_heat_map(x, y, values)`

class `spinetoolbox.spine_db_editor.widgets.graph_layout_generator.ProgressBarWidget`

Bases: `PySide6.QtWidgets.QWidget`

set_layout_generator(*layout_generator*)

paintEvent(*event*)

class `spinetoolbox.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGeneratorRunnable`(*identifier*,

ver-
tex_count,
src_inds=(
dst_inds=(
spread=0,
heavy_pos
max_iters=
weight_exp
2)

Bases: `PySide6.QtCore.QRunnable`

Computes the layout for the Entity Graph View.

class **Signals**

Bases: `PySide6.QtCore.QObject`

finished

layout_available

progressed

msg

```

stop(_checked=False)
set_show_previews(checked)
_is_stopped()
_layout_progressed(iteration)
_message_available(text)
_layout_available(x, y)
_preview_available(x, y)
run()

```

`spinetoolbox.spine_db_editor.widgets.graph_view_mixin`

Contains the GraphViewMixin class.

Module Contents

Classes

<code>GraphViewMixin</code>	Provides the graph view for the DS form.
-----------------------------	--

class `spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin(*args, **kwargs)`

Provides the graph view for the DS form.

VERTEX_EXTENT = 64

_ARC_WIDTH

_ARC_LENGTH_HINT

_renew_fetch_parents()

_stop_extending_graph(_=False)

init_models()

connect_signals()

Connects signals.

_refresh_icons(item_type, db_map_data)

Runs when entity classes are added or updated in the db. Refreshes icons of entities in graph.

Parameters

db_map_data (*dict*) – list of dictionary-items keyed by DiffDatabaseMapping instance.

_accepts_object_item(item, db_map)

_accepts_relationship_item(item, db_map)

`_handle_objects_added(db_map_data)`

Runs when objects are added to the db. Adds the new objects to the graph if needed.

Parameters

`db_map_data` (*dict*) – list of dictionary-items keyed by DiffDatabaseMapping instance.

`_handle_relationships_added(db_map_data)`

Runs when relationships are added to the db. Adds the new relationships to the graph if needed.

Parameters

`db_map_data` (*dict*) – list of dictionary-items keyed by DiffDatabaseMapping instance.

`_handle_objects_updated(db_map_data)`

Runs when objects are updated in the db. Refreshes names of objects in graph.

Parameters

`db_map_data` (*dict*) – list of dictionary-items keyed by DiffDatabaseMapping instance.

`_handle_objects_removed(db_map_data)`

Runs when objects are removed from the db. Rebuilds graph if needed.

Parameters

`db_map_data` (*dict*) – list of dictionary-items keyed by DiffDatabaseMapping instance.

`_handle_relationships_updated(db_map_data)`

Runs when relationships are updated in the db.

Parameters

`db_map_data` (*dict*) – list of dictionary-items keyed by DiffDatabaseMapping instance.

`_handle_relationships_removed(db_map_data)`

Runs when relationships are removed from the db. Rebuilds graph if needed.

Parameters

`db_map_data` (*dict*) – list of dictionary-items keyed by DiffDatabaseMapping instance.

`add_db_map_ids_to_items(db_map_data, type_)`

Goes through items of given type and adds the corresponding db_map ids. This could mean either restoring removed (db_map, id) tuples previously removed, or adding new (db_map, id) tuples.

Parameters

`db_map_data` (*dict(DiffDatabaseMapping, list)*) – List of added items keyed by db_map

Returns

tuples (db_map, id) that didn't match any item in the view.

Return type

list

`hide_removed_entities(db_map_data, type_)`

Hides removed entities while saving them into a set. This allows entities to be restored in case the user undoes the operation.

`_handle_entity_graph_visibility_changed(visible)`

`_handle_tree_selection_changed(selected)`

Stores the given selection of entity tree indexes and builds graph.

`rebuild_graph(_checked=False)`

build_graph(*persistent=False*)

Builds the graph.

Parameters

persistent (*bool, optional*) – If True, elements in the current graph (if any) retain their position in the new one.

_stop_layout_generators()

_complete_graph(*layout_gen_id, x, y*)

Parameters

- **layout_gen_id** (*object*) –
- **x** (*list*) – Horizontal coordinates
- **y** (*list*) – Vertical coordinates

_get_selected_entity_ids()

Returns a set of ids corresponding to selected entities in the trees.

Returns

selected object ids set: selected relationship ids

Return type

set

_get_db_map_relationships_for_graph(*db_map_object_ids, db_map_relationship_ids*)

_update_graph_data()

Updates data for graph according to selection in trees.

_get_object_key(*db_map_object_id*)

_get_relationship_key(*db_map_relationship_id*)

_update_src_dst_inds(*db_map_object_id_lists*)

_get_parameter_positions(*parameter_name*)

_make_layout_generator()

Returns a layout generator for the current graph.

Returns

GraphLayoutGeneratorRunnable

_make_new_items(*x, y*)

Returns new items for the graph.

Parameters

- **x** (*list*) –
- **y** (*list*) –

_add_new_items()

start_relationship(*db_map, relationship_class, obj_item*)

Starts a relationship from the given object item.

Parameters

- **db_map** (*DiffDatabaseMapping*) –

- `relationship_class` (*dict*) –
- `obj_item` (*..graphics_items.ObjectItem*) –

`finalize_relationship(relationship_class, *object_items)`
Tries to add relationships between the given object items.

Parameters

- `relationship_class` (*dict*) –
- `object_items` (*..graphics_items.ObjectItem*) –

`_begin_add_relationships()`
`_end_add_relationships()`
`add_objects_at_position(pos)`
`get_pdf_file_path()`
`closeEvent(event)`
Handle close window.

Parameters

- `event` (*QCloseEvent*) – Closing event

`spinetoolbox.spine_db_editor.widgets.item_metadata_editor`

Contains machinery to deal with item metadata editor.

Module Contents

Classes

<code>ItemMetadataEditor</code>	A DB editor helper class that manages entity and parameter value metadata editor.
---------------------------------	---

`class spinetoolbox.spine_db_editor.widgets.item_metadata_editor.ItemMetadataEditor`(*item_metadata_table_view*,
db_editor,
meta-
data_editor,
db_mgr)

A DB editor helper class that manages entity and parameter value metadata editor.

Parameters

- `item_metadata_table_view` (*ItemMetadataTableView*) – editor’s view
- `db_editor` (*SpineDBEditor*) – database editor
- `metadata_editor` (*MetadataEditor*) – metadata editor
- `db_mgr` (*SpineDBManager*) – database manager

connect_signals(ui)

Connects user interface signals.

Parameters

ui (*Ui_MainWindow*) – DB editor’s user interface

init_models(db_maps)

Initializes editor’s models.

Parameters

db_maps (*Iterable of DiffDatabaseMapping*) – database mappings

_reload_entity_metadata(current_index, previous_index)

Loads entity metadata for selected object or relationship.

Parameters

- **current_index** (*QModelIndex*) – currently selected index in object/relationship tree
- **previous_index** (*QModelIndex*) – unused

_reload_value_metadata(current_index, previous_index)

Loads parameter value metadata for selected value.

Parameters

- **current_index** (*QModelIndex*) – currently selected index in object/relationship parameter value table
- **previous_index** (*QModelIndex*) – unused

rollback(db_maps)

Rolls back database changes.

Parameters

db_maps (*Iterable of DiffDatabaseMapping*) – rolled back databases

spinetoolbox.spine_db_editor.widgets.manage_items_dialogs

Classes for custom QDialogs to add edit and remove database items.

Module Contents**Classes**

<i>ManageItemsDialogBase</i>	Init class.
<i>ManageItemsDialog</i>	A dialog with a CopyPasteTableView and a QDialogButtonBox. Base class for all
<i>GetObjectClassesMixin</i>	Provides a method to retrieve object classes for AddObjectsDialog and AddRelationshipClassesDialog.
<i>GetObjectsMixin</i>	Provides a method to retrieve objects for AddRelationshipsDialog and EditRelationshipsDialog.
<i>GetRelationshipClassesMixin</i>	Provides a method to retrieve relationship classes for AddRelationshipsDialog and EditRelationshipsDialog.
<i>ShowIconColorEditorMixin</i>	Provides methods to show an <i>IconColorEditor</i> upon request.

```
class spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ManageItemsDialogBase(parent,
                                                                 db_mgr)
```

Bases: PySide6.QtWidgets.QDialog

Init class.

Parameters

- **parent** ([SpineDBEditor](#)) – data store widget
- **db_mgr** ([SpineDBManager](#)) –

make_table_view()

connect_signals()

Connect signals to slots.

resize_window_to_columns(*height=None*)

```
class spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ManageItemsDialog(parent,
                                                                 db_mgr)
```

Bases: [ManageItemsDialogBase](#)

A dialog with a CopyPasteTableView and a QDialogButtonBox. Base class for all dialogs to query user's preferences for adding/editing/managing data items.

Init class.

Parameters

- **parent** ([SpineDBEditor](#)) – data store widget
- **db_mgr** ([SpineDBManager](#)) –

connect_signals()

Connect signals to slots.

_handle_model_data_changed(*top_left, bottom_right, roles*)

Reimplement in subclasses to handle changes in model data.

set_model_data(*index, data*)

Update model data.

_handle_model_reset()

Resize columns and form.

```
class spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetObjectClassesMixin
```

Provides a method to retrieve object classes for AddObjectsDialog and AddRelationshipClassesDialog.

make_db_map_obj_cls_lookup()

object_class_name_list(*row*)

Return a list of object_class names present in all databases selected for given row. Used by *ManageObjectsDelegate*.

```
class spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetObjectsMixin
```

Provides a method to retrieve objects for AddRelationshipsDialog and EditRelationshipsDialog.

make_db_map_obj_lookup()

object_name_list(row, column)

Return a list of object names present in all databases selected for given row. Used by *ManageRelationshipsDelegate*.

class

`spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetRelationshipClassesMixin`

Provides a method to retrieve relationship classes for *AddRelationshipsDialog* and *EditRelationshipsDialog*.

make_db_map_rel_cls_lookup()

class `spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ShowIconColorEditorMixin`

Provides methods to show an *IconColorEditor* upon request.

show_icon_color_editor(index)

`spinetoolbox.spine_db_editor.widgets.mass_select_items_dialogs`

Classes for custom QDialogs to add edit and remove database items.

Module Contents

Classes

<code>_SelectDatabases</code>	A widget that shows checkboxes for each database.
<code>MassSelectItemsDialog</code>	A dialog to query a selection of dbs and items from the user.
<code>MassRemoveItemsDialog</code>	A dialog to query user's preferences for mass removing db items.
<code>MassExportItemsDialog</code>	A dialog to let users chose items for JSON export.

class `spinetoolbox.spine_db_editor.widgets.mass_select_items_dialogs._SelectDatabases`(db_maps, checked_states, parent)

Bases: `PySide6.QtWidgets.QWidget`

A widget that shows checkboxes for each database.

Parameters

- **db_maps** (tuple of *DatabaseMappingBase*) – database maps
- **checked_states** (dict, optional) – mapping from item name to check state boolean
- **parent** (*QWidget*) – parent widget

checked_state_changed

checked_states()

Collects the checked states of databases.

Returns

mapping from database mapping to checked state boolean

Return type

dict

any_checked()

Checks if any of the checkboxes is checked.

Returns

True if any check box is checked, False otherwise

Return type

bool

```
class spinetoolbox.spine_db_editor.widgets.mass_select_items_dialogs.MassSelectItemsDialog(parent,
                                                                 db_mgr,
                                                                 *db_maps,
                                                                 stored_state,
                                                                 ok_button_text)
```

Bases: [spinetoolbox.widgets.custom_qwidgets.SelectDatabaseItemsDialog](#)

A dialog to query a selection of dbs and items from the user.

Parameters

- **parent** ([SpineDBEditor](#)) – parent widget
- **db_mgr** ([SpineDBManager](#)) – database manager
- ***db_maps** – the dbs to select items from
- **stored_state** (*dict*, *Optional*) – widget’s previous state
- **ok_button_text** (*str*, *optional*) – alternative label for the OK button

state_storing_requested

_handle_check_box_state_changed(_checked)

Enables or disables the OK button.

accept()

```
class spinetoolbox.spine_db_editor.widgets.mass_select_items_dialogs.MassRemoveItemsDialog(parent,
                                                                 db_mgr,
                                                                 *db_maps,
                                                                 stored_state=None)
```

Bases: [MassSelectItemsDialog](#)

A dialog to query user’s preferences for mass removing db items.

Initialize class.

Parameters

- **parent** ([SpineDBEditor](#)) –
- **db_mgr** ([SpineDBManager](#)) –
- **db_maps** (*DiffDatabaseMapping*) – the dbs to select items from
- **stored_state** (*dict*, *Optional*) – widget’s previous state

accept()

```
class spinetoolbox.spine_db_editor.widgets.mass_select_items_dialogs.MassExportItemsDialog(parent,
                                                                                       db_mgr,
                                                                                       *db_maps,
                                                                                       stored_state=None)
```

Bases: *MassSelectItemsDialog*

A dialog to let users chose items for JSON export.

Parameters

- **parent** (*SpineDBEditor*) –
- **db_mgr** (*SpineDBManager*) –
- **db_maps** (*DiffDatabaseMapping*) – the dbs to select items from
- **stored_state** (*dict*, *Optional*) – widget’s previous state

_warn_checked_non_data_items = **False**

data_submitted

accept()

```
spinetoolbox.spine_db_editor.widgets.metadata_editor
```

Contains machinery to deal with metadata editor.

Module Contents

Classes

MetadataEditor

A DB editor helper class that manages metadata editor.

```
class spinetoolbox.spine_db_editor.widgets.metadata_editor.MetadataEditor(metadata_table_view,
                                                                           db_editor,
                                                                           db_mgr)
```

A DB editor helper class that manages metadata editor.

Parameters

- **metadata_table_view** (*MetadataTableView*) – editor’s view
- **db_editor** (*SpineDBEditor*) – database editor
- **db_mgr** (*SpineDBManager*) – database manager

connect_signals(*ui*)

Connects user interface signals.

Parameters

- **ui** (*Ui_MainWindow*) – DB editor’s user interface

init_models(*db_maps*)

Initializes editor’s models.

Parameters

db_maps (*Iterable of DiffDatabaseMapping*) – database mappings

metadata_model()

Returns metadata model.

Returns

model

Return type

MetadataModel

rollback(db_maps)

Rolls back database changes.

Parameters

db_maps (*Iterable of DiffDatabaseMapping*) – rolled back databases

spinetoolbox.spine_db_editor.widgets.multi_spine_db_editor

Contains the MultiSpineDBEditor class.

Module Contents

Classes

<i>MultiSpineDBEditor</i>	Database editor's tabbed main window.
<i>_CustomStatusBar</i>	

class spinetoolbox.spine_db_editor.widgets.multi_spine_db_editor.**MultiSpineDBEditor**(*db_mgr*,
db_url_codenames=None)

Bases: *spinetoolbox.widgets.multi_tab_window.MultiTabWindow*

Database editor's tabbed main window.

Parameters

- **db_mgr** (*SpineDBManager*) – database manager
- **db_url_codenames** (*dict, optional*) – mapping from database URL to its codename

_make_other()

Creates a new MultiTabWindow of this type.

Returns

new MultiTabWindow

Return type

MultiTabWindow

_connect_tab_signals(tab)

Connects Spine Db editor window (tab) signals.

Parameters

tab (*SpineDBEditor*) – Spine Db editor window

Returns

True if ok, False otherwise

Return type

bool

`_disconnect_tab_signals(index)`

Disconnects signals of Spine Db editor window (tab) in given index.

Parameters

index (*int*) – Tab index

Returns

True if ok, False otherwise

Return type

bool

`_make_new_tab(db_url_codenames=None)`

Creates a new tab.

Parameters

- ***args** – positional arguments needed to make a new tab
- ****kwargs** – keyword arguments needed to make a new tab

`show_plus_button_context_menu(global_pos)`

Opens a context menu for the tool bar.

Parameters

global_pos (*QPoint*) – menu position on screen

`make_context_menu(index)`

Creates a context menu for given tab.

Parameters

index (*int*) – tab index

Returns

context menu or None if tab was not found

Return type

QMenu

`_insert_statusbar_button(button)`

Inserts given button to the ‘beginning’ of the status bar and decorates it with a shooting label.

Parameters

button (*OpenFileButton*) –

`insert_sqlite_file_open_button(file_path)`**`insert_file_open_button(file_path)`****`_open_sqlite_url(url, codename)`**

Opens sqlite url.

`show_user_guide(checked=False)`

Opens Spine db editor documentation page in browser.

`_show_waiting_for_fetcher()`

Shows a message box to user informing that a tab is waiting for fetcher to finish working.

`_close_waiting_for_fetcher()`

class `spinetoolbox.spine_db_editor.widgets.multi_spine_db_editor._CustomStatusBar`(*parent=None*)

Bases: `PySide6.QtWidgets.QStatusBar`

`spinetoolbox.spine_db_editor.widgets.object_name_list_editor`

Contains the `ObjectNameListEditor` class.

Module Contents

Classes

<code>SearchBarDelegate</code>	A custom delegate to use with <code>ObjectNameListEditor</code> .
<code>ObjectNameListEditor</code>	A dialog to select the object name list for a relationship using Google-like search bars.

class `spinetoolbox.spine_db_editor.widgets.object_name_list_editor.SearchBarDelegate`

Bases: `PySide6.QtWidgets.QItemDelegate`

A custom delegate to use with `ObjectNameListEditor`.

data_committed

setModelData(*editor, model, index*)

createEditor(*parent, option, index*)

updateEditorGeometry(*editor, option, index*)

close_editor(*editor, index, model*)

eventFilter(*editor, event*)

class `spinetoolbox.spine_db_editor.widgets.object_name_list_editor.ObjectNameListEditor`(*parent, index, object_class_names, object_names_lists, current_object_names*)

Bases: `spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ManageItemsDialog`

A dialog to select the object name list for a relationship using Google-like search bars.

Initializes widget.

Parameters

- **parent** (`SpineDBEditor`) –

- **index** (*QModelIndex*) –
- **object_class_names** (*list*) – string object_class names
- **object_names_lists** (*list*) – lists of string object names
- **current_object_names** (*list*) –

init_model(*object_class_names, object_names_lists, current_object_names*)

accept()

`spinetoolbox.spine_db_editor.widgets.parameter_view_mixin`

Contains the ParameterViewMixin class.

Module Contents

Classes

<i>ParameterViewMixin</i>	Provides stacked parameter tables for the Spine db editor.
---------------------------	--

class `spinetoolbox.spine_db_editor.widgets.parameter_view_mixin.ParameterViewMixin`(*args,
**kwargs)

Provides stacked parameter tables for the Spine db editor.

connect_signals()

Connects signals to slots.

init_models()

Initializes models.

show_object_name_list_editor(*index, rel_cls_id, db_map*)

Shows the object names list editor.

Parameters

- **index** (*QModelIndex*) –
- **rel_cls_id** (*int*) –
- **db_map** (*DiffDatabaseMapping*) –

_set_default_parameter_data(*index=None*)

Sets default rows for parameter models according to given index.

Parameters

- index** (*QModelIndex*) – and index of the object or relationship tree

set_default_parameter_data(*default_data, default_db_map*)

clear_all_filters()

_reset_filters()

Resets filters.

`_handle_graph_selection_changed(selected_items)`

Resets filter according to graph selection.

`_handle_object_tree_selection_changed(selected_indexes)`

Resets filter according to object tree selection.

`_handle_relationship_tree_selection_changed(selected_indexes)`

Resets filter according to relationship tree selection.

`_handle_entity_tree_selection_changed(selected_indexes, object_tree)`

Combines object and relationship selections from object and relationship tree views.

Parameters

- **`selected_indexes`** (*dict*) – mapping from database map to set of alternative ids
- **`object_tree`** (*bool*) – if True, the selection was made in the Object tree, else in the Relationship tree

`_handle_alternative_selection_changed(selected_db_map_alt_ids)`

Resets filter according to selection in alternative tree view.

`_handle_scenario_alternative_selection_changed(selected_db_map_alt_ids)`

Resets filter according to selection in scenario tree view.

`_update_alternative_selection(selected_db_map_alt_ids, other_tree_view, this_tree_view)`

Combines alternative selections from alternative and scenario tree views.

Parameters

- **`selected_db_map_alt_ids`** (*dict*) – mapping from database map to set of alternative ids
- **`other_tree_view`** (*AlternativeTreeView or ScenarioTreeView*) – tree view whose selection didn't change
- **`this_tree_view`** (*AlternativeTreeView or ScenarioTreeView*) – tree view whose selection changed

`_clear_all_other_selections(current, other=None)`

Clears all the other selections besides the one that was just made.

Parameters

- **`current`** – the tree where the selection that was just made
- **`other`** (*optional*) – other optional tree

`static _dict_intersection(dict1, dict2)`

Creates a dictionary from two dicts that is either their union or intersection based on their keys.

spinetoolbox.spine_db_editor.widgets.pivot_table_header_view

Contains custom QHeaderView for the pivot table.

Module Contents**Classes**

<i>PivotTableHeaderView</i>	Header view for the pivot table.
<i>ParameterValuePivotHeaderView</i>	Header view for the pivot table in parameter value and index expansion mode.
<i>ScenarioAlternativePivotHeaderView</i>	Header view for the pivot table in parameter value and index expansion mode.

class spinetoolbox.spine_db_editor.widgets.pivot_table_header_view.**PivotTableHeaderView**(*orientation*,
area,
pivot_table_view)

Bases: PySide6.QtWidgets.QHeaderView

Header view for the pivot table.

Parameters

- **orientation** (*Qt.Orientation*) – Qt.Orientation.Horizontal or Qt.Orientation.Vertical
- **area** (*str*) – which pivot area the header represents: “columns”, “rows” or “frozen”
- **pivot_table_view** (*PivotTableView*) – parent view

property *area*

header_dropped

dragEnterEvent (*event*)

dragMoveEvent (*event*)

dropEvent (*event*)

class spinetoolbox.spine_db_editor.widgets.pivot_table_header_view.**ParameterValuePivotHeaderView**(*orientation*,
area,
pivot_ta)

Bases: *PivotTableHeaderView*

Header view for the pivot table in parameter value and index expansion mode.

Parameters

- **orientation** (*Qt.Orientation*) – Qt.Orientation.Horizontal or Qt.Orientation.Vertical
- **area** (*str*) – which pivot area the header represents: “columns”, “rows” or “frozen”
- **pivot_table_view** (*PivotTableView*) – parent view

_column_selection()

Lists current column’s indexes that contain some data.

Returns

column indexes

Return type

list of QModelIndex

_add_column_to_plot(*action*)

Adds a single column to existing plot window.

_plot_column()

Plots a single column not the selection.

_column_indexes(*column*)

Makes indexes for given column.

Parameters

column (*int*) – column

Returns

column indexes

Return type

list of QModelIndex

_set_x_flag()

Sets the X flag for a column.

contextMenuEvent(*event*)

Shows context menu.

Parameters

event (*QContextMenuEvent*) –

class spinetoolbox.spine_db_editor.widgets.pivot_table_header_view.ScenarioAlternativePivotHeaderView(*orientation*, *area*, *parent*)

Bases: [PivotTableHeaderView](#)

Header view for the pivot table in parameter value and index expansion mode.

Parameters

- **orientation** (*Qt.Orientation*) – Qt.Orientation.Horizontal or Qt.Orientation.Vertical
- **area** (*str*) – which pivot area the header represents: “columns”, “rows” or “frozen”
- **pivot_table_view** ([PivotTableView](#)) – parent view

context_menu_requested

Requests a header context menu be shown at given global position.

contextMenuEvent(*event*)

spinetoolbox.spine_db_editor.widgets.scenario_generator

Contains a dialog for generating scenarios from selected alternatives.

Module Contents**Classes**

<code>_ScenarioNameResolution</code>	Generic enumeration.
<code>ScenarioGenerator</code>	A dialog where users can generate scenarios from given alternatives.

Functions

<code>_ensure_unique(scenario_alternatives)</code>	Removes duplicate scenario alternatives.
<code>_find_base_alternative(names)</code>	Returns the name of a 'base' alternative or empty string if not found.
<code>_suffix(item_count)</code>	Returns a formattable string with enough zero padding to hold item_count digits.

class spinetoolbox.spine_db_editor.widgets.scenario_generator._ScenarioNameResolution

Bases: enum.Enum

Generic enumeration.

Derive from this class to define new enumerations.

NO_CONFLICT

OVERWRITE

LEAVE_AS_IS

CANCEL_OPERATION

class spinetoolbox.spine_db_editor.widgets.scenario_generator.ScenarioGenerator(*parent*,
db_map,
alternatives,
spine_db_editor)

Bases: PySide6.QtWidgets.QWidget

A dialog where users can generate scenarios from given alternatives.

Parameters

- **parent** (*QWidget*) – parent widget
- **db_map** (*DiffDatabaseMapping*) – database mapping that contains the alternatives
- **alternatives** (*Iterable of CacheItem*) – alternatives from which the scenarios are generated
- **spine_db_editor** (*SpineDBEditor*) – database editor instance

_TYPE_LABELS = ('All combinations', 'Scenario for each alternative')

accept()

Generates scenarios and closes the dialog.

The operation may get cancelled by user if there are conflicts in scenario names.

_generate_scenarios(*new_scenarios*, *scenarios_to_modify*, *scenario_alternatives*)

Generates scenarios with all possible combinations of given alternatives.

Parameters

- **new_scenarios** (*Iterable of str*) – names of new scenarios to create
- **scenarios_to_modify** (*Iterable of str*) – names of scenarios to modify
- **scenario_alternatives** (*list of list*) – alternative items for each scenario

_check_existing_scenarios(*proposed_scenario_names*, *existing_scenario_names*)

Checks if proposed scenarios exist, and if so, prompts users what to do.

Parameters

- **proposed_scenario_names** (*Iterable of str*) – proposed scenario names
- **existing_scenario_names** (*set of str*) – existing scenario names

Returns

action to take

Return type

_ScenarioNameResolution

_enable_base_alternative(*check_box_state*)

Enables and disables base alternative combo box.

Parameters

check_box_state (*int*) – state of ‘Use base alternative’ check box

_insert_base_alternative(*scenario_alternatives*)

Prepends base alternative to scenario alternatives if it has been enabled.

If base alternative is already in scenario alternatives, make sure it comes first.

Parameters

scenario_alternatives (*list of list*) – scenario alternatives

`spinetoolbox.spine_db_editor.widgets.scenario_generator._ensure_unique`(*scenario_alternatives*)

Removes duplicate scenario alternatives.

Parameters

scenario_alternatives (*list of list*) – scenario alternatives

`spinetoolbox.spine_db_editor.widgets.scenario_generator._find_base_alternative`(*names*)

Returns the name of a ‘base’ alternative or empty string if not found.

Basically, checks if “Base” is in names, otherwise searches for the first case-insensitive version of “base”.

Parameters

names (*list of str*) – alternative names

Returns

base alternative name

Return type

str

`spinetoolbox.spine_db_editor.widgets.scenario_generator._suffix(item_count)`

Returns a formattable string with enough zero padding to hold item_count digits.

Parameters

item_count (*int*) – maximum number of items

Returns

string in the form ‘{:0n}’ where n is the number of digits in item_count

Return type

str

`spinetoolbox.spine_db_editor.widgets.select_position_parameters_dialog`

Classes for custom QDialogs to add items to databases.

Module Contents

Classes

SelectPositionParametersDialog

ParameterNameDelegate

A delegate for the database name.

class `spinetoolbox.spine_db_editor.widgets.select_position_parameters_dialog.SelectPositionParametersDialog`

Bases: `PySide6.QtWidgets.QDialog`

selection_made

accept()

_parameter_position_x()

_parameter_position_y()

class `spinetoolbox.spine_db_editor.widgets.select_position_parameters_dialog.ParameterNameDelegate`(parent, db_model, *db_items)

Bases: `PySide6.QtWidgets.QStyledItemDelegate`

A delegate for the database name.

setModelData(editor, model, index)

Send signal.

setEditorData(editor, index)

Do nothing. We’re setting editor data right away in createEditor.

updateEditorGeometry(editor, option, index)

_close_editor(*editor, index*)

Closes editor. Needed by SearchBarEditor.

createEditor(*parent, option, index*)

Returns editor.

`spinetoolbox.spine_db_editor.widgets.spine_db_editor`

Contains the SpineDBEditor class.

Module Contents

Classes

<i>SpineDBEditorBase</i>	Base class for SpineDBEditor (i.e. Spine database editor).
<i>SpineDBEditor</i>	A widget to visualize Spine dbs.

class `spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase`(*db_mgr*)

Bases: `PySide6.QtWidgets.QMainWindow`

Base class for SpineDBEditor (i.e. Spine database editor).

Parameters

db_mgr (`SpineDBManager`) – The manager to use

property `toolbox`

property `settings_subgroup`

property `db_names`

property `first_db_map`

property `db_url_codenames`

msg

msg_error

file_exported

sqlite_file_exported

static `is_db_map_editor()`

Always returns True as SpineDBEditors are truly database editors.

Unless, of course, the database can one day be opened in read-only mode. In that case this method should return False.

Returns

Always True

Return type

bool

`load_db_urls(db_url_codenames, create=False, update_history=True)`

`init_add_undo_redo_actions()`

`load_previous_urls(_=False)`

`load_next_urls(_=False)`

`open_db_file(_=False)`

`add_db_file(_=False)`

`create_db_file(_=False)`

`_make_docks_menu()`

Returns a menu with all dock toggle/view actions. Called by `self.add_main_menu()`.

Returns

QMenu

`add_main_menu()`

Adds a menu with main actions to toolbar.

`_browse_commits()`

`connect_signals()`

Connects signals to slots.

`vacuum(_checked=False)`

`update_undo_redo_actions(_)`

`_replace_undo_redo_actions(new_undo_action, new_redo_action)`

`_refresh_undo_redo_actions()`

`update_commit_enabled(_clean=False)`

`init_models()`

Initializes models.

`add_message(msg)`

Pushes message to notification stack.

Parameters

msg (str) – String to show in the notification

`refresh_copy_paste_actions()`

Runs when menus are about to show. Enables or disables actions according to selection status.

`copy(checked=False)`

Copies data to clipboard.

`paste(checked=False)`

Pastes data from clipboard.

`import_data(data)`

`import_file(checked=False)`

Import file. It supports SQLite, JSON, and Excel.

import_from_json(*file_path*)

import_from_sqlite(*file_path*)

import_from_excel(*file_path*)

show_mass_export_items_dialog(*checked=False*)

Shows dialog for user to select dbs and items for export.

_store_export_settings(*state*)

Stores export items dialog settings.

_clean_up_export_items_dialog()

Cleans up export items dialog.

export_session(*checked=False*)

Exports changes made in the current session as reported by DiffDatabaseMapping.

mass_export_items(*db_map_item_types*)

duplicate_object(*object_item*)

Duplicates an object.

Parameters

object_item (*ObjectTreeItem* of *ObjectItem*) –

duplicate_scenario(*db_map, scen_id*)

Duplicates a scenario.

Parameters

- **db_map** (*DiffDatabaseMapping*) –
- **scen_id** (*int*) –

export_data(*db_map_ids_for_export*)

Exports data from given dictionary into a file.

Parameters

db_map_ids_for_export – Dictionary mapping db maps to keyword arguments for `spinedb_api.export_data`

refresh_session(*checked=False*)

commit_session(*checked=False*)

Commits dirty database maps.

rollback_session(*checked=False*)

Rolls back dirty database maps.

receive_session_committed(*db_maps, cookie*)

receive_session_rolled_back(*db_maps*)

receive_session_refreshed(*db_maps*)

show_mass_remove_items_form(*checked=False*)

Opens the purge items dialog.

_store_purge_settings(*state*)

Stores Purge items dialog state.

Parameters

state (*dict*) – dialog state

_clean_up_purge_items_dialog()

Removes references to purge items dialog.

show_parameter_value_editor(*index*, *plain=False*)

Shows the parameter_value editor for the given index of given table view.

receive_error_msg(*db_map_error_log*)

_update_export_enabled()

Update export enabled.

_log_items_change(*msg*)

Enables or disables actions and informs the user about what just happened.

_handle_items_added(*item_type*, *db_map_data*)

_handle_items_updated(*item_type*, *db_map_data*)

_handle_items_removed(*item_type*, *db_map_data*)

restore_ui()

Restore UI state from previous session.

save_window_state()

Save window state parameters (size, position, state) via QSettings.

tear_down()

Performs clean up duties.

Returns

True if editor is ready to close, False otherwise

Return type

bool

_prompt_to_commit_changes()

Prompts the user to commit or rollback changes to ‘dirty’ db maps.

Returns

QMessageBox status code

Return type

int

_get_commit_msg(*db_names*)

Prompts user for commit message.

Parameters

db_names (*Iterable of str*) – database names

Returns

commit message

Return type

str

_get_rollback_confirmation(*db_names*)

Prompts user for confirmation before rolling back the session.

Parameters

db_names (*Iterable of str*) – database names

Returns

True if user confirmed, False otherwise

Return type

bool

_purge_change_notifiers()

Tears down change notifiers.

closeEvent(*event*)

Handle close window.

Parameters

event (*QCloseEvent*) – Closing event

static _get_base_dir()

class spinetoolbox.spine_db_editor.widgets.spine_db_editor.**SpineDBEditor**(*db_mgr*,
db_url_codenames=None)

Bases: *spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin*,
spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin, *spinetoolbox.spine_db_editor.widgets.parameter_view_mixin.ParameterViewMixin*, *spinetoolbox.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin*, *SpineDBEditorBase*

A widget to visualize Spine dbs.

Initializes everything.

Parameters

db_mgr (*SpineDBManager*) – The manager to use

pinned_values_updated

emit_pinned_values_updated()

connect_signals()

Connects signals to slots.

init_models()

Initializes models.

_restart_timer_refresh_tab_order(*_visible=False*)

_refresh_tab_order()

tabify_and_raise(*docks*)

Tabifies docks in given list, then raises the first.

Parameters

docks (*list*) –

restore_dock_widgets()

Docks all floating and or hidden QDockWidgets back to the window.

begin_style_change()

Begins a style change operation.

end_style_change()

Ends a style change operation.

apply_stacked_style(_checked=False)

Applies the stacked style, inspired in the former tree view.

apply_pivot_style(_checked=False)

Applies the pivot style, inspired in the former tabular view.

apply_graph_style(_checked=False)

Applies the graph style, inspired in the former graph view.

receive_session_rolled_back(db_maps)

Reacts to session rolled back event.

tear_down()

Performs clean up duties.

Returns

True if editor is ready to close, False otherwise

Return type

bool

spinetoolbox.spine_db_editor.widgets.tabular_view_header_widget

Contains TabularViewHeaderWidget class.

Module Contents**Classes**

TabularViewHeaderWidget

A draggable QWidget.

```
class spinetoolbox.spine_db_editor.widgets.tabular_view_header_widget.TabularViewHeaderWidget(identifier,
                                                                 area,
                                                                 menu=None,
                                                                 parent=None)
```

Bases: PySide6.QtWidgets.QFrame

A draggable QWidget.

Parameters

- **identifier** (*str*) –
- **area** (*str*) – either “rows”, “columns”, or “frozen”
- **menu** (*FilterMenu*, *optional*) –
- **parent** (*QWidget*, *optional*) – Parent widget

property identifier

property area

header_dropped

_H_MARGIN = 3

_SPACING = 16

mousePressEvent(*event*)

Register drag start position

mouseMoveEvent(*event*)

Start dragging action if needed

mouseReleaseEvent(*event*)

Forget drag start position

dragEnterEvent(*event*)

dropEvent(*event*)

`spinetoolbox.spine_db_editor.widgets.tabular_view_mixin`

Contains TabularViewMixin class.

Module Contents

Classes

TabularViewMixin

Provides the pivot table and its frozen table for the Database editor.

class `spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin`(*args,
**kwargs)

Provides the pivot table and its frozen table for the Database editor.

property current_object_class_id_list

property current_object_class_name_list

property current_object_class_ids

_PARAMETER_VALUE = '&Value'

_INDEX_EXPANSION = '&Index'

_RELATIONSHIP = 'Re&lationship'

_SCENARIO_ALTERNATIVE = '&Scenario'

_PARAMETER = 'parameter'

```

_ALTERNATIVE = 'alternative'

_INDEX = 'index'

populate_pivot_action_group()

connect_signals()
    Connects signals to slots.

refresh_views()

update_filter_menus(action)

_needs_to_update_headers(item_type, db_map_data)

_reload_pivot_table_if_needed(item_type, db_map_data)

_connect_pivot_table_header_signals()
    Connects signals of pivot table's header views.

init_models()
    Initializes models.

_set_model_data(index, value)

static _is_class_index(index)
    Returns whether the given tree index is a class index.

    Parameters
        index (QModelIndex) – index from object or relationship tree

    Returns
        bool

_handle_pivot_action_triggered(action)

_handle_pivot_table_visibility_changed(visible)

_handle_object_tree_selection_changed(selected_indexes)

_handle_relationship_tree_selection_changed(selected_indexes)

_handle_entity_tree_current_changed(current_index)

_update_class_attributes(current_index)
    Updates current class (type and id) and reloads pivot table for it.

static _get_current_class_item(current_index)

static _make_get_id(action)
    Returns a function to compute the db_map-id tuple of an item.

_get_db_map_entities()
    Returns a dict mapping db maps to a list of dict entity items in the current class.

    Returns
        dict

```

load_empty_relationship_data(*db_map_class_objects=None*)

Returns a dict containing all possible relationships in the current class.

Parameters

db_map_class_objects (*dict*) –

Returns

Key is db_map-object_id tuple, value is None.

Return type

dict

load_full_relationship_data(*db_map_relationships=None, action='add'*)

Returns a dict of relationships in the current class.

Parameters

db_map_relationships (*dict*) –

Returns

Key is db_map-object id tuple, value is relationship id.

Return type

dict

load_relationship_data()

Returns a dict that merges empty and full relationship data.

Returns

Key is object id tuple, value is True if a relationship exists, False otherwise.

Return type

dict

load_scenario_alternative_data(*db_map_scenarios=None, db_map_alternatives=None*)

Returns a dict containing all scenario alternatives.

Returns

Key is db_map-id tuple, value is None or rank.

Return type

dict

_get_db_map_parameter_value_or_def_ids(*item_type*)

Returns a dict mapping db maps to a list of integer parameter (value or def) ids from the current class.

Parameters

item_type (*str*) – either “parameter_value” or “parameter_definition”

Returns

dict

_get_db_map_parameter_values_or_defs(*item_type*)

Returns a dict mapping db maps to list of dict parameter (value or def) items from the current class.

Parameters

item_type (*str*) – either “parameter_value” or “parameter_definition”

Returns

dict

load_empty_parameter_value_data(*db_map_entities=None, db_map_parameter_ids=None, db_map_alternative_ids=None*)

Returns a dict containing all possible combinations of entities and parameters for the current class in all db_maps.

Parameters

- **db_map_entities** (*dict, optional*) – if given, only load data for these db maps and entities
- **db_map_parameter_ids** (*dict, optional*) – if given, only load data for these db maps and parameter definitions
- **db_map_alternative_ids** (*dict, optional*) – if given, only load data for these db maps and alternatives

Returns

Key is a tuple object_id, ..., parameter_id, value is None.

Return type

dict

load_full_parameter_value_data(*db_map_parameter_values=None, action='add'*)

Returns a dict of parameter values for the current class.

Parameters

- **db_map_parameter_values** (*list, optional*) –
- **action** (*str*) –

Returns

Key is a tuple object_id, ..., parameter_id, value is the parameter_value.

Return type

dict

_indexes(*value*)

load_empty_expanded_parameter_value_data(*db_map_entities=None, db_map_parameter_ids=None, db_map_alternative_ids=None*)

Makes a dict of expanded parameter values for the current class.

Parameters

- **db_map_parameter_values** (*list, optional*) –
- **action** (*str*) –

Returns

mapping from unique value id tuple to value tuple

Return type

dict

load_full_expanded_parameter_value_data(*db_map_parameter_values=None, action='add'*)

Makes a dict of expanded parameter values for the current class.

Parameters

- **db_map_parameter_values** (*list, optional*) –
- **action** (*str*) –

Returns

mapping from unique value id tuple to value tuple

Return type

dict

load_parameter_value_data()

Returns a dict that merges empty and full parameter_value data.

Returns

Key is a tuple object_id, ..., parameter_id, value is the parameter_value or None if not specified.

Return type

dict

load_expanded_parameter_value_data()

Returns all permutations of entities as well as parameter indexes and values for the current class.

Returns

Key is a tuple object_id, ..., index, while value is None.

Return type

dict

get_pivot_preferences()

Returns saved pivot preferences.

Returns

pivot tuple, or None if no preference stored

Return type

tuple, NoneType

do_reload_pivot_table()

Reloads pivot table.

_can_build_pivot_table()

clear_pivot_table()

wipe_out_filter_menus()

make_pivot_headers()

Turns top left indexes in the pivot table into TabularViewHeaderWidget.

_resize_pivot_header_columns()

_make_inserted_frozen_headers(*parent_index*, *first_column*, *last_column*)

Turns the first row of columns in the frozen table into TabularViewHeaderWidgets.

Parameters

- **parent_index** (*QModelIndex*) – frozen table column's parent index
- **first_column** (*int*) – first inserted column
- **last_column** (*int*) – last inserted column

_make_all_frozen_headers()

Turns the first row of columns in the frozen table into TabularViewHeaderWidgets.

_make_frozen_headers(*first_column, last_column*)

_check_frozen_value_selected(*parent, first_row, last_row*)

Ensures that at least one row is selected in frozen table when number of rows change.

create_filter_menu(*identifier*)

Returns a filter menu for given object_class identifier.

Parameters

identifier (*int*) –

Returns

TabularViewFilterMenu

create_header_widget(*identifier, area, with_menu=True*)

Returns a TabularViewHeaderWidget for given object_class identifier.

Parameters

- **identifier** (*str*) –

- **area** (*str*) –

- **with_menu** (*bool*) –

Returns

TabularViewHeaderWidget

static _get_insert_index(*pivot_list, catcher, position*)

Returns an index for inserting a new element in the given pivot list.

Returns

int

handle_header_dropped(*dropped, catcher, position=""*)

Updates pivots when a header is dropped.

Parameters

- **dropped** (*TabularViewHeaderWidget*) – drag source widget

- **catcher** (*TabularViewHeaderWidget or PivotTableHeaderView or FrozenTableView*) – drop target widget

- **position** (*str*) – either “before”, “after”, or “”

_change_selected_frozen_row(*current, previous*)

Sets the frozen value from selection in frozen table.

change_filter(*identifier, valid_values, has_filter*)

_add_values_to_frozen_table(*frozen_values*)

Adds values to frozen table.

Parameters

frozen_values (*set of tuple*) – values to add

_remove_values_from_frozen_table(*frozen_values*)

Removes values from frozen table.

Parameters

frozen_values (*set of tuple*) – values to remove

reload_frozen_table()

Resets the frozen model according to new selection in entity trees.

find_frozen_values(*frozen*)

Returns a list of tuples containing unique values for the frozen indexes.

Parameters

frozen (*tuple*) – A tuple of currently frozen indexes

Returns

frozen value

Return type

list

_change_frozen_value()

Updated frozen value according to selected row in Frozen table.

receive_session_rolled_back(*db_maps*)

Reacts to session rolled back event.

accepts_entity_item(*item*, *db_map*)

accepts_parameter_item(*item*, *db_map*)

accepts_member_object_item(*item*, *db_map*)

accepts_ith_member_object_item(*i*, *item*, *db_map*)

_frozen_table_reload_disabled()

spinetoolbox.spine_db_editor.widgets.tree_view_mixin

Contains the TreeViewMixin class.

Module Contents

Classes

<i>TreeViewMixin</i>	Provides object and relationship trees for the Spine db editor.
----------------------	---

class spinetoolbox.spine_db_editor.widgets.tree_view_mixin.**TreeViewMixin**(*args, **kwargs)

Provides object and relationship trees for the Spine db editor.

connect_signals()

Connects signals to slots.

init_models()

Initializes models.

static **_db_map_items**(*indexes*)

Groups items from given tree indexes by db map.

Returns

lists of dictionary items keyed by DiffDatabaseMapping

Return type

dict

_db_map_ids(*indexes*)

_db_map_class_ids(*indexes*)

export_selected(*selected_indexes*)

Exports data from given indexes in the entity tree.

show_add_object_classes_form()

Shows dialog to add new object classes.

show_add_objects_form(*parent_item*)

Shows dialog to add new objects.

show_add_object_group_form(*object_class_item*)

Shows dialog to add new object group.

show_manage_members_form(*object_item*)

Shows dialog to manage an object group.

show_add_relationship_classes_form(*parent_item*)

Shows dialog to add new relationship_class.

show_add_relationships_form(*parent_item*)

Shows dialog to add new relationships.

show_manage_relationships_form(*parent_item*)

edit_entity_tree_items(*selected_indexes*)

Starts editing given indexes.

show_edit_object_classes_form(*items*)

show_edit_objects_form(*items*)

show_edit_relationship_classes_form(*items*)

show_edit_relationships_form(*items*)

remove_entity_tree_items(*selected_indexes*)

Shows form to remove items from object treeview.

show_remove_entity_tree_items_form(*selected*)

`spinetoolbox.spine_db_editor.widgets.url_toolbar`

Contains the `UrlToolBar` class and helpers.

Module Contents

Classes

`UrlToolBar`

`_FilterWidget`

`_FilterArrayWidget`

`_DBListWidget`

`_UrlFilterDialog`

class `spinetoolbox.spine_db_editor.widgets.url_toolbar.UrlToolBar(db_editor)`

Bases: `PySide6.QtWidgets.QToolBar`

property `line_edit`

`_add_open_project_url_menu()`

`_update_ds_url_menu_enabled()`

`_connect_project_item_model_signals(slot)`

`_disconnect_project_item_model_signals(slot)`

`_update_open_project_url_menu()`

`_open_ds_url(action)`

`add_main_menu(menu)`

`_update_history_actions_availability()`

`add_urls_to_history(db_urls)`

Adds url to history.

Parameters

`db_urls` (*list of str*) –

`get_previous_urls()`

Returns previous urls in history.

Returns

list of str

get_next_urls()

Returns next urls in history.

Returns

list of str

_handle_line_edit_return_pressed()

set_current_urls(urls)

_show_filter_menu(_checked=False)

```
class spinetoolbox.spine_db_editor.widgets.url_toolbar._FilterWidget(db_mgr, db_map,
                                                                    item_type, filter_type,
                                                                    active_item,
                                                                    parent=None)
```

Bases: PySide6.QtWidgets.QTreeWidget

sizeHint()

filter_config()

```
class spinetoolbox.spine_db_editor.widgets.url_toolbar._FilterArrayWidget(db_mgr, db_map,
                                                                    parent=None)
```

Bases: PySide6.QtWidgets.QWidget

filter_selection_changed

filtered_url_codename()

sizeHint()

moveEvent(ev)

```
class spinetoolbox.spine_db_editor.widgets.url_toolbar._DBListWidget(db_mgr, db_maps,
                                                                    parent=None)
```

Bases: PySide6.QtWidgets.QTreeWidget

db_filter_selection_changed

sizeHint()

filtered_url_codenames()

```
class spinetoolbox.spine_db_editor.widgets.url_toolbar._UrlFilterDialog(db_mgr, db_maps,
                                                                    parent=None)
```

Bases: PySide6.QtWidgets.QDialog

filter_accepted

sizeHint()

_update_filter_enabled()

accept()

Submodules

`spinetoolbox.spine_db_editor.graphics_items`

Classes for drawing graphics items on graph view's `QGraphicsScene`.

Module Contents

Classes

<code>EntityItem</code>	Base class for <code>ObjectItem</code> and <code>RelationshipItem</code> .
<code>RelationshipItem</code>	Represents a relationship in the Entity graph.
<code>ObjectItem</code>	Represents an object in the Entity graph.
<code>ArcItem</code>	Connects a <code>RelationshipItem</code> to an <code>ObjectItem</code> .
<code>CrossHairsItem</code>	Creates new relationships directly in the graph.
<code>CrossHairsRelationshipItem</code>	Represents the relationship that's being created using the <code>CrossHairsItem</code> .
<code>CrossHairsArcItem</code>	Connects a <code>CrossHairsRelationshipItem</code> with the <code>CrossHairsItem</code> ,
<code>ObjectLabelItem</code>	Provides a label for <code>ObjectItem</code> 's.

Functions

<code>make_figure_graphics_item(scene[, z, static])</code>	Creates a <code>FigureCanvas</code> and adds it to the given scene.
--	---

`spinetoolbox.spine_db_editor.graphics_items.make_figure_graphics_item(scene, z=0, static=True)`

Creates a `FigureCanvas` and adds it to the given scene. Used for creating heatmaps and associated colorbars.

Parameters

- **scene** (`QGraphicsScene`) –
- **z** (`int`, *optional*) – z value. Defaults to 0.
- **static** (`bool`, *optional*) – if True (the default) the figure canvas is not movable

Returns

the graphics item that represents the canvas Figure: the figure in the canvas

Return type

`QGraphicsProxyWidget`

class `spinetoolbox.spine_db_editor.graphics_items.EntityItem(spine_db_editor, x, y, extent, db_map_ids)`

Bases: `PySide6.QtWidgets.QGraphicsRectItem`

Base class for `ObjectItem` and `RelationshipItem`.

Parameters

- **spine_db_editor** (`SpineDBEditor`) – ‘owner’
- **x** (`float`) – x-coordinate of central point

- **y** (*float*) – y-coordinate of central point
- **extent** (*int*) – Preferred extent
- **db_map_ids** (*tuple*) – tuple of (db_map, id) tuples

abstract property entity_type

property db_map_ids

property original_db_map_ids

property entity_class_type

property entity_name

property first_entity_class_id

property entity_class_name

property first_db_map_id

property first_id

property first_db_map

property display_data

property display_database

property db_maps

abstract _make_tool_tip()

abstract default_parameter_data()

entity_class_id(*db_map*)

entity_id(*db_map*)

db_map_data(*db_map*)

db_map_id(*db_map*)

boundingRect()

moveBy(*dx*, *dy*)

_init_bg()

refresh_icon()

Refreshes the icon.

_set_renderer(*renderer*)

shape()

Returns a shape containing the entire bounding rect, to work better with icon transparency.

paint(*painter*, *option*, *widget=None*)

Shows or hides the selection halo.

_paint_as_selected()

_paint_as_deselected()

add_arc_item(*arc_item*)

Adds an item to the list of arcs.

Parameters

arc_item (*ArcItem*) –

apply_zoom(*factor*)

Applies zoom.

Parameters

factor (*float*) – The zoom factor.

apply_rotation(*angle*, *center*)

Applies rotation.

Parameters

- **angle** (*float*) – The angle in degrees.
- **center** (*QPointF*) – Rotates around this point.

block_move_by(*dx*, *dy*)

mouseMoveEvent(*event*)

Moves the item and all connected arcs.

Parameters

event (*QGraphicsSceneMouseEvent*) –

update_arcs_line()

Moves arc items.

itemChange(*change*, *value*)

Keeps track of item's movements on the scene.

Parameters

- **change** (*GraphicsItemChange*) – a flag signalling the type of the change
- **value** – a value related to the change

Returns

the same value given as input

setVisible(*on*)

Sets visibility status for this item and all arc items.

Parameters

on (*bool*) –

_make_menu()

contextMenuEvent(*e*)

Shows context menu.

Parameters

e (*QGraphicsSceneMouseEvent*) – Mouse event

remove_db_map_ids(*db_map_ids*)

Removes *db_map_ids*.

add_db_map_ids(*db_map_ids*)

class spinetoolbox.spine_db_editor.graphics_items.**RelationshipItem**(*spine_db_editor*, *x*, *y*, *extent*, *db_map_ids*)

Bases: [EntityItem](#)

Represents a relationship in the Entity graph.

Initializes the item.

Parameters

- **spine_db_editor** (*GraphViewForm*) – ‘owner’
- **x** (*float*) – x-coordinate of central point
- **y** (*float*) – y-coordinate of central point
- **extent** (*int*) – preferred extent
- **db_map_ids** (*tuple*) – tuple of (db_map, id) tuples

property entity_type

property object_class_id_list

property object_name_list

default_parameter_data()

Return data to put as default in a parameter table when this item is selected.

object_id_list(*db_map*)

db_representation(*db_map*)

_make_tool_tip()

_init_bg()

follow_object_by(*dx*, *dy*)

add_arc_item(*arc_item*)

Adds an item to the list of arcs.

Parameters

arc_item ([ArcItem](#)) –

itemChange(*change*, *value*)

Rotates svg item if the relationship is 2D. This makes it possible to define e.g. an arrow icon for relationships that express direction.

_rotate_svg_item()

class spinetoolbox.spine_db_editor.graphics_items.**ObjectItem**(*spine_db_editor*, *x*, *y*, *extent*, *db_map_ids*)

Bases: [EntityItem](#)

Represents an object in the Entity graph.

Initializes the item.

Parameters

- **spine_db_editor** (*GraphViewForm*) – ‘owner’
- **x** (*float*) – x-coordinate of central point
- **y** (*float*) – y-coordinate of central point
- **extent** (*int*) – preferred extent
- **db_map_ids** (*tuple*) – tuple of (db_map, id) tuples

property entity_type**default_parameter_data()**

Return data to put as default in a parameter table when this item is selected.

db_representation(*db_map*)**shape()**

Returns a shape containing the entire bounding rect, to work better with icon transparency.

update_name()

Refreshes the name.

_make_tool_tip()**block_move_by**(*dx*, *dy*)**mouseDoubleClickEvent**(*e*)**_make_menu()****_duplicate()****_refresh_relationship_classes()****_populate_expand_collapse_menu**(*menu*)

Populates the ‘Expand’ or ‘Collapse’ menu.

Parameters**menu** (*QMenu*) –**_populate_add_relationships_menu**(*menu*)

Populates the ‘Add relationships’ menu.

Parameters**menu** (*QMenu*) –**_get_db_map_relationship_ids_to_expand_or_collapse**(*action*)**_expand**(*action*)**_collapse**(*action*)**_start_relationship**(*action*)**class** spinetoolbox.spine_db_editor.graphics_items.**ArcItem**(*rel_item*, *obj_item*, *width*)

Bases: PySide6.QtWidgets.QGraphicsPathItem

Connects a RelationshipItem to an ObjectItem.

Initializes item.

Parameters

- **rel_item** (*spinetoolbox.widgets.graph_view_graphics_items.RelationshipItem*) – relationship item
- **obj_item** (*spinetoolbox.widgets.graph_view_graphics_items.ObjectItem*) – object item
- **width** (*float*) – Preferred line width

_make_pen()

moveBy(dx, dy)

Does nothing. This item is not moved the regular way, but follows the EntityItems it connects.

update_line()

mousePressEvent(event)

Accepts the event so it's not propagated.

other_item(item)

apply_zoom(factor)

Applies zoom.

Parameters

- **factor** (*float*) – The zoom factor.

class spinetoolbox.spine_db_editor.graphics_items.**CrossHairsItem**(*args, **kwargs)

Bases: [RelationshipItem](#)

Creates new relationships directly in the graph.

Initializes the item.

Parameters

- **spine_db_editor** (*GraphViewForm*) – ‘owner’
- **x** (*float*) – x-coordinate of central point
- **y** (*float*) – y-coordinate of central point
- **extent** (*int*) – preferred extent
- **db_map_ids** (*tuple*) – tuple of (db_map, id) tuples

property entity_class_name

property entity_name

_make_tool_tip()

refresh_icon()

Refreshes the icon.

set_plus_icon()

set_check_icon()

set_normal_icon()

set_ban_icon()

set_icon(*unicode*, *color*=0)

Refreshes the icon.

mouseMoveEvent(*event*)

Moves the item and all connected arcs.

Parameters

event (*QGraphicsSceneMouseEvent*) –

block_move_by(*dx*, *dy*)

contextMenuEvent(*e*)

Shows context menu.

Parameters

e (*QGraphicsSceneMouseEvent*) – Mouse event

class spinetoolbox.spine_db_editor.graphics_items.**CrossHairsRelationshipItem**(*args,
**kwargs)

Bases: [*RelationshipItem*](#)

Represents the relationship that's being created using the CrossHairsItem.

Initializes the item.

Parameters

- **spine_db_editor** (*GraphViewForm*) – 'owner'
- **x** (*float*) – x-coordinate of central point
- **y** (*float*) – y-coordinate of central point
- **extent** (*int*) – preferred extent
- **db_map_ids** (*tuple*) – tuple of (db_map, id) tuples

_make_tool_tip()

refresh_icon()

Refreshes the icon.

contextMenuEvent(*e*)

Shows context menu.

Parameters

e (*QGraphicsSceneMouseEvent*) – Mouse event

class spinetoolbox.spine_db_editor.graphics_items.**CrossHairsArcItem**(*rel_item*, *obj_item*, *width*)

Bases: [*ArcItem*](#)

Connects a CrossHairsRelationshipItem with the CrossHairsItem, and with all the ObjectItem's in the relationship so far.

Initializes item.

Parameters

- **rel_item** (*spinetoolbox.widgets.graph_view_graphics_items.RelationshipItem*) – relationship item
- **obj_item** (*spinetoolbox.widgets.graph_view_graphics_items.ObjectItem*) – object item

- **width** (*float*) – Preferred line width

_make_pen()

class `spinetoolbox.spine_db_editor.graphics_items.ObjectLabelItem(entity_item)`

Bases: `PySide6.QtWidgets.QGraphicsTextItem`

Provides a label for `ObjectItem`'s.

Initializes item.

Parameters

entity_item (`spinetoolbox.widgets.graph_view_graphics_items.EntityItem`) –
The parent item.

entity_name_edited

setPlainText(*text*)

Set texts and resets position.

Parameters

text (*str*) –

reset_position()

Adapts item geometry so text is always centered.

`spinetoolbox.spine_db_editor.main`

Module Contents

Functions

<code>main()</code>	Launches Spine Db Editor as its own application.
<code>_make_argument_parser()</code>	Builds a command line argument parser.

`spinetoolbox.spine_db_editor.main.main()`

Launches Spine Db Editor as its own application.

`spinetoolbox.spine_db_editor.main._make_argument_parser()`

Builds a command line argument parser.

Returns

parser

Return type

`ArgumentParser`

spinetoolbox.spine_db_editor.scenario_generation

Contains functions for automatically generating scenarios from a set of alternatives.

Module Contents

Functions

<code>all_combinations(alternatives)</code>	Creates all possible combinations of alternatives.
<code>unique_alternatives(alternatives)</code>	Creates all possible single-alternative scenarios.

`spinetoolbox.spine_db_editor.scenario_generation.all_combinations(alternatives)`

Creates all possible combinations of alternatives.

Parameters

alternatives (*Iterable of Any*) – alternatives

Returns

lists containing alternatives for each scenario

Return type

list of list

`spinetoolbox.spine_db_editor.scenario_generation.unique_alternatives(alternatives)`

Creates all possible single-alternative scenarios.

Parameters

alternatives (*Iterable of Any*) – alternatives

Returns

tuples containing alternatives for each scenario

Return type

list of list

spinetoolbox.widgets

Init file for widgets package. Intentionally empty.

Submodules

spinetoolbox.widgets.about_widget

A widget for presenting basic information about the application.

Module Contents

Classes

AboutWidget

About widget class.

class spinetoolbox.widgets.about_widget.**AboutWidget**(*toolbox*)

Bases: PySide6.QtWidgets.QWidget

About widget class.

Parameters

toolbox (*ToolboxUI*) – QMainWindow instance

copy_to_clipboard(*_*)

Copies package and Python info to clipboard.

calc_pos()

Calculate the top-left corner position of this widget in relation to main window position and size in order to show about window in the middle of the main window.

setup_license_text()

Add license to QTextBrowser.

keyPressEvent(*e*)

Close form when Escape, Enter, Return, or Space bar keys are pressed.

Parameters

e (*QKeyEvent*) – Received key press event.

closeEvent(*event=None*)

Handle close window.

Parameters

event (*QEvent*) – Closing event if 'X' is clicked.

mousePressEvent(*e*)

Save mouse position at the start of dragging.

Parameters

e (*QMouseEvent*) – Mouse event

mouseReleaseEvent(*e*)

Save mouse position at the end of dragging.

Parameters

e (*QMouseEvent*) – Mouse event

mouseMoveEvent(*e*)

Moves the window when mouse button is pressed and mouse cursor is moved.

Parameters

e (*QMouseEvent*) – Mouse event

`spinetoolbox.widgets.add_project_item_widget`

Widget shown to user when a new Project Item is created.

Module Contents

Classes

<i>AddProjectItemWidget</i>	A widget to query user's preferences for a new item.
---	--

```
class spinetoolbox.widgets.add_project_item_widget.AddProjectItemWidget(toolbox, x, y, class_,  
                                                                           spec="")
```

Bases: PySide6.QtWidgets.QWidget

A widget to query user's preferences for a new item.

toolbox

Parent widget

Type

ToolboxUI

x

X coordinate of new item

Type

int

y

Y coordinate of new item

Type

int

Initialize class.

connect_signals()

Connect signals to slots.

handle_name_changed()

Update label to show upcoming folder name.

handle_ok_clicked()

Check that given item name is valid and add it to project.

abstract call_add_item()

Creates new Item according to user's selections.

Must be reimplemented by subclasses.

keyPressEvent(e)

Close Setup form when escape key is pressed.

Parameters

e (*QKeyEvent*) – Received key press event.

closeEvent (*event=None*)

Handle close window.

Parameters

event (*QEvent*) – Closing event if ‘X’ is clicked.

spinetoolbox.widgets.add_up_spine_opt_wizard

Classes for custom QDialogs for julia setup.

Module Contents

Classes

<i>_PageId</i>	Enum where members are also (and must be) ints
<i>AddUpSpineOptWizard</i>	A wizard to install & upgrade SpineOpt.
<i>IntroPage</i>	
<i>SelectJuliaPage</i>	
<i>CheckPreviousInstallPage</i>	
<i>AddUpSpineOptPage</i>	A QWizards page with a log. Useful for pages that need to capture the output of a process.
<i>SuccessPage</i>	
<i>FailurePage</i>	
<i>TroubleshootProblemsPage</i>	
<i>TroubleshootSolutionPage</i>	
<i>ResetRegistryPage</i>	A QWizards page with a log. Useful for pages that need to capture the output of a process.
<i>AddUpSpineOptAgainPage</i>	A QWizards page with a log. Useful for pages that need to capture the output of a process.
<i>TotalFailurePage</i>	

Functions

<i>_clear_layout</i> (layout)

class spinetoolbox.widgets.add_up_spine_opt_wizard._PageId

Bases: `enum.IntEnum`

Enum where members are also (and must be) ints

Initialize self. See `help(type(self))` for accurate signature.

INTRO

SELECT_JULIA

CHECK_PREVIOUS_INSTALL

ADD_UP_SPINE_OPT

SUCCESS

FAILURE

TROUBLESHOOT_PROBLEMS

TROUBLESHOOT_SOLUTION

RESET_REGISTRY

ADD_UP_SPINE_OPT_AGAIN

TOTAL_FAILURE

```
class spinetoolbox.widgets.add_up_spine_opt_wizard.AddUpSpineOptWizard(parent, julia_exe,  
                                                                    julia_project)
```

Bases: PySide6.QtWidgets.QWizard

A wizard to install & upgrade SpineOpt.

Parameters

- **parent** (*QWidget*) – the parent widget (SettingsWidget)
- **julia_exe** (*str*) – path to Julia executable
- **julia_project** (*str*) – path to Julia project

```
class spinetoolbox.widgets.add_up_spine_opt_wizard.IntroPage(parent)
```

Bases: PySide6.QtWidgets.QWizardPage

nextId()

```
class spinetoolbox.widgets.add_up_spine_opt_wizard.SelectJuliaPage(parent, julia_exe,  
                                                                    julia_project)
```

Bases: PySide6.QtWidgets.QWizardPage

initializePage()

_select_julia_exe()

_select_julia_project()

nextId()

```
class spinetoolbox.widgets.add_up_spine_opt_wizard.CheckPreviousInstallPage(parent)
```

Bases: PySide6.QtWidgets.QWizardPage

isComplete()

cleanupPage()

initializePage()

```

    _handle_check_install_finished(ret)

    nextId()

class spinetoolbox.widgets.add_up_spine_opt_wizard.AddUpSpineOptPage(parent)
    Bases: spinetoolbox.widgets.custom_qwidgets.QWizardProcessPage
    A QWizards page with a log. Useful for pages that need to capture the output of a process.
    initializePage()
    _handle_spine_opt_add_up_finished(ret)
    nextId()

class spinetoolbox.widgets.add_up_spine_opt_wizard.SuccessPage(parent)
    Bases: PySide6.QtWidgets.QWizardPage
    initializePage()
    nextId()

class spinetoolbox.widgets.add_up_spine_opt_wizard.FailurePage(parent)
    Bases: PySide6.QtWidgets.QWizardPage
    _handle_check_box_clicked(checked=False)
    initializePage()
    nextId()

class spinetoolbox.widgets.add_up_spine_opt_wizard.TroubleshootProblemsPage(parent)
    Bases: PySide6.QtWidgets.QWizardPage
    isComplete()
    _show_log(_=False)
    nextId()

class spinetoolbox.widgets.add_up_spine_opt_wizard.TroubleshootSolutionPage(parent)
    Bases: PySide6.QtWidgets.QWizardPage
    cleanupPage()
    initializePage()
    _initialize_page_solution1()
    _initialize_page_solution2()
    nextId()

class spinetoolbox.widgets.add_up_spine_opt_wizard.ResetRegistryPage(parent)
    Bases: spinetoolbox.widgets.custom_qwidgets.QWizardProcessPage
    A QWizards page with a log. Useful for pages that need to capture the output of a process.
    initializePage()
    _handle_registry_reset_finished(ret)

```

nextId()

class spinetoolbox.widgets.add_up_spine_opt_wizard.**AddUpSpineOptAgainPage**(*parent*)

Bases: [AddUpSpineOptPage](#)

A QWizards page with a log. Useful for pages that need to capture the output of a process.

nextId()

class spinetoolbox.widgets.add_up_spine_opt_wizard.**TotalFailurePage**(*parent*)

Bases: PySide6.QtWidgets.QWizardPage

nextId()

spinetoolbox.widgets.add_up_spine_opt_wizard.**_clear_layout**(*layout*)

spinetoolbox.widgets.array_editor

Contains an editor widget for array type parameter values.

Module Contents

Classes

[*ArrayEditor*](#)

Editor widget for Arrays.

class spinetoolbox.widgets.array_editor.**ArrayEditor**(*parent=None*)

Bases: PySide6.QtWidgets.QWidget

Editor widget for Arrays.

Parameters

parent (*QWidget*, *optional*) – parent widget

set_value(*value*)

Sets the parameter_value for editing in this widget.

Parameters

value (*Array*) – value for editing

value()

Returns the array currently being edited.

Returns

array

Return type

Array

_check_if_plotting_enabled(*type_name*)

Checks is array's data type allows the array to be plotted.

Parameters

type_name (*str*) – data type's name

`_change_value_type`(*type_name*)

`open_value_editor`(*index*)

Opens an editor widget for array element.

Parameters

`index` (*QModelIndex*) – element’s index

`_show_table_context_menu`(*position*)

Shows the table’s context menu.

Parameters

`position` (*QPoint*) – menu’s position on the table

`_update_plot`(*topLeft=None, bottomRight=None, roles=None*)

Updates the plot widget.

`_open_header_editor`(*column*)

`spinetoolbox.widgets.array_value_editor`

An editor dialog for Array elements.

Module Contents

Classes

ArrayValueEditor

Editor widget for Array elements.

class `spinetoolbox.widgets.array_value_editor.ArrayValueEditor`(*index, value_type, parent=None*)

Bases: `spinetoolbox.widgets.parameter_value_editor_base.ParameterValueEditorBase`

Editor widget for Array elements.

Parameters

- **`index`** (*QModelIndex*) – an index to a parameter_value in parent_model
- **`parent`** (*QWidget, optional*) – a parent widget

`_set_data`(*value*)

See base class.

`spinetoolbox.widgets.code_text_edit`

Provides simple text editor for programming purposes.

Module Contents

Classes

<i>CodeTextEdit</i>	A plain text edit with syntax highlighting and line numbers.
<i>LineNumberArea</i>	

```
class spinetoolbox.widgets.code_text_edit.CodeTextEdit(*arg, **kwargs)
```

```
    Bases: PySide6.QtWidgets.QPlainTextEdit
```

```
    A plain text edit with syntax highlighting and line numbers.
```

```
    insertFromMimeData(source)
```

```
    file_selected(status)
```

```
    set_lexer_name(lexer_name)
```

```
    setPlainText(text)
```

```
    setDocument(doc)
```

```
    line_number_area_width()
```

```
    _update_line_number_area_width(new_block_count=0)
```

```
    _update_line_number_area(rect, dy)
```

```
    _update_line_number_area_cursor_position()
```

```
    resizeEvent(event)
```

```
    line_number_area_paint_event(ev)
```

```
class spinetoolbox.widgets.code_text_edit.LineNumberArea(editor)
```

```
    Bases: PySide6.QtWidgets.QWidget
```

```
    sizeHint()
```

```
    paintEvent(ev)
```

```
spinetoolbox.widgets.commit_dialog
```

```
Classes for custom QDialogs to add edit and remove database items.
```

Module Contents

Classes

<i>CommitDialog</i>	A dialog to query user's preferences for new commit.
---------------------	--

class spinetoolbox.widgets.commit_dialog.**CommitDialog**(parent, *db_names)

Bases: PySide6.QtWidgets.QDialog

A dialog to query user's preferences for new commit.

Parameters

- **parent** (*QWidget*) – the parent widget
- **db_names** (*Iterable of str*) – database names

receive_text_changed()

Called when text changes in the commit msg text edit. Enable/disable commit button accordingly.

spinetoolbox.widgets.custom_combobox

Contains a custom combo box for the custom open project dialog.

Module Contents

Classes

<i>ElidedCombobox</i>	Combobox with elided text.
<i>OpenProjectDialogComboBox</i>	

class spinetoolbox.widgets.custom_combobox.**ElidedCombobox**

Bases: PySide6.QtWidgets.QComboBox

Combobox with elided text.

paintEvent(event)

class spinetoolbox.widgets.custom_combobox.**OpenProjectDialogComboBox**

Bases: PySide6.QtWidgets.QComboBox

keyPressEvent(e)

Interrupts Enter and Return key presses when QComboBox is in focus. This is needed to prevent showing the 'Not a valid Spine Toolbox project' Notifier every time Enter is pressed.

Parameters

- **e** (*QKeyEvent*) – Received key press event.

spinetoolbox.widgets.custom_delegates

Custom item delegates.

Module Contents

Classes

ComboBoxDelegate

CheckBoxDelegate

A delegate that places a fully functioning QCheckBox.

RankDelegate

A delegate that places a QCheckBox but draws a number instead of the check.

class spinetoolbox.widgets.custom_delegates.**ComboBoxDelegate**(*items*)

Bases: PySide6.QtWidgets.QStyledItemDelegate

createEditor(*parent, option, index*)

paint(*painter, option, index*)

setEditorData(*editor, index*)

setModelData(*editor, model, index*)

updateEditorGeometry(*editor, option, index*)

_finalize_editing(*editor*)

class spinetoolbox.widgets.custom_delegates.**CheckBoxDelegate**(*parent, centered=True*)

Bases: PySide6.QtWidgets.QStyledItemDelegate

A delegate that places a fully functioning QCheckBox.

Parameters

- **parent** (*QWidget*) –
- **centered** (*bool*) – whether or not the checkbox should be center-aligned in the widget

data_committed

createEditor(*parent, option, index*)

Important, otherwise an editor is created if the user clicks in this cell. ** Need to hook up a signal to the model.

paint(*painter, option, index*)

Paint a checkbox without the label.

static _do_paint(*painter, checkbox_style_option, index*)

editorEvent(*event, model, option, index*)

Change the data in the model and the state of the checkbox when user presses left mouse button and this cell is editable. Otherwise do nothing.

setModelData(*editor, model, index*)

Do nothing. Model data is updated by handling the *data_committed* signal.

get_checkbox_rect(*option*)

class spinetoolbox.widgets.custom_delegates.**RankDelegate**(*parent, centered=True*)

Bases: [CheckBoxDelegate](#)

A delegate that places a QCheckBox but draws a number instead of the check.

Parameters

- **parent** (*QWidget*) –
- **centered** (*bool*) – whether or not the checkbox should be center-aligned in the widget

static _do_paint(*painter, checkbox_style_option, index*)

spinetoolbox.widgets.custom_editors

Custom editors for model/view programming.

Module Contents

Classes

CustomLineEdit	A custom QLineEdit to handle data from models.
ParameterValueLineEdit	A custom QLineEdit to handle data from models.
PivotHeaderTableLineEdit	Line editor that is visible on Pivot view's header tables due to a clever hack.
_CustomLineEditDelegate	A delegate for placing a CustomLineEdit on the first row of SearchBarEditor.
SearchBarEditor	A Google-like search bar, implemented as a QTableView with a _CustomLineEditDelegate in the first row.
CheckListEditor	A check list editor.
_IconPainterDelegate	A delegate to highlight decorations in a QListWidget.
IconColorEditor	An editor to let the user select an icon and a color for an object_class.

class spinetoolbox.widgets.custom_editors.**CustomLineEdit**

Bases: PySide6.QtWidgets.QLineEdit

A custom QLineEdit to handle data from models.

set_data(*data*)

Sets editor's text.

Parameters

data (*Any*) – anything convertible to string

data()

Returns editor's text.

Returns

editor's text

Return type

str

keyPressEvent(*event*)

Prevents shift key press to clear the contents.

class spinetoolbox.widgets.custom_editors.ParameterValueLineEditor

Bases: [CustomLineEditor](#)

A custom QLineEdit to handle data from models.

set_data(*data*)

See base class.

data()

See base class.

class spinetoolbox.widgets.custom_editors.PivotHeaderTableLineEditor(*parent=None*)

Bases: [CustomLineEditor](#)

Line editor that is visible on Pivot view's header tables due to a clever hack.

Parameters

parent (*QWidget*, *optional*) – parent widget

fix_geometry()

Fixes editor's position after reparenting.

class spinetoolbox.widgets.custom_editors._CustomLineEditDelegate

Bases: PySide6.QtWidgets.QStyledItemDelegate

A delegate for placing a CustomLineEditor on the first row of SearchBarEditor.

text_edited**setModelData(*editor*, *model*, *index*)****createEditor(*parent*, *option*, *index*)**

Create editor and 'forward' *textEdited* signal.

eventFilter(*editor*, *event*)

Handle all sort of special cases.

class spinetoolbox.widgets.custom_editors.SearchBarEditor(*parent*, *tutor=None*)

Bases: PySide6.QtWidgets.QTableView

A Google-like search bar, implemented as a QTableView with a _CustomLineEditDelegate in the first row.

Parameters

- **parent** (*QWidget*, *optional*) – parent widget
- **tutor** (*QWidget*, *optional*) – another widget used for positioning.

data_committed**set_data(*current*, *items*)**

Populates model.

Parameters

- **current** (*str*) – item that is currently selected from given items

- **items** (*Sequence of str*) – items to show in the list

set_base_offset(*offset*)

Changes the base offset that is applied to the editor's position.

Parameters

offset (*QPoint*) – new offset

update_geometry(*option*)

Updates geometry.

Parameters

option (*QStyleOptionViewItem*) – style information

refit()

Changes the position and size of the editor to fit the window.

data()

Returns editor's final data.

Returns

editor data

Return type

str

_handle_delegate_text_edited(*text*)

Filters model as the first row is being edited.

Parameters

text (*str*) – text the user has entered on the first row

_proxy_model_filter_accepts_row(*source_row*, *source_parent*)

Always accept first row while filtering the rest.

Parameters

- **source_row** (*int*) – source row index
- **source_parent** (*QModelIndex*) – parent index for source row

Returns

True if row is accepted, False otherwise

Return type

bool

keyPressEvent(*event*)

Sets data from current index into first index as the user navigates through the table using the up and down keys.

currentChanged(*current*, *previous*)

edit_first_index()

Edits first index if valid and not already being edited.

mousePressEvent(*event*)

Commits data.

class spinetoolbox.widgets.custom_editors.**CheckListEditor**(*parent*, *tutor=None*)

Bases: PySide6.QtWidgets.QTableView

A check list editor.

Parameters

- **parent** (*QWidget*) – parent widget
- **tutor** (*QWidget*, *optional*) – a widget that helps in positioning

keyPressEvent(*event*)

Toggles checked state if the user presses space.

toggle_selected(*index*)

Adds or removes given index from selected items.

Parameters

- index** (*QModelIndex*) – index to toggle

mouseMoveEvent(*event*)

Sets the current index to the one under mouse.

mousePressEvent(*event*)

Toggles checked state of pressed index.

set_data(*items*, *checked_items*)

Sets data and updates geometry.

Parameters

- **items** (*Sequence(str)*) – All items.
- **checked_items** (*Sequence(str)*) – Initially checked items.

data()

Returns a comma separated list of checked items.

Returns

str

update_geometry(*option*)

Updates geometry.

Parameters

- option** (*QStyleOptionViewItem*) – style information

class spinetoolbox.widgets.custom_editors.**_IconPainterDelegate**

Bases: PySide6.QtWidgets.QStyledItemDelegate

A delegate to highlight decorations in a QListWidget.

paint(*painter*, *option*, *index*)

Paints selected items using the highlight brush.

class spinetoolbox.widgets.custom_editors.**IconColorEditor**(*parent*)

Bases: PySide6.QtWidgets.QDialog

An editor to let the user select an icon and a color for an object_class.

Parameters

- parent** (*QWidget*) – parent widget

`_proxy_model_filter_accepts_row`(*source_row*, *source_parent*)

Filters icons according to search terms.

Parameters

- **`source_row`** (*int*) – source row index
- **`source_parent`** (*QModelIndex*) – parent index for source row

Returns

True if row is accepted, False otherwise

Return type

bool

`connect_signals`()

Connects signals to slots.

`set_data`(*data*)

Sets current icon data.

Parameters

`data` (*int*) – database icon data

`data`()

Gets current icon data.

Returns

database icon data

Return type

int

`spinetoolbox.widgets.custom_menus`

Classes for custom context menus and pop-up menus.

Module Contents

Classes

<i>CustomContextMenu</i>	Context menu master class for several context menus.
<i>OpenProjectDialogComboBoxContextMenu</i>	Context menu master class for several context menus.
<i>CustomPopupMenu</i>	Popup menu master class for several popup menus.
<i>ItemSpecificationMenu</i>	Context menu class for item specifications.
<i>RecentProjectsPopupMenu</i>	Recent projects menu embedded to 'File-Open recent' QAction.
<i>KernelsPopupMenu</i>	Menu embedded into 'Consoles->Start Jupyter Console' QMenu.
<i>FilterMenuBase</i>	Filter menu.

`class` `spinetoolbox.widgets.custom_menus.CustomContextMenu`(*parent*, *position*)

Bases: `PySide6.QtWidgets.QMenu`

Context menu master class for several context menus.

Parameters

- **parent** (*QWidget*) – Parent for menu widget (ToolboxUI)
- **position** (*QPoint*) – Position on screen

add_action(*text, icon=QIcon(), enabled=True*)

Adds an action to the context menu.

Parameters

- **text** (*str*) – Text description of the action
- **icon** (*QIcon*) – Icon for menu item
- **enabled** (*bool*) – Is action enabled?

set_action(*option*)

Sets the action which was clicked.

Parameters

- **option** (*str*) – string with the text description of the action

get_action()

Returns the clicked action, a string with a description.

class spinetoolbox.widgets.custom_menus.**OpenProjectDialogComboBoxContextMenu**(*parent, position*)

Bases: [*CustomContextMenu*](#)

Context menu master class for several context menus.

Parameters

- **parent** (*QWidget*) – Parent for menu widget
- **position** (*QPoint*) – Position on screen

class spinetoolbox.widgets.custom_menus.**CustomPopupMenu**(*parent*)

Bases: `PySide6.QtWidgets.QMenu`

Popup menu master class for several popup menus.

Parameters

- **parent** (*QWidget*) – Parent widget of this pop-up menu

add_action(*text, slot, enabled=True, tooltip=None, icon=None*)

Adds an action to the popup menu.

Parameters

- **text** (*str*) – Text description of the action
- **slot** (*method*) – Method to connect to action's triggered signal
- **enabled** (*bool*) – Is action enabled?
- **tooltip** (*str*) – Tool tip for the action
- **icon** (*QIcon*) – Action icon

class spinetoolbox.widgets.custom_menus.**ItemSpecificationMenu**(*toolbox, index, item=None*)

Bases: [*CustomPopupMenu*](#)

Context menu class for item specifications.

Parameters

- **toolbox** (*ToolboxUI*) – Toolbox that requests this menu, used as parent.
- **index** (*QModelIndex*) – the index
- **item** (*ProjectItem*, *optional*) – passed to `show_specification_form`

class `spinetoolbox.widgets.custom_menus.RecentProjectsPopupMenu(parent)`

Bases: *CustomPopupMenu*

Recent projects menu embedded to 'File-Open recent' QAction.

Parameters

parent (*QWidget*) – Parent widget of this menu (ToolboxUI)

has_recents()

Returns True if recent projects available, False otherwise.

add_recent_projects()

Reads the previous project names and paths from QSettings. Adds them to the QMenu as QActions.

call_clear_recents(*checked*)

Slot for Clear recents menu item.

Parameters

checked (*bool*) – Argument sent by triggered signal

call_open_project(*checked*, *p*)

Slot for catching the user selected action from the recent projects menu.

Parameters

- **checked** (*bool*) – Argument sent by triggered signal
- **p** (*str*) – Full path to a project file

class `spinetoolbox.widgets.custom_menus.KernelsPopupMenu(parent)`

Bases: *CustomPopupMenu*

Menu embedded into 'Consoles->Start Jupyter Console' QMenu.

Parameters

parent (*QWidget*) – Parent widget of this menu (ToolboxUI)

add_kernel(*kernel_name*, *resource_dir*, *cond*, *ico*, *deats*)

Adds a kernel entry as an action to this menu.

call_open_console(*checked*, *kernel_name*, *icon*, *conda*)

Slot for catching the user selected action from the kernel's menu.

Parameters

- **checked** (*bool*) – Argument sent by triggered signal
- **kernel_name** (*str*) – Kernel name to launch
- **icon** (*QIcon*) – Icon representing the kernel language
- **conda** (*bool*) – Is this a Conda kernel spec?

```
class spinetoolbox.widgets.custom_menus.FilterMenuBase(parent)
```

Bases: PySide6.QtWidgets.QMenu

Filter menu.

Parameters

parent (*QWidget*) – a parent widget

_set_up(*make_filter_model*, *args, **kwargs)

connect_signals()

add_items_to_filter_list(*items*)

remove_items_from_filter_list(*items*)

_clear_filter()

_check_filter()

_change_filter()

abstract emit_filter_changed(*valid_values*)

wipe_out()

`spinetoolbox.widgets.custom_qcombobox`

Class for a custom QComboBox.

Module Contents

Classes

CustomQComboBox

A custom QComboBox for showing kernels in Settings->Tools.

```
class spinetoolbox.widgets.custom_qcombobox.CustomQComboBox
```

Bases: PySide6.QtWidgets.QComboBox

A custom QComboBox for showing kernels in Settings->Tools.

mouseMoveEvent(*e*)

Catch mouseMoveEvent and accept it because the comboBox popup (QListView) has mouse tracking on as default. This makes sure the comboBox popup appears in correct position and clicking on the combobox repeatedly does not move the Settings window.

spinetoolbox.widgets.custom_qgraphicsscene

Custom QGraphicsScene used in the Design View.

Module Contents

Classes

<i>CustomGraphicsScene</i>	A custom QGraphicsScene. It provides signals to notify about items,
<i>DesignGraphicsScene</i>	A scene for the Design view.

class spinetoolbox.widgets.custom_qgraphicsscene.CustomGraphicsScene

Bases: PySide6.QtWidgets.QGraphicsScene

A custom QGraphicsScene. It provides signals to notify about items, and a method to center all items in the scene.

At the moment it's used by DesignGraphicsScene and the GraphViewMixin

item_move_finished

Emitted when an item has finished moving.

center_items()

Centers toplevel items in the scene.

class spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene(*parent, toolbox*)

Bases: *CustomGraphicsScene*

A scene for the Design view.

Mainly, it handles drag and drop events of ProjectItemDragMixin sources.

Parameters

- **parent** (*QObject*) – scene's parent object
- **toolbox** (*ToolboxUI*) – reference to the main window

_handle_timeout()

clear_icons_and_links()

mouseMoveEvent(*event*)

Moves link drawer.

mousePressEvent(*event*)

Puts link drawer to sleep and log message if it looks like the user doesn't know what they're doing.

mouseReleaseEvent(*event*)

Makes link if drawer is released over a valid connector button.

_finish_link()

emit_connection_failed()

keyPressEvent(*event*)

Puts link drawer to sleep if user presses ESC.

connect_signals()

Connect scene signals.

project_item_icons()

handle_selection_changed()

Synchronizes selection with the project tree.

set_bg_color(*color*)

Change background color when this is changed in Settings.

Parameters

color (*QColor*) – Background color

set_bg_choice(*bg_choice*)

Set background choice when this is changed in Settings.

Parameters

bg (*str*) – “grid”, “tree”, or “solid”

dragLeaveEvent(*event*)

Accept event.

dragEnterEvent(*event*)

Accept event. Then call the super class method only if drag source is not a ProjectItemDragMixin.

dragMoveEvent(*event*)

Accept event. Then call the super class method only if drag source is not a ProjectItemDragMixin.

dropEvent(*event*)

Only accept drops when the source is an instance of ProjectItemDragMixin. Capture text from event’s mimedata and show the appropriate ‘Add Item form.’

event(*event*)

Accepts GraphicsSceneHelp events without doing anything, to not interfere with our usage of QToolTip.showText in graphics_items.ExclamationIcon.

drawBackground(*painter, rect*)

Reimplemented method to make a custom background.

Parameters

- **painter** (*QPainter*) – Painter that is used to paint background
- **rect** (*QRectF*) – The exposed (viewport) rectangle in scene coordinates

_draw_solid_bg(*painter, rect*)

Draws solid bg.

_draw_grid_bg(*painter, rect*)

Draws grid bg.

_draw_tree_bg(*painter, rect*)

Draws ‘tree of life’ bg.

select_link_drawer(*drawer_type*)

Selects current link drawer.

Parameters

drawer_type ([LinkType](#)) – selected link drawer’s type

spinetoolbox.widgets.custom_qgraphicsviews

Classes for custom QGraphicsViews for the Design and Graph views.

Module Contents

Classes

<i>CustomQGraphicsView</i>	Super class for Design and Entity QGraphicsViews.
<i>DesignQGraphicsView</i>	QGraphicsView for the Design View.

class spinetoolbox.widgets.custom_qgraphicsviews.**CustomQGraphicsView**(*parent*)

Bases: PySide6.QtWidgets.QGraphicsView

Super class for Design and Entity QGraphicsViews.

parent

Parent widget

Type

QWidget

Init CustomQGraphicsView.

abstract property **_qsettings**

property **zoom_factor**

reset_zoom()

Resets zoom to the default factor.

keyPressEvent(*event*)

Overridden method. Enable zooming with plus and minus keys (comma resets zoom). Send event downstream to QGraphicsItems if pressed key is not handled here.

Parameters

event ([QKeyEvent](#)) – Pressed key

mousePressEvent(*event*)

Set rubber band selection mode if Control pressed. Enable resetting the zoom factor from the middle mouse button.

mouseReleaseEvent(*event*)

Reestablish scroll hand drag mode.

_use_smooth_zoom()

wheelEvent(*event*)

Zooms in/out.

Parameters

event (*QWheelEvent*) – Mouse wheel event

resizeEvent(*event*)

Updates zoom if needed when the view is resized.

Parameters

event (*QResizeEvent*) – a resize event

setScene(*scene*)

Sets a new scene to this view.

Parameters

scene (*ShrinkingScene*) – a new scene

_handle_item_move_finished(*item*)

_update_zoom_limits()

Updates the minimum zoom limit and the zoom level with which the view fits all the items in the scene.

abstract _compute_max_zoom()

_handle_zoom_time_line_advanced(*pos*)

Performs zoom whenever the smooth zoom time line advances.

_handle_transformation_time_line_finished()

Cleans up after the smooth transformation time line finishes.

_handle_resize_time_line_finished()

Cleans up after resizing time line finishes.

zoom_in()

Perform a zoom in with a fixed scaling.

zoom_out()

Perform a zoom out with a fixed scaling.

gentle_zoom(*factor*, *zoom_focus=None*)

Perform a zoom by a given factor.

Parameters

- **factor** (*float*) – a scaling factor relative to the current scene scaling
- **zoom_focus** (*QPoint*) – focus of the zoom, e.g. mouse pointer position

_zoom(*factor*)

_get_viewport_scene_rect()

Returns the viewport rect mapped to the scene.

Returns

QRectF

_ensure_item_visible(*item*)

Resets zoom if item is not visible.

_set_preferred_scene_rect()

Sets the scene rect to the result of uniting the scene viewport rect and the items bounding rect.

class spinetoolbox.widgets.custom_qgraphicsviews.**DesignQGraphicsView**(*parent*)

Bases: [CustomQGraphicsView](#)

QGraphicsView for the Design View.

Parameters

parent (*QWidget*) – parent widget

property _qsettings**set_ui**(*toolbox*)

Set a new scene into the Design View when app is started.

reset_zoom()

Resets zoom to the default factor.

_compute_max_zoom()**add_icon**(*item_name*)

Adds project item's icon to the scene.

Parameters

item_name (*str*) – project item's name

remove_icon(*item_name*)

Removes project item's icon from scene.

Parameters

item_name (*str*) – name of the icon to remove

add_link(*src_connector*, *dst_connector*)

Pushes an AddLinkCommand to the toolbox undo stack.

Parameters

- **src_connector** ([ConnectorButton](#)) – source connector button
- **dst_connector** ([ConnectorButton](#)) – destination connector button

do_add_link(*connection*)

Adds given connection to the Design view.

Parameters

connection (*Connection*) – the connection to add

do_update_link(*updated_connection*)

Replaces a link on the Design view.

Parameters

updated_connection (*Connection*) – connection that was updated

remove_links(*links*)

Pushes a RemoveConnectionsCommand to the Toolbox undo stack.

Parameters

links (*list of Link*) – links to remove

do_remove_link(*connection*)

Removes a link from the scene.

Parameters

connection (*ConnectionBase*) – link’s connection

remove_selected_links()

take_link(*link*)

Remove link, then start drawing another one from the same source connector.

add_jump(*src_connector*, *dst_connector*)

Pushes an AddJumpCommand to the Toolbox undo stack.

Parameters

- **src_connector** (*ConnectorButton*) – source connector button
- **dst_connector** (*ConnectorButton*) – destination connector button

do_add_jump(*jump*)

Adds given jump to the Design view.

Parameters

jump (*Jump*) – jump to add

do_update_jump(*updated_jump*)

Replaces a jump link on the Design view.

Parameters

updated_jump (*Jump*) – jump that was updated

do_remove_jump(*jump*)

Removes a jump from the scene.

Parameters

jump (*Jump*) – link’s jump

contextMenuEvent(*event*)

Shows context menu for the blank view

Parameters

event (*QContextMenuEvent*) – Event

spinetoolbox.widgets.custom_qlineedit

Classes for custom line edits.

Module Contents

Classes

PropertyQLineEdit

A custom QLineEdit for Project Item Properties.

class `spinetoolbox.widgets.custom_qlineedit.PropertyQLineEdit`

Bases: `spinetoolbox.widgets.custom_qwidgets.UndoRedoMixin`, `PySide6.QtWidgets.QLineEdit`

A custom QLineEdit for Project Item Properties.

setText(*text*)

Overridden to prevent the cursor going to the end whenever the user is still editing. This happens because we set the text programmatically in undo/redo implementations.

spinetoolbox.widgets.custom_qtableview

Custom QTableView classes that support copy-paste and the like.

Module Contents**Classes**

<i>CopyPasteTableView</i>	Custom QTableView class with copy and paste methods.
<i>AutoFilterCopyPasteTableView</i>	Custom QTableView class with autofilter functionality.
<i>IndexedParameterValueTableViewBase</i>	Custom QTableView base class with copy and paste methods for indexed parameter values.
<i>TimeSeriesFixedResolutionTableView</i>	A QTableView for fixed resolution time series table.
<i>IndexedValueTableView</i>	A QTableView class with for variable resolution time series and time patterns.
<i>ArrayTableView</i>	Custom QTableView with copy and paste methods for single column tables.
<i>MapTableView</i>	Custom QTableView with copy and paste methods for map tables.

Functions

<i>_range(selection)</i>	Returns the top left and bottom right corners of selection.
<i>_could_be_time_stamp(s)</i>	Evaluates if given string could be a time stamp.
<i>system_lc_numeric()</i>	

Attributes

<i>_</i>	
<i>_NOT_TIME_STAMP</i>	

`spinetoolbox.widgets.custom_qtableview._`

class `spinetoolbox.widgets.custom_qtableview.CopyPasteTableView`(*parent=None*)

Bases: `PySide6.QtWidgets.QTableView`

Custom `QTableView` class with copy and paste methods.

property `copy_action`

property `paste_action`

init_copy_and_paste_actions()

Initializes copy and paste actions and connects relevant signals.

set_external_copy_and_paste_actions(*copy_action*, *paste_action*)

Sets the view to use external copy and paste actions.

Note that this doesn't connect the actions' trigger signals; the owner of the actions is responsible for handling them.

Parameters

- **copy_action** (`QAction`) – copy action
- **paste_action** (`QAction`) – paste action

delete_content(*_=False*)

Deletes content from editable indexes in current selection.

can_copy()

copy(*_=False*)

Copies current selection to clipboard in excel format.

can_paste()

paste(*_=False*)

Paste data from clipboard.

static **_read_pasted_text**(*text*)

Parses a tab separated CSV text table.

Parameters

text (*str*) – a CSV formatted table

Returns

a list of rows

Return type

list

paste_on_selection()

Pastes clipboard data on selection, but not beyond. If data is smaller than selection, repeat data to fit selection.

paste_normal()

Pastes clipboard data, overwriting cells if needed.

class `spinetoolbox.widgets.custom_qtableview.AutoFilterCopyPasteTableView`(*parent*)

Bases: [`CopyPasteTableView`](#)

Custom `QTableView` class with autofilter functionality.

Parameters**parent** (*QObject*) –**setModel**(*model*)

Disconnects the sectionPressed signal which seems to be connected by the super method. Otherwise pressing the header just selects the column.

Parameters**model** (*QAbstractItemModel*) –**_trigger_filter_menu**(*_*)

Shows current column's auto filter menu.

show_auto_filter_menu(*logical_index*)

Called when user clicks on a horizontal section header. Shows/hides the auto filter widget.

Parameters**logical_index** (*int*) – header section index

class spinetoolbox.widgets.custom_qtableview.**IndexedParameterValueTableViewBase**(*parent=None*)

Bases: [CopyPasteTableView](#)

Custom QTableView base class with copy and paste methods for indexed parameter values.

copy(*_=False*)

Copies current selection to clipboard in CSV format.

Returns

True if data was copied on the clipboard, False otherwise

Return type

bool

abstract static _read_pasted_text(*text*)

Reads CSV formatted table.

abstract paste(*_=False*)

Pastes data from clipboard to selection.

class spinetoolbox.widgets.custom_qtableview.**TimeSeriesFixedResolutionTableView**(*parent=None*)

Bases: [IndexedParameterValueTableViewBase](#)

A QTableView for fixed resolution time series table.

paste(*_=True*)

Pastes data from clipboard.

static _read_pasted_text(*text*)

Parses the given CSV table. Parsing is locale aware.

Parameters**text** (*str*) – a CSV table containing numbers**Returns**

A list of floats

Return type

list of float

_paste_to_values_column(*values*, *first_row*, *paste_length*)

Pastes data to the Values column.

Parameters

- **values** (*list*) – a list of float values to paste
- **first_row** (*int*) – index of the first row where to paste
- **paste_length** (*int*) – length of the paste selection (can be different from len(values))

Returns

A tuple (list(pasted indexes), list(pasted values))

Return type

tuple

class spinetoolbox.widgets.custom_qtableview.**IndexedValueTableView**(*parent=None*)

Bases: [*IndexedParameterValueTableViewBase*](#)

A QTableView class with for variable resolution time series and time patterns.

paste(*_=False*)

Pastes data from clipboard.

_paste_two_columns(*data_indexes*, *data_values*, *first_row*, *paste_length*)

Pastes data indexes and values.

Parameters

- **data_indexes** (*list*) – a list of data indexes (time stamps/durations)
- **data_values** (*list*) – a list of data values
- **first_row** (*int*) – first row index
- **paste_length** (*int*) – selection length for pasting

Returns

a tuple (modified model indexes, modified model values)

Return type

tuple

_paste_single_column(*values*, *first_row*, *first_column*, *paste_length*)

Pastes a single column of data.

Parameters

- **values** (*list*) – a list of data to paste (data indexes or values)
- **first_row** (*int*) – first row index
- **paste_length** (*int*) – selection length for pasting

Returns

a tuple (modified model indexes, modified model values)

Return type

tuple

static **_read_pasted_text**(*text*)

Parses a given CSV table.

Parameters

text (*str*) – a CSV table

Returns

a tuple (data indexes, data values)

Return type

tuple

class spinetoolbox.widgets.custom_qtableview.**ArrayTableView**(parent=None)

Bases: [*IndexedParameterValueTableViewBase*](#)

Custom QTableView with copy and paste methods for single column tables.

copy(_=False)

Copies current selection to clipboard in CSV format.

Returns

True if data was copied on the clipboard, False otherwise

Return type

bool

paste(_=False)

Pastes data from clipboard.

static **_read_pasted_text**(text)

Reads the first column of given CSV table.

Parameters

text (str) – a CSV table

Returns

data column

Return type

list of str

class spinetoolbox.widgets.custom_qtableview.**MapTableView**(parent=None)

Bases: [*CopyPasteTableView*](#)

Custom QTableView with copy and paste methods for map tables.

copy(_=False)

Copies current selection to clipboard in Excel compatible CSV format.

Returns

True if data was copied on the clipboard, False otherwise

Return type

bool

delete_content(_=False)

Deletes content in current selection.

paste(_=False)

Pastes data from clipboard.

Returns

True if data was pasted successfully, False otherwise

Return type

bool

static `_read_pasted_text`(*text*)

Parses a given CSV table.

Parameters

text (*str*) – a CSV table

Returns

a list of table rows

Return type

list of list

`spinetoolbox.widgets.custom_qtableview._range`(*selection*)

Returns the top left and bottom right corners of selection.

Parameters

selection (*QItemSelection*) – a list of selected *QItemSelection* objects

Returns

a tuple (top row, bottom row, left column, right column)

Return type

tuple of ints

`spinetoolbox.widgets.custom_qtableview._NOT_TIME_STAMP`

`spinetoolbox.widgets.custom_qtableview._could_be_time_stamp`(*s*)

Evaluates if given string could be a time stamp.

This is to deal with special cases that are not intended as time stamps but could end up as one by the very greedy *DateTime* constructor.

Parameters

s (*str*) – string to evaluate

Returns

True if *s* could be a time stamp, False otherwise

Return type

bool

`spinetoolbox.widgets.custom_qtableview.system_lc_numeric`()

`spinetoolbox.widgets.custom_qtextbrowser`

Class for a custom *QTextBrowser* for showing the logs and tool output.

Module Contents

Classes

<i>CustomQTextBrowser</i>	Custom <i>QTextBrowser</i> class.
<i>MonoSpaceFontTextBrowser</i>	Custom <i>QTextBrowser</i> class.

class spinetoolbox.widgets.custom_qtextbrowser.CustomQTextBrowser(*parent*)

Bases: PySide6.QtWidgets.QTextBrowser

Custom QTextBrowser class.

Parameters

parent (*QWidget*) – Parent widget

_ALL_RUNS = 'All executions'

set_toolbox(*toolbox*)

append(*text*)

Appends new text block to the end of the *original* document.

If the document contains more text blocks after the addition than a set limit, blocks are deleted at the start of the contents.

Parameters

text (*str*) – text to add

contextMenuEvent(*event*)

Reimplemented method to add a clear action into the default context menu.

Parameters

event (*QContextMenuEvent*) – Received event

clear()

_populate_executions_menu()

reset_executions_button_text()

_select_execution(*action*)

static _make_log_entry_title(*title*)

start_execution(*timestamp*)

Creates cursors (log entry points) for given items in event log.

Parameters

timestamp (*str*) – time stamp

add_log_message(*item_name*, *filter_id*, *message*)

Adds a message to an item's execution log.

Parameters

- **item_name** (*str*) – item name
- **filter_id** (*str*) – filter identifier
- **message** (*str*) – formatted message

execution_timestamps()

select_all_executions()

select_execution(*timestamp*)

_set_execution_visible(*timestamp*, *visible*)

`set_item_log_selected(selected)`

class `spinetoolbox.widgets.custom_qtextbrowser.MonoSpaceFontTextBrowser(parent)`

Bases: *CustomQTextBrowser*

Custom QTextBrowser class.

Parameters

parent (*QWidget*) – Parent widget

`spinetoolbox.widgets.custom_qtreeview`

Classes for custom QTreeView.

Module Contents

Classes

<i>CopyPasteTreeView</i>	Custom QTreeView class with copy and paste support.
<i>SourcesTreeView</i>	Custom QTreeView class for 'Sources' in Tool specification editor widget.
<i>CustomTreeView</i>	Custom QTreeView class for Tool specification editor form to enable keyPressEvent.

class `spinetoolbox.widgets.custom_qtreeview.CopyPasteTreeView(parent)`

Bases: PySide6.QtWidgets.QTreeView

Custom QTreeView class with copy and paste support.

Parameters

parent (*QWidget*) – parent widget

can_copy()

Returns True if tree view has a selection to copy from.

Returns

True if there is something to copy

Return type

bool

can_paste()

Returns whether it is possible to paste into this view.

Returns

True if pasting is possible, False otherwise

Return type

bool

copy()

Copy current selection to clipboard.

The default implementation copies the data as linefeed separated list.

Returns

True if data was successfully copied, False otherwise

Return type

bool

paste()

Pastes data to the view.

class spinetoolbox.widgets.custom_qtreeview.**SourcesTreeView**(*parent*)

Bases: PySide6.QtWidgets.QTreeView

Custom QTreeView class for 'Sources' in Tool specification editor widget.

Parameters

parent (*QWidget*) – parent widget

files_dropped**del_key_pressed****dragEnterEvent**(*event*)

Accept file and folder drops from the filesystem.

dragMoveEvent(*event*)

Accept event.

dropEvent(*event*)

Emit files_dropped signal with a list of files for each dropped url.

keyPressEvent(*event*)

Overridden method to make the view support deleting items with a delete key.

class spinetoolbox.widgets.custom_qtreeview.**CustomTreeView**(*parent*)

Bases: PySide6.QtWidgets.QTreeView

Custom QTreeView class for Tool specification editor form to enable keyPressEvent.

Parameters

parent (*QWidget*) – The parent of this view

del_key_pressed**keyPressEvent**(*event*)

Overridden method to make the view support deleting items with a delete key.

spinetoolbox.widgets.custom_qwidgets

Custom QWidgets for Filtering and Zooming.

Module Contents

Classes

<i>ElidedTextMixin</i>	
<i>UndoRedoMixin</i>	
<i>FilterWidget</i>	Filter widget class.
<i>CustomWidgetAction</i>	A QWidgetAction with custom hovering.
<i>ToolBarWidgetAction</i>	An action with a tool bar.
<i>ToolBarWidgetBase</i>	A toolbar on the right, with enough space to print a text beneath.
<i>ToolBarWidget</i>	A toolbar on the right, with enough space to print a text beneath.
<i>MenuItemToolBarWidget</i>	A menu item with a toolbar on the right.
<i>_MenuToolBar</i>	A custom tool bar for MenuItemToolBarWidget.
<i>TitleWidgetAction</i>	A titled separator.
<i>WrapLabel</i>	A QLabel that always wraps text.
<i>HyperTextLabel</i>	A QLabel that supports hyperlinks.
<i>QWizardProcessPage</i>	A QWizards page with a log. Useful for pages that need to capture the output of a process.
<i>LabelWithCopyButton</i>	A read only QLabel with a QToolButton that copies the text to clipboard.
<i>ElidedLabel</i>	A QLabel with elided text.
<i>HorizontalSpinBox</i>	
<i>PropertyQSpinBox</i>	A spinbox where undo and redo key strokes apply to the project.
<i>SelectDatabaseItemsDialog</i>	Dialog that lets selecting database items.
<i>PurgeSettingsDialog</i>	Dialog that lets selecting database items.
<i>ResizingViewMixin</i>	

```
class spinetoolbox.widgets.custom_qwidgets.ElidedTextMixin(*args, **kwargs)
```

```
    setText(text)
    _update_text(text)
    _set_text_elided(width=None)
    _elided_offset()
    text()
    resizeEvent(event)
```

```
class spinetoolbox.widgets.custom_qwidgets.UndoRedoMixin
```

```
    keyPressEvent(e)
        Overridden to catch and pass on the Undo and Redo commands when this line edit has the focus.
```


Parameters**e** (*QKeyEvent*) – Event

class spinetoolbox.widgets.custom_qwidgets.**FilterWidget**(*parent, make_filter_model, *args, **kwargs*)

Bases: PySide6.QtWidgets.QWidget

Filter widget class.

Init class.

Parameters

- **parent** (*QWidget, optional*) – parent widget
- **make_filter_model** (*Callable*) – callable that constructs the filter model
- ***args** – arguments forwarded to `make_filter_model`
- ****kwargs** – keyword arguments forwarded to `make_filter_model`

okPressed**cancelPressed****set_filter_list**(*items*)**connect_signals**()**save_state**()

Saves the state of the FilterCheckboxListModel.

reset_state()

Sets the state of the FilterCheckboxListModel to saved state.

clear_filter()

Selects all items in FilterCheckBoxListModel.

has_filter()

Returns true if any item is filtered in FilterCheckboxListModel false otherwise.

_apply_filter()

Apply current filter and save state.

_cancel_filter()

Cancel current edit of filter and set the state to the stored state.

_filter_list()

Filter list with current text.

_text_edited(*new_text*)

Callback for edit text, starts/restarts timer. Start timer after text is edited, restart timer if text is edited before last time out.

class spinetoolbox.widgets.custom_qwidgets.**CustomWidgetAction**(*parent=None*)

Bases: PySide6.QtWidgets.QWidgetAction

A QWidgetAction with custom hovering.

Class constructor.

Parameters**parent** (*QMenu*) – the widget's parent

`_handle_hovered()`

Hides other menus that might be shown in the parent widget and repaints it. This is to emulate the behavior of QAction.

class `spinetoolbox.widgets.custom_qwidgets.ToolBarWidgetAction`(*text*, *parent=None*, *compact=False*)

Bases: [*CustomWidgetAction*](#)

An action with a tool bar.

`tool_bar`

Type

QToolBar

Class constructor.

Parameters

parent (*QMenu*) – the widget’s parent

eventFilter(*obj*, *ev*)

`_handle_hovered()`

Hides other menus that might be shown in the parent widget and repaints it. This is to emulate the behavior of QAction.

class `spinetoolbox.widgets.custom_qwidgets.ToolBarWidgetBase`(*text*, *parent=None*, *io_files=None*)

Bases: `PySide6.QtWidgets.QWidget`

A toolbar on the right, with enough space to print a text beneath.

`tool_bar`

Type

QToolBar

Class constructor.

Parameters

- **text** (*str*) –
- **parent** (*QWidget*) – the widget’s parent

class `spinetoolbox.widgets.custom_qwidgets.ToolBarWidget`(*text*, *parent=None*, *io_files=None*)

Bases: [*ToolBarWidgetBase*](#)

A toolbar on the right, with enough space to print a text beneath.

`tool_bar`

Type

QToolBar

Class constructor.

Parameters

- **text** (*str*) –
- **parent** (*QWidget*) – the widget’s parent

class `spinetoolbox.widgets.custom_qwidgets.MenuItemToolBarWidget`(*text*, *parent=None*,
compact=False)

Bases: `ToolBarWidgetBase`

A menu item with a toolbar on the right.

Class constructor.

Parameters

- **text** (*str*) –
- **parent** (*QWidget*) – the widget’s parent
- **compact** (*bool*) – if True, the widget uses the minimal space

paintEvent(*event*)

Draws the menu item, then calls the `super()` method to draw the tool bar.

class `spinetoolbox.widgets.custom_qwidgets._MenuToolBar`(*parent=None*)

Bases: `PySide6.QtWidgets.QToolBar`

A custom tool bar for `MenuItemToolBarWidget`.

enabled_changed

_align_buttons()

Align all buttons to bottom so frames look good.

add_frame(*left*, *right*, *title*)

Add frame around given actions, with given title.

Parameters

- **left** (*QAction*) –
- **right** (*QAction*) –
- **title** (*str*) –

is_enabled()

addAction(*actions*)

Overriden method to customize tool buttons.

addAction(**args*, ***kwargs*)

Overriden method to customize the tool button.

sizeHint()

Make room for frames if needed.

paintEvent(*ev*)

Paint the frames.

_setup_action_button(*action*)

Customizes the `QToolButton` associated with given action:

1. Makes sure that the text honors the action’s mnemonics.
2. Installs this as event filter on the button (see `self.eventFilter()`).

Must be called everytime an action is added to the tool bar.

Parameters**action** (*QAction*) – Action to set up**actionEvent**(*ev*)

Updates `self._enabled`: True if at least one non-separator action is enabled, False otherwise. Emits `self.enabled_changed` accordingly.

eventFilter(*obj, ev*)

Installed on each action's `QToolButton`. Ignores Up and Down key press events, so they are handled by the toolbar for custom navigation.

keyPressEvent(*ev*)

Navigates over the tool bar buttons.

hideEvent(*ev*)**class** `spinetoolbox.widgets.custom_qwidgets.TitleWidgetAction`(*title, parent=None*)

Bases: `CustomWidgetAction`

A titled separator.

Class constructor.

Parameters**parent** (*QMenu*) – the widget's parent**H_MARGIN** = 5**V_MARGIN** = 2**static** `_add_line`(*widget, layout*)**isSeparator**()**class** `spinetoolbox.widgets.custom_qwidgets.WrapLabel`(*text="", parent=None*)

Bases: `PySide6.QtWidgets.QLabel`

A `QLabel` that always wraps text.

class `spinetoolbox.widgets.custom_qwidgets.HyperTextLabel`(*text="", parent=None*)

Bases: `WrapLabel`

A `QLabel` that supports hyperlinks.

class `spinetoolbox.widgets.custom_qwidgets.QWizardProcessPage`(*parent*)

Bases: `PySide6.QtWidgets.QWizardPage`

A `QWizards` page with a log. Useful for pages that need to capture the output of a process.

class `_ExecutionManager`

A descriptor that stores a `QProcessExecutionManager`. When `execution_finished` is emitted, it shows the button to copy the process log.

public_name**private_name****__set_name__**(*owner, name*)**__get__**(*obj, objtype=None*)

```
    __set__(obj, value)

msg
msg_warning
msg_error
msg_success
msg_proc
msg_proc_error
_exec_mgr
_connect_signals()
_handle_copy_clicked(_=False)
_add_msg(msg)
_add_msg_warning(msg)
_add_msg_error(msg)
_add_msg_success(msg)
isComplete()
cleanupPage()

class spinetoolbox.widgets.custom_qwidgets.LabelWithCopyButton(text="", parent=None)
    Bases: PySide6.QtWidgets.QWidget
    A read only QLabel with a QToolButton that copies the text to clipboard.

class spinetoolbox.widgets.custom_qwidgets.ElidedLabel(*args, **kwargs)
    Bases: ElidedTextMixin, PySide6.QtWidgets.QLabel
    A QLabel with elided text.

class spinetoolbox.widgets.custom_qwidgets.HorizontalSpinBox(*args, **kwargs)
    Bases: PySide6.QtWidgets.QToolBar
    valueChanged
    value()
    setMinimum(minimum)
    setValue(value, strict=False)
    _dec_value()
    _inc_value()
    _focus_line_edit()
```

class spinetoolbox.widgets.custom_qwidgets.**PropertyQSpinBox**

Bases: [*UndoRedoMixin*](#), PySide6.QtWidgets.QSpinBox

A spinbox where undo and redo key strokes apply to the project.

class spinetoolbox.widgets.custom_qwidgets.**SelectDatabaseItemsDialog**(*checked_states*,
ok_button_text=None,
parent=None)

Bases: PySide6.QtWidgets.QDialog

Dialog that lets selecting database items.

Parameters

- **checked_states** (*dict*, *optional*) – checked states for each item
- **ok_button_text** (*str*, *optional*) – alternative label for the OK button
- **parent** (*QWidget*, *optional*) – parent widget

_warn_checked_non_data_items = True

_ok_button_can_be_disabled = True

show()

Sets the OK button enabled before showing the dialog

get_checked_states()

Returns current item checked states.

Returns

mapping from database item name to checked flag

Return type

dict

_handle_check_box_state_changed(*_checked*)

class spinetoolbox.widgets.custom_qwidgets.**PurgeSettingsDialog**(*checked_states*,
ok_button_text=None,
parent=None)

Bases: [*SelectDatabaseItemsDialog*](#)

Dialog that lets selecting database items.

Parameters

- **checked_states** (*dict*, *optional*) – checked states for each item
- **ok_button_text** (*str*, *optional*) – alternative label for the OK button
- **parent** (*QWidget*, *optional*) – parent widget

_ok_button_can_be_disabled = False

class spinetoolbox.widgets.custom_qwidgets.**ResizingViewMixin**(*args, **kwargs)

rowsInserted(*parent*, *start*, *end*)

abstract _do_resize()

spinetoolbox.widgets.datetime_editor

An editor widget for editing datetime database (relationship) parameter values.

Module Contents**Classes**

<i>DatetimeEditor</i>	An editor widget for DateTime type parameter values.
-----------------------	--

Functions

<i>_QDateTime_to_datetime(dt)</i>	Converts a QDateTime object to Python's datetime.datetime type.
<i>_datetime_to_QDateTime(dt)</i>	Converts Python's datetime.datetime object to QDateTime.

`spinetoolbox.widgets.datetime_editor._QDateTime_to_datetime(dt)`

Converts a QDateTime object to Python's datetime.datetime type.

`spinetoolbox.widgets.datetime_editor._datetime_to_QDateTime(dt)`

Converts Python's datetime.datetime object to QDateTime.

class `spinetoolbox.widgets.datetime_editor.DatetimeEditor`(*parent=None*)

Bases: `PySide6.QtWidgets.QWidget`

An editor widget for DateTime type parameter values.

parent

a parent widget

Type

`QWidget`

_change_datetime(*new_datetime*)

Updates the internal DateTime value

set_value(*value*)

Sets the value to be edited.

value()

Returns the editor's current value.

spinetoolbox.widgets.duration_editor

An editor widget for editing duration database (relationship) parameter values.

Module Contents

Classes

<i>DurationEditor</i>	An editor widget for Duration type parameter values.
-----------------------	--

class spinetoolbox.widgets.duration_editor.**DurationEditor**(parent=None)

Bases: PySide6.QtWidgets.QWidget

An editor widget for Duration type parameter values.

parent

a parent widget

Type

QWidget

_change_duration()

Updates the value being edited.

set_value(value)

Sets the value for editing.

value()

Returns the current Duration.

spinetoolbox.widgets.indexed_value_table_context_menu

Context menus for parameter value editor widgets.

Module Contents

Classes

<i>ContextMenuBase</i>	Context menu base for parameter value editor tables.
<i>ArrayTableContextMenu</i>	Context menu for array editor tables.
<i>IndexedValueTableContextMenu</i>	Context menu for time series and time pattern editor tables.
<i>MapTableContextMenu</i>	Context menu for map editor tables.

Functions

<code>_unique_row_ranges(selections)</code>	Merged ranges in given selections to unique ranges.
<code>_unique_column_ranges(selections)</code>	Merged ranges in given selections to unique ranges.
<code>_merge_intervals(intervals)</code>	Merges given intervals if they overlap.

Attributes

<code>_INSERT_SINGLE_COLUMN_AFTER</code>
<code>_INSERT_SINGLE_ROW_AFTER</code>
<code>_INSERT_MULTIPLE_COLUMNS_AFTER</code>
<code>_INSERT_MULTIPLE_ROWS_AFTER</code>
<code>_INSERT_SINGLE_COLUMN_BEFORE</code>
<code>_INSERT_SINGLE_ROW_BEFORE</code>
<code>_INSERT_MULTIPLE_COLUMNS_BEFORE</code>
<code>_INSERT_MULTIPLE_ROWS_BEFORE</code>
<code>_OPEN_EDITOR</code>
<code>_PLOT</code>
<code>_PLOT_IN_WINDOW</code>
<code>_REMOVE_COLUMNS</code>
<code>_REMOVE_ROWS</code>
<code>_TRIM_COLUMNS</code>

```
spinetoolbox.widgets.indexed_value_table_context_menu._INSERT_SINGLE_COLUMN_AFTER =
'Insert column after'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._INSERT_SINGLE_ROW_AFTER = 'Insert
row after'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._INSERT_MULTIPLE_COLUMNS_AFTER =
'Insert columns after...'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._INSERT_MULTIPLE_ROWS_AFTER =
'Insert rows after...'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._INSERT_SINGLE_COLUMN_BEFORE =
'Insert column before'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._INSERT_SINGLE_ROW_BEFORE = 'Insert row before'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._INSERT_MULTIPLE_COLUMNS_BEFORE = 'Insert columns before...'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._INSERT_MULTIPLE_ROWS_BEFORE = 'Insert rows before...'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._OPEN_EDITOR = 'Edit...'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._PLOT = 'Plot...'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._PLOT_IN_WINDOW = 'Plot in window'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._REMOVE_COLUMNS = 'Remove columns'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._REMOVE_ROWS = 'Remove rows'
```

```
spinetoolbox.widgets.indexed_value_table_context_menu._TRIM_COLUMNS = 'Trim columns'
```

```
class spinetoolbox.widgets.indexed_value_table_context_menu.ContextMenuBase(table_view,  
                                                                           position)
```

Bases: PySide6.QtWidgets.QMenu

Context menu base for parameter value editor tables.

Parameters

- **table_view** (*QTableView*) – the view where the menu is invoked
- **position** (*QPoint*) – menu's position on the table view

_add_default_actions()

Adds default actions to the menu.

_first_row()

Returns the first selected row.

Returns

index to the first row

Return type

int

_insert_multiple_rows_after()

Prompts for row count, then inserts new rows below the current selection.

_insert_multiple_rows_before()

Prompts for row count, then inserts new rows above the current selection.

_insert_single_row_after()

Inserts a single row below the current selection.

_insert_single_row_before()

Inserts a single row above the current selection.

_last_row()

Returns the last selected row.

Returns

index to the last row

Return type

int

_prompt_row_count()

Prompts for number of rows to insert.

Returns

number of rows

Return type

int

_remove_rows()

Removes selected rows.

```
class spinetoolbox.widgets.indexed_value_table_context_menu.ArrayTableContextMenu(editor,
                                                                                   ta-
                                                                                   ble_view,
                                                                                   position)
```

Bases: [*ContextMenuBase*](#)

Context menu for array editor tables.

Parameters

- **editor** ([*ArrayEditor*](#)) – array editor widget
- **table_view** ([*QTableView*](#)) – the view where the menu is invoked
- **position** ([*QPoint*](#)) – menu’s position

_show_value_editor()

Opens the value element editor.

```
class spinetoolbox.widgets.indexed_value_table_context_menu.IndexedValueTableContextMenu(table_view,
                                                                                          po-
                                                                                          si-
                                                                                          tion)
```

Bases: [*ContextMenuBase*](#)

Context menu for time series and time pattern editor tables.

Parameters

- **table_view** ([*QTableView*](#)) – the view where the menu is invoked
- **position** ([*QPoint*](#)) – menu’s position

```
class spinetoolbox.widgets.indexed_value_table_context_menu.MapTableContextMenu(editor,
                                                                                   table_view,
                                                                                   position)
```

Bases: [*ContextMenuBase*](#)

Context menu for map editor tables.

Parameters

- **editor** ([MapEditor](#)) – map editor widget
- **table_view** ([QTableView](#)) – the view where the menu is invoked
- **position** ([QPoint](#)) – table cell index

`_first_column()`

Returns the first selected column.

Returns

index to the first column

Return type

int

`_insert_multiple_columns_after()`

Prompts for column count, then inserts new columns right from the current selection.

`_insert_multiple_columns_before()`

Prompts for column count, then inserts new columns left from the current selection.

`_insert_single_column_before()`

Inserts a single column left from the current selection.

`_insert_single_column_after()`

Inserts a single column right from the current selection.

`_last_column()`

Returns the last selected column.

Returns

index to the last column

Return type

int

`_prompt_column_count()`

Prompts for number of column to insert.

Returns

number of columns

Return type

int

`_remove_columns()`

Removes selected columns

`_show_value_editor()`

Opens the value element editor.

`_plot(checked=False)`

Plots current indexes.

`_plot_in_window(action)`

Plots the selected cells in an existing window.

`_trim_columns()`

Removes excessive columns from the table.

`spinetoolbox.widgets.indexed_value_table_context_menu._unique_row_ranges(selections)`

Merged ranges in given selections to unique ranges.

Parameters

selections (*list of QItemSelectionRange*) – selected ranges

Returns

a list of [first_row, last_row] ranges

Return type

list of list

`spinetoolbox.widgets.indexed_value_table_context_menu._unique_column_ranges(selections)`

Merged ranges in given selections to unique ranges.

Parameters

selections (*list of QItemSelectionRange*) – selected ranges

Returns

a list of [first_row, last_row] ranges

Return type

list of list

`spinetoolbox.widgets.indexed_value_table_context_menu._merge_intervals(intervals)`

Merges given intervals if they overlap.

Parameters

intervals (*list of list*) – a list of intervals in the form [first, last]

Returns

merged intervals in the form [first, last]

Return type

list of list

`spinetoolbox.widgets.install_julia_wizard`

Classes for custom QDialogs for julia setup.

Module Contents

Classes

<i>_PageId</i>	Enum where members are also (and must be) ints
<i>InstallJuliaWizard</i>	A wizard to install julia
<i>JillNotFoundPage</i>	
<i>IntroPage</i>	
<i>SelectDirsPage</i>	
<i>InstallJuliaPage</i>	A QWizards page with a log. Useful for pages that need to capture the output of a process.
<i>SuccessPage</i>	
<i>FailurePage</i>	

Attributes

<i>jill_install</i>

`spinetoolbox.widgets.install_julia_wizard.jill_install`

class `spinetoolbox.widgets.install_julia_wizard._PageId`

Bases: `enum.IntEnum`

Enum where members are also (and must be) ints

Initialize self. See `help(type(self))` for accurate signature.

INTRO

SELECT_DIRS

INSTALL

SUCCESS

FAILURE

class `spinetoolbox.widgets.install_julia_wizard.InstallJuliaWizard(parent)`

Bases: `PySide6.QtWidgets.QWizard`

A wizard to install julia

Initialize class.

Parameters

parent (*QWidget*) – the parent widget (SettingsWidget)

julia_exe_selected

```
    set_julia_exe()

    accept()

class spinetoolbox.widgets.install_julia_wizard.JillNotFoundPage(parent)
    Bases: PySide6.QtWidgets.QWizardPage

class spinetoolbox.widgets.install_julia_wizard.IntroPage(parent)
    Bases: PySide6.QtWidgets.QWizardPage
    nextId()

class spinetoolbox.widgets.install_julia_wizard.SelectDirsPage(parent)
    Bases: PySide6.QtWidgets.QWizardPage
    initializePage()
    _select_install_dir()
    _select_symlink_dir()
    nextId()

class spinetoolbox.widgets.install_julia_wizard.InstallJuliaPage(parent)
    Bases: spinetoolbox.widgets.custom\_qwidgets.QWizardProcessPage
    A QWizards page with a log. Useful for pages that need to capture the output of a process.
    cleanupPage()
    initializePage()
    _handle_julia_install_finished(ret)
    nextId()

class spinetoolbox.widgets.install_julia_wizard.SuccessPage(parent)
    Bases: PySide6.QtWidgets.QWizardPage
    initializePage()
    nextId()

class spinetoolbox.widgets.install_julia_wizard.FailurePage(parent)
    Bases: PySide6.QtWidgets.QWizardPage
    initializePage()
    nextId()
```

[spinetoolbox.widgets.jump_properties_widget](#)

Contains jump properties widget's business logic.

Module Contents

Classes

*JumpPropertiesWidget*Widget for jump link properties.

```
class spinetoolbox.widgets.jump_properties_widget.JumpPropertiesWidget(toolbox,
                                                                    base_color=None)
```

Bases: *spinetoolbox.widgets.properties_widget.PropertiesWidgetBase*

Widget for jump link properties.

Parameters

toolbox (*ToolboxUI*) – The toolbox instance where this widget should be embedded

_load_condition_into_ui (*condition*)

_make_condition_from_ui ()

_change_condition ()

Stores jump condition to link.

_show_tool_spec_form (*_checked=False*)

_set_save_script_button_enabled ()

_update_add_args_button_enabled (*_selected, _deselected*)

_do_update_add_args_button_enabled ()

_update_remove_args_button_enabled (*_selected, _deselected*)

_do_update_remove_args_button_enabled ()

_populate_cmd_line_args_model ()

_push_update_cmd_line_args_command (*cmd_line_args*)

_remove_arg (*_=False*)

_add_args (*_=False*)

set_link (*jump*)

Hooks the widget to given jump link, so that user actions are reflected in the jump link's configuration.

Parameters

jump (*LoggingJump*) – link to hook into

unset_link ()

Releases the widget from any links.

set_condition (*jump, condition*)

update_cmd_line_args (*jump, cmd_line_args*)

spinetoolbox.widgets.jupyter_console_widget

Class for a custom RichJupyterWidget that can run Tool instances.

Module Contents**Classes**

<i>JupyterConsoleWidget</i>	Base class for all embedded console widgets that can run tool instances.
-----------------------------	--

Attributes

traitlets_logger

asyncio_logger

`spinetoolbox.widgets.jupyter_console_widget.traitlets_logger`

`spinetoolbox.widgets.jupyter_console_widget.asyncio_logger`

class `spinetoolbox.widgets.jupyter_console_widget.JupyterConsoleWidget`(*toolbox, kernel_name, owner=None*)

Bases: `qtconsole.rich_jupyter_widget.RichJupyterWidget`

Base class for all embedded console widgets that can run tool instances.

Parameters

- **toolbox** (`ToolboxUI`) – QMainWindow instance
- **kernel_name** (*str*) – Kernel name to start
- **owner** (`ProjectItem`, *NoneType*) – Item that owns the console.

property `owner_names`

console_closed

request_start_kernel(*conda=False*)

Requests Spine Engine to launch a kernel manager for the given kernel_name.

Parameters

conda (*bool*) – Conda kernel or not

Returns

Path to connection file if kernel manager was launched successfully, None otherwise

Return type

str or None

release_exec_mgr_resources()

Closes `_io.TextIOWrapper` files.

`_handle_kernel_started_msg(msg)`

Handles the response message from KernelExecutionManager.

Parameters

msg (*dict*) – Message with item_name, type, etc. keys

Returns

Path to a connection file if engine started the requested kernel manager successfully, None otherwise.

Return type

str or None

`_execute(source, hidden)`

Catches exit or similar commands and closes the console immediately if user so chooses.

`insert_text_to_console(msg)`

Inserts given message to console.

Parameters

msg (*str*) – Text to insert

`set_connection_file(connection_file)`

Sets connection file obtained from engine to this console.

Parameters

connection_file (*str*) – Path to a connection file obtained from a running kernel manager.

`connect_to_kernel()`

Connects a local kernel client to a kernel manager running on Spine Engine.

`request_restart_kernel_manager()`

Restarts kernel manager on engine and connects a new kernel client to it.

`request_shutdown_kernel_manager()`

Sends a shutdown kernel manager request to engine.

`name()`

Returns console name for display purposes.

`shutdown_kernel_client()`

Shuts down local kernel client.

`dragEnterEvent(e)`

Rejects dropped project items.

`_context_menu_make(pos)`

Reimplemented to add actions to console context-menus.

`copy_input()`

Copies only input.

`closeEvent(e)`

Catches close event to shut down the kernel client and sends a signal to Toolbox to request Spine Engine to shut down the kernel manager.

spinetoolbox.widgets.kernel_editor

Widget for showing the progress of making a Julia or Python kernel.

Module Contents**Classes**

<i>KernelEditorBase</i>	Base class for kernel editors.
<i>MiniPythonKernelEditor</i>	A Simple Python kernel maker. The Python executable path is passed in
<i>MiniJuliaKernelEditor</i>	A Simple Julia Kernel maker. The julia exe and project are passed in

Functions

<i>format_event_message</i> (msg_type, message[, show_datetime])	Formats message for the kernel editor text browser.
<i>format_process_message</i> (msg_type, message)	Formats process message for the kernel editor text browser.

class spinetoolbox.widgets.kernel_editor.**KernelEditorBase**(parent, python_or_julia)

Bases: PySide6.QtWidgets.QDialog

Base class for kernel editors.

Parameters

- **parent** (*SettingsWidget*) – Parent widget
- **python_or_julia** (*str*) – kernel type; valid values: “julia”, “python”

connect_signals()

Connects signals to slots.

_show_close_button(failed=False)

make_kernel()

abstract _do_make_kernel()

new_kernel_name()

Returns the new kernel name after it's been created.

_solve_new_kernel_name()

Finds out the new kernel name after a new kernel has been created.

check_options(prgm, kernel_name, display_name, python_or_julia)

Checks that user options are valid before advancing with kernel making.

Parameters

- **prgm** (*str*) – Full path to Python or Julia program

- **kernel_name** (*str*) – Kernel name
- **display_name** (*str*) – Kernel display name
- **python_or_julia** (*str*) – Either ‘python’ or ‘julia’

Returns

True if all user input is valid for making a new kernel, False otherwise

Return type

bool

abstract _python_kernel_name()

abstract _python_kernel_display_name()

_python_interpreter_name()

make_python_kernel (*checked=False*)

Makes a new Python kernel. Offers to install ipykernel package if it is missing from the selected Python environment. Overwrites existing kernel with the same name if this is ok by user.

static is_package_installed (*python_path, package_name*)

Checks if given package is installed to given Python environment.

Parameters

- **python_path** (*str*) – Full path to selected Python interpreter
- **package_name** (*str*) – Package name

Returns

True if installed, False if not

Return type

(bool)

start_package_install_process (*python_path, package_name*)

Starts installing the given package using pip.

Parameters

- **python_path** (*str*) – Full path to selected Python interpreter
- **package_name** (*str*) – Package name to install using pip

handle_package_install_process_finished (*retval*)

Handles installing package finished.

Parameters

retval (*int*) – Process return value. 0: success, !=0: failure

start_kernelspec_install_process (*prgm, k_name, d_name*)

Installs kernel specifications for the given Python environment. Runs e.g. this command in QProcess

python -m ipykernel install --user --name python-X.Y --display-name PythonX.Y

Creates new kernel specs into %APPDATA%\jupyterkernels. Existing directory will be overwritten.

Note: We cannot use --sys.prefix here because if we have selected to create a kernel for some other python that was used in launching the app, the kernel will be created into a location that is not discoverable by jupyter and hence not by Spine Toolbox. E.g. when sys.executable is C:\Python36python.exe, and we have selected that as the python for Spine Toolbox (Settings->Tools->Python interpreter is empty), creating a

kernel with `--sys-prefix` creates kernel specs into `C:\Python36\share\jupyter\kernels\python-3.6`. This is ok and the kernel spec is discoverable by jupyter and Spine Toolbox.

BUT when `sys.executable` is `C:\Python36\python.exe`, and we have selected another python for Spine Toolbox (Settings->Tools->Python interpreter is `C:\Python38\python.exe`), creating a kernel with `--sys-prefix` creates a kernel into `C:\Python38\share\jupyter\kernels\python-3.8-sys-prefix`. This is not discoverable by jupyter nor Spine Toolbox. You would need to start the app using `C:\Python38\python.exe` to see and use that kernel spec.

Using `--user` option instead, creates kernel specs that are discoverable by any python that was used in starting Spine Toolbox.

Parameters

- **prgm** (*str*) – Full path to Python interpreter for which the kernel is created
- **k_name** (*str*) – Kernel name
- **d_name** (*str*) – Kernel display name

handle_kernelspec_install_process_finished(*retval*)

Handles case when the process for installing the kernel has finished.

Parameters

retval (*int*) – Process return value. 0: success, !0: failure

abstract _julia_kernel_name()

_julia_executable()

_julia_project()

make_julia_kernel(*checked=False*)

Makes a new Julia kernel. Offers to install IJulia package if it is missing from the selected Julia project. Overwrites existing kernel with the same name if this is ok by user.

_is_rebuild_ijulia_needed()

is_ijulia_installed(*program, project*)

Checks if IJulia is installed for the given project. Note: Trying command ‘using IJulia’ does not work since it automatically tries loading it from the `LOAD_PATH` if not it’s not found in the active project.

Returns

0 when process failed to start, 1 when IJulia is installed, 2 when IJulia is not installed.

Return type

int

start_ijulia_install_process(*julia, project*)

Starts installing IJulia package to given Julia project.

Parameters

- **julia** (*str*) – Full path to selected Julia executable
- **project** (*str*) – Julia project (e.g. dir path or ‘@.’, or ‘.’)

handle_ijulia_install_finished(*ret*)

Runs when IJulia install process finishes.

Parameters

ret (*int*) – Process return value. 0: success, !0: failure

start_ijulia_rebuild_process(*program, project*)

Starts rebuilding IJulia.

handle_ijulia_rebuild_finished(*ret*)

Runs when IJulia rebuild process finishes.

Parameters

ret (*int*) – Process return value. 0: success, !0: failure

start_ijulia_installkernel_process(*program, project, kernel_name*)

Installs the kernel using IJulia.installkernel function. Given *kernel_name* is the new kernel DISPLAY name prefix. IJulia strips the whitespace and uncapitalizes this to make the kernel name automatically. Julia version is concatenated to both kernel and display names automatically (This cannot be changed).

handle_installkernel_process_finished(*retval*)

Checks whether the IJulia.installkernel process finished successfully.

Parameters

retval (*int*) – Process return value. 0: success, !0: failure

restore_dialog_dimensions()

Restore widget location, dimensions, and state from previous session.

add_message(*msg*)

Append regular message to kernel editor text browser.

Parameters

msg (*str*) – String written to QTextBrowser

add_success_message(*msg*)

Append message with green text color to kernel editor text browser.

Parameters

msg (*str*) – String written to QTextBrowser

add_error_message(*msg*)

Append message with red color to kernel editor text browser.

Parameters

msg (*str*) – String written to QTextBrowser

add_warning_message(*msg*)

Append message with yellow (golden) color to kernel editor text browser.

Parameters

msg (*str*) – String written to QTextBrowser

add_process_message(*msg*)

Writes message from stdout to kernel editor text browser.

Parameters

msg (*str*) – String written to QTextBrowser

add_process_error_message(*msg*)

Writes message from stderr to kernel editor text browser.

Parameters

msg (*str*) – String written to QTextBrowser

_save_ui()

class spinetoolbox.widgets.kernel_editor.**MiniPythonKernelEditor**(*parent, python_exe*)

Bases: [*KernelEditorBase*](#)

A Simple Python kernel maker. The Python executable path is passed in the constructor, then calling `make_kernel` starts the process.

Parameters

- **parent** ([*SettingsWidget*](#)) – Parent widget
- **python_or_julia** (*str*) – kernel type; valid values: “julia”, “python”

abstract `_julia_kernel_name()`

`_python_kernel_name()`

`_python_kernel_display_name()`

`_do_make_kernel()`

handle_kernelspec_install_process_finished(*retval*)

Handles case when the process for installing the kernel has finished.

Parameters

- **retval** (*int*) – Process return value. 0: success, !=0: failure

set_kernel_name()

Retrieves Python version in a subprocess and makes a kernel name based on it.

class spinetoolbox.widgets.kernel_editor.**MiniJuliaKernelEditor**(*parent, julia_exe, julia_project*)

Bases: [*KernelEditorBase*](#)

A Simple Julia Kernel maker. The julia exe and project are passed in the constructor, then calling `make_kernel` starts the process.

Parameters

- **parent** ([*SettingsWidget*](#)) – Parent widget
- **python_or_julia** (*str*) – kernel type; valid values: “julia”, “python”

`_julia_kernel_name()`

abstract `_python_kernel_name()`

abstract `_python_kernel_display_name()`

`_do_make_kernel()`

handle_installkernel_process_finished(*retval*)

Checks whether the IJulia.installkernel process finished successfully.

Parameters

- **retval** (*int*) – Process return value. 0: success, !=0: failure

spinetoolbox.widgets.kernel_editor.**format_event_message**(*msg_type, message, show_datetime=True*)

Formats message for the kernel editor text browser. This is a copy of `helpers.format_event_message()` but the colors have been edited for a text browser with a white background.

spinetoolbox.widgets.kernel_editor.**format_process_message**(*msg_type, message*)

Formats process message for the kernel editor text browser.

spinetoolbox.widgets.link_properties_widget

Link properties widget.

Module Contents

Classes

<i>LinkPropertiesWidget</i>	Widget for connection link properties.
-----------------------------	--

```
class spinetoolbox.widgets.link_properties_widget.LinkPropertiesWidget(toolbox,
                                                                    base_color=None)
```

Bases: *spinetoolbox.widgets.properties_widget.PropertiesWidgetBase*

Widget for connection link properties.

Parameters

toolbox (*ToolboxUI*) – The toolbox instance where this widget should be embedded

set_link(*connection*)

Hooks the widget to given link, so that user actions are reflected in the link’s filter configuration.

Parameters

connection (*LoggingConnection*) –

unset_link()

Releases the widget from any links.

_handle_auto_check_filters_state_changed(*checked*)

Updates filters’ auto enabled setting.

Parameters

checked (*bool*) – True if the checkbox is checked, False otherwise

set_auto_check_filters_state(*checked*)

Sets the checked status of filter default online status check box

Parameters

checked (*bool*) – True if the checkbox is checked

_populate_filter_validation_menu()

Adds actions to filter validation menu.

Returns

menu actions

Return type

dict

_update_filter_validation_options(*checked*)

_handle_write_index_value_changed(*value*)

_handle_use_datapackage_state_changed(*_state*)

_handle_use_memory_db_state_changed(*_state*)

`_handle_purge_before_writing_state_changed(_state)`

`_open_purge_settings_dialog(_=False)`
 Opens the purge settings dialog.

`_handle_purge_settings_changed()`
 Pushes a command that sets new purge settings onto undo stack.

`_clean_up_purge_settings_dialog()`
 Cleans things related to purge settings dialog.

`load_connection_options()`

`spinetoolbox.widgets.map_editor`

An editor widget for editing a map type parameter values.

Module Contents

Classes

<i>MapEditor</i>	A widget for editing maps.
------------------	----------------------------

class `spinetoolbox.widgets.map_editor.MapEditor`(*parent=None*)

Bases: `PySide6.QtWidgets.QWidget`

A widget for editing maps.

parent

Type

`QWidget`

`_convert_leaves(_)`

`_show_table_context_menu(position)`

Opens table context menu.

Parameters

position (`QPoint`) – menu's position

`set_value(value)`

Sets the parameter_value to be edited.

`value()`

Returns the parameter_value currently being edited.

`open_value_editor(index)`

Opens value editor dialog for given map model index.

Parameters

index (`QModelIndex`) – index

`_open_header_editor(column)`

`spinetoolbox.widgets.map_value_editor`

An editor dialog for map indexes and values.

Module Contents

Classes

<i>MapValueEditor</i>	Dialog for editing parameter values in Map value editor.
-----------------------	--

class `spinetoolbox.widgets.map_value_editor.MapValueEditor`(*index*, *parent=None*)

Bases: `spinetoolbox.widgets.parameter_value_editor_base.ParameterValueEditorBase`

Dialog for editing parameter values in Map value editor.

Parameters

- **index** (*QModelIndex*) – an index to a parameter_value in parent_model
- **parent** (*QWidget*, *optional*) – a parent widget

_set_data(*value*)

See base class.

`spinetoolbox.widgets.multi_tab_spec_editor`

Contains the MultiTabSpecEditor class.

Module Contents

Classes

<i>MultiTabSpecEditor</i>	A main window that has a tab widget as its central widget.
---------------------------	--

class `spinetoolbox.widgets.multi_tab_spec_editor.MultiTabSpecEditor`(*toolbox*, *item_type*)

Bases: `spinetoolbox.widgets.multi_tab_window.MultiTabWindow`

A main window that has a tab widget as its central widget.

Parameters

- **qsettings** (*QSettings*) – Toolbox settings
- **settings_group** (*str*) – this window's settings group in qsettings

property `new_tab_title`

Title for new tabs.

_make_other()

Creates a new MultiTabWindow of this type.

Returns

new MultiTabWindow

Return type

MultiTabWindow

_make_new_tab(*args, **kwargs)

Creates a new tab.

Parameters

- ***args** – positional arguments needed to make a new tab
- ****kwargs** – keyword arguments needed to make a new tab

_connect_tab_signals(tab)

Connects spec editor window (tab) signals.

Parameters

tab (*SpecificationEditorWindowBase*) – Specification editor window

Returns

True if ok, False otherwise

Return type

bool

_disconnect_tab_signals(index)

Disconnects signals of spec editor window (tab) in given index.

Parameters

index (*int*) – Tab index

Returns

True if ok, False otherwise

Return type

bool

show_plus_button_context_menu(global_pos)

Opens a context menu for the tool bar.

Parameters

global_pos (*QPoint*) – menu position on screen

spinetoolbox.widgets.multi_tab_window

Contains the MultiTabWindow and TabBarPlus classes.

Module Contents

Classes

<i>MultiTabWindow</i>	A main window that has a tab widget as its central widget.
<i>TabBarPlus</i>	Tab bar that has a plus button floating to the right of the tabs.

class spinetoolbox.widgets.multi_tab_window.**MultiTabWindow**(*qsettings*, *settings_group*)

Bases: PySide6.QtWidgets.QMainWindow

A main window that has a tab widget as its central widget.

Parameters

- **qsettings** (*QSettings*) – Toolbox settings
- **settings_group** (*str*) – this window’s settings group in *qsettings*

property *accepting_new_tabs*

property *new_tab_title*

Title for new tabs.

_tab_slots

_other_editor_windows

abstract *_make_other*()

Creates a new MultiTabWindow of this type.

Returns

new MultiTabWindow

Return type

MultiTabWindow

others()

List of other MultiTabWindows of the same type.

Returns

other MutliTabWindows windows

Return type

list of MultiTabWindow

abstract *_make_new_tab*(*args, **kwargs)

Creates a new tab.

Parameters

- ***args** – positional arguments neede to make a new tab
- ****kwargs** – keyword arguments needed to make a new tab

abstract *show_plus_button_context_menu*(*global_pos*)

Opens a context menu for the tool bar.

Parameters

global_pos (*QPoint*) – menu position on screen

connect_signals()

Connects window's signals.

name()

Generates name based on the current tab and total tab count.

Returns

a name

Return type

str

all_tabs()

Iterates over tab contents widgets.

Yields

QWidget – tab contents widget

add_new_tab(*args, **kwargs)

Creates a new tab and adds it at the end of the tab bar.

Parameters

- ***args** – parameters forwarded to `MutliTabWindow._make_new_tab()`
- ****kwargs** – parameters forwarded to `MultiTabwindow._make_new_tab()`

insert_new_tab(index, *args, **kwargs)

Creates a new tab and inserts it at the given index.

Parameters

- **index** (*int*) – insertion point index
- ***args** – parameters forwarded to `MutliTabWindow._make_new_tab()`
- ****kwargs** – parameters forwarded to `MultiTabwindow._make_new_tab()`

_add_connect_tab(tab, text)

Appends a new tab and connects signals.

Parameters

- **tab** (*QWidget*) – tab contents widget
- **text** (*str*) – appended tab title

_insert_connect_tab(index, tab, text)

Inserts a new tab and connects signals.

Parameters

- **index** (*int*) – insertion point index
- **tab** (*QWidget*) – tab contents widget
- **text** (*str*) – inserted tab title

_remove_disconnect_tab(index)

Disconnects and removes a tab.

Parameters

index (*int*) – tab index

_connect_tab(*index*)

Connects signals from a tab contents widget.

Parameters

index (*int*) – tab index

_connect_tab_signals(*tab*)

Connects signals from a tab contents widget.

Parameters

tab (*QWidget*) – tab contents widget

Returns

True if signals were connected successfully, False otherwise

Return type

bool

_disconnect_tab_signals(*index*)

Disconnects signals from given tab.

Parameters

index (*int*) – tab index

Returns

True if signals were disconnected successfully, False otherwise

Return type

bool

_handle_tab_window_title_changed(*tab*, *title*)

Updates tab's title.

Parameters

- **tab** (*QWidget*) – tab's content widget
- **title** (*str*) – new tab title; if empty, one will be generated

_take_tab(*index*)

Removes a tab and returns its contents.

Parameters

index (*int*) – tab index

Returns

widget the tab was holding and tab's title

Return type

tuple

move_tab(*index*, *other=None*)

Moves a tab to another MultiTabWindow.

Parameters

- **index** (*int*) – tab index
- **other** ([MultiTabWindow](#), *optional*) – target window; if None, creates a new window

detach(*index*, *hot_spot*, *offset*=0)

Detaches the tab at given index into another MultiTabWindow window and starts dragging it.

Parameters

- **index** (*int*) –
- **hot_spot** (*QPoint*) –
- **offset** (*int*) –

start_drag(*hot_spot*, *offset*=0)

Starts dragging a detached tab.

Parameters

- **hot_spot** (*QPoint*) – The anchor point of the drag in widget coordinates.
- **offset** (*int*) – Horizontal offset of the tab in the bar.

_frame_height()

Calculates the total ‘thickness’ of window frame in vertical direction.

Returns

frame height

Return type

int

timerEvent(*event*)

Performs the drag, i.e., moves the window with the mouse cursor. As soon as the mouse hovers the tab bar of another MultiTabWindow, reattaches it.

mouseReleaseEvent(*event*)

Stops the drag. This only happens when the detached tab is not reattached to another window.

reattach(*index*, *tab*, *text*)

Reattaches a tab that has been dragged over this window’s tab bar.

Parameters

- **index** (*int*) – Index in this widget’s tab bar where the detached tab has been dragged.
- **tab** (*QWidget*) – The widget in the tab being dragged.
- **text** (*str*) – The title of the tab.

handle_close_request_from_tab()

Catches close event triggered by a QAction in tab’s QToolBar. Calls QTabWidgets close tabs method to ensure that the last closed tab closes the editor window.

_close_tab(*index*)

Closes the tab at index.

Parameters

index (*int*) – tab index

set_current_tab(*tab*)

Sets the tab that is shown on the window.

Parameters

tab (*QWidget*) – tab’s contents widget

make_context_menu(*index*)

Creates a context menu for given tab.

Parameters

index (*int*) – tab index

Returns

context menu or None if tab was not found

Return type

QMenu

restore_ui()

Restore UI state from previous session.

save_window_state()

Save window state parameters (size, position, state) via QSettings.

closeEvent(*event*)

class spinetoolbox.widgets.multi_tab_window.**TabBarPlus**(*parent*)

Bases: PySide6.QtWidgets.QTabBar

Tab bar that has a plus button floating to the right of the tabs.

Parameters

parent ([MultiSpineDBEditor](#)) –

plus_clicked

resizeEvent(*event*)

Sets the dimension of the plus button. Also, makes the tab bar as wide as the parent.

tabLayoutChange()

_move_plus_button()

Places the plus button at the right of the last tab.

mousePressEvent(*event*)

Registers the position of the press, in case we need to detach the tab.

mouseMoveEvent(*event*)

Detaches a tab either if the user moves beyond the limits of the tab bar, or if it's the only one.

_send_release_event(*pos*)

Sends a mouse release event at given position in local coordinates. Called just before detaching a tab.

Parameters

pos (*QPoint*) –

mouseReleaseEvent(*event*)

start_dragging(*index*)

Stars dragging the given index. This happens when a detached tab is reattached to this bar.

Parameters

index (*int*) –

index_under_mouse()

Returns the index under the mouse cursor, or None if the cursor isn't over the tab bar. Used to check for drop targets.

Returns

int or NoneType

contextMenuEvent(event)**spinetoolbox.widgets.notification**

Contains a notification widget.

Module Contents**Classes**

<i>Notification</i>	Custom pop-up notification widget with fade-in and fade-out effect.
<i>ButtonNotification</i>	A notification with a button.
<i>LinkNotification</i>	A notification that may have a link.
<i>ChangeNotifier</i>	
param parent	

```
class spinetoolbox.widgets.notification.Notification(parent, txt,
                                                    anim_duration=_FADE_IN_OUT_DURATION,
                                                    life_span=None, word_wrap=True,
                                                    corner=Qt.TopRightCorner)
```

Bases: PySide6.QtWidgets.QFrame

Custom pop-up notification widget with fade-in and fade-out effect.

Parameters

- **parent** (*QWidget*) – Parent widget
- **txt** (*str*) – Text to display in notification
- **anim_duration** (*int*) – Duration of the animation in msec
- **life_span** (*int*) – How long does the notification stays in place in msec
- **word_wrap** (*bool*) –
- **corner** (*Qt.Corner*) –

_FADE_IN_OUT_DURATION = 500

opacity

show()

Shows widget and moves it to the selected corner of the parent widget.

get_opacity()

opacity getter.

set_opacity(*op*)

opacity setter.

update_opacity(*value*)

Updates graphics effect opacity.

start_self_destruction()

Starts fade-out animation and closing of the notification.

enterEvent(*e*)

Pauses timer as the mouse hovers the notification.

leaveEvent(*e*)

Starts self destruction after the mouse leaves the notification.

remaining_time()

```
class spinetoolbox.widgets.notification.ButtonNotification(*args, button_text="",
                                                         button_slot=None, **kwargs)
```

Bases: [Notification](#)

A notification with a button.

Parameters

- **parent** (*QWidget*) – Parent widget
- **txt** (*str*) – Text to display in notification
- **anim_duration** (*int*) – Duration of the animation in msec
- **life_span** (*int*) – How long does the notification stays in place in msec
- **word_wrap** (*bool*) –
- **corner** (*Qt.Corner*) –

```
class spinetoolbox.widgets.notification.LinkNotification(*args, open_link=None, **kwargs)
```

Bases: [Notification](#)

A notification that may have a link.

Parameters

- **parent** (*QWidget*) – Parent widget
- **txt** (*str*) – Text to display in notification
- **anim_duration** (*int*) – Duration of the animation in msec
- **life_span** (*int*) – How long does the notification stays in place in msec
- **word_wrap** (*bool*) –
- **corner** (*Qt.Corner*) –

```
class spinetoolbox.widgets.notification.ChangeNotifier(parent, undo_stack, settings, settings_key,
                                                         corner=Qt.BottomRightCorner)
```

Bases: [PySide6.QtCore.QObject](#)

Parameters

- **parent** (*QWidget*) –
- **undo_stack** (*QUndoStack*) –
- **settings** (*QSettings*) –
- **settings_key** (*str*) –
- **corner** (*int*) –

_ANIMATION_LIFE_SPAN = 5000

_push_notification(*index*)

tear_down()

Tears down the notifier.

spinetoolbox.widgets.open_project_widget

Contains a class for a widget that represents a ‘Open Project Directory’ dialog.

Module Contents

Classes

<i>OpenProjectDialog</i>	A dialog that lets user select a project to open either by choosing
<i>CustomQFileSystemModel</i>	Custom file system model.
<i>DirValidator</i>	

class spinetoolbox.widgets.open_project_widget.**OpenProjectDialog**(*toolbox*)

Bases: PySide6.QtWidgets.QDialog

A dialog that lets user select a project to open either by choosing an old .proj file or by choosing a project directory.

Parameters

toolbox (*ToolboxUI*) – QMainWindow instance

set_keyboard_shortcuts()

Creates keyboard shortcuts for the ‘Root’, ‘Home’, etc. buttons.

connect_signals()

Connects signals to slots.

expand_and_resize(*p*)

Expands, resizes, and scrolls the tree view to the current directory when the file model has finished loading the path. Slot for the file model’s directoryLoaded signal. The directoryLoaded signal is emitted only if the directory has not been cached already. Note, that this is only used when the open project dialog is opened

Parameters

p (*str*) – Directory that has been loaded

validator_state_changed()

Changes the combobox border color according to the current state of the validator.

current_index_changed(*i*)

Combobox selection changed. This slot is processed when a new item is selected from the drop-down list. This is not processed when new item txt is `QValidator.Intermediate`.

Parameters

i (*int*) – Selected row in combobox

current_changed(*current*, *previous*)

Processed when the current item in file system tree view has been changed with keyboard or mouse. Updates the text in combobox.

Parameters

- **current** (*QModelIndex*) – Currently selected index
- **previous** (*QModelIndex*) – Previously selected index

set_selected_path(*index*)

Sets the text in the combobox as the selected path in the file system tree view.

Parameters

index (*QModelIndex*) – The index which was mouse clicked.

combobox_text_edited(*text*)

Updates selected path when combobox text is edited. Note: pressing enter in combobox does not trigger this.

selection()

Returns the selected path from dialog.

go_root(*checked=False*)

Slot for the ‘Root’ button. Scrolls the treeview to show and select the user’s root directory.

Note: We need to expand and scroll the tree view here after `setCurrentIndex` just in case the directory has been loaded already.

go_home(*checked=False*)

Slot for the ‘Home’ button. Scrolls the treeview to show and select the user’s home directory.

go_documents(*checked=False*)

Slot for the ‘Documents’ button. Scrolls the treeview to show and select the user’s documents directory.

go_desktop(*checked=False*)

Slot for the ‘Desktop’ button. Scrolls the treeview to show and select the user’s desktop directory.

open_project(*index*)

Opens project if index contains a valid Spine Toolbox project. Slot for the mouse `doubleClicked` signal. Prevents showing the ‘Not a valid spine toolbox project’ notification if user just wants to collapse a directory.

Parameters

index (*QModelIndex*) – File model index which was double clicked

done(*r*)

Checks that selected path exists and is a valid Spine Toolbox directory when ok button is clicked or when enter is pressed without the combobox being in focus.

Parameters

r (*int*) –

static `update_recents(entry, qsettings)`

Adds a new entry to QSettings variable that remembers the five most recent project storages.

Parameters

- **entry** (*str*) – Abs. path to a directory that most likely contains other Spine Toolbox Projects as well. First entry is also used as the initial path for File->New Project dialog.
- **qsettings** (*QSettings*) – Toolbox qsettings object

static `remove_directory_from_recents(p, qsettings)`

Removes directory from the recent project storages.

Parameters

- **p** (*str*) – Full path to a project directory
- **qsettings** (*QSettings*) – Toolbox qsettings object

show_context_menu(pos)

Shows the context menu for the QCombobox with a ‘Clear history’ entry.

Parameters

- **pos** (*QPoint*) – Mouse position

closeEvent(event=None)

Handles dialog closing.

Parameters

- **event** (*QCloseEvent*) – Close event

class `spinetoolbox.widgets.open_project_widget.CustomQFileSystemModel`

Bases: PySide6.QtWidgets.QFileSystemModel

Custom file system model.

columnCount(*parent=QModelIndex()*)

Returns one.

class `spinetoolbox.widgets.open_project_widget.DirValidator(parent=None)`

Bases: PySide6.QtGui.QValidator

validate(*txt, pos*)

Returns Invalid if input is invalid according to this validator’s rules, Intermediate if it is likely that a little more editing will make the input acceptable and Acceptable if the input is valid.

Parameters

- **txt** (*str*) – Text to validate
- **pos** (*int*) – Cursor position

Returns

Invalid, Intermediate, or Acceptable

Return type

QValidator.State

`spinetoolbox.widgets.parameter_value_editor`

An editor dialog for editing database (relationship) parameter values.

Module Contents

Classes

<i>ParameterValueEditor</i>	Dialog for editing parameter values in Database editor.
---	---

class `spinetoolbox.widgets.parameter_value_editor.ParameterValueEditor`(*index*, *parent=None*, *plain=False*)

Bases: *`spinetoolbox.widgets.parameter_value_editor_base.ParameterValueEditorBase`*

Dialog for editing parameter values in Database editor.

Parameters

- **index** (*QModelIndex*) – an index to a `parameter_value` in `parent_model`
- **parent** (*QWidget*, *optional*) – a parent widget
- **plain** (*bool*) – if `True`, allow only plain value editing, otherwise allow all parameter types

_set_data(*value*)

See base class.

`spinetoolbox.widgets.parameter_value_editor_base`

A base for editor windows for editing parameter values.

Module Contents

Classes

<i>ValueType</i>	Enum to identify value types that use different editors.
<i>ParameterValueEditorBase</i>	Dialog for editing parameter values.

Attributes

<i>_SELECTORS</i>

class `spinetoolbox.widgets.parameter_value_editor_base.ValueType`

Bases: `enum.Enum`

Enum to identify value types that use different editors.

PLAIN_VALUE**MAP****TIME_SERIES_FIXED_RESOLUTION****TIME_SERIES_VARIABLE_RESOLUTION****TIME_PATTERN****ARRAY****DATETIME****DURATION**`spinetoolbox.widgets.parameter_value_editor_base._SELECTORS`

```
class spinetoolbox.widgets.parameter_value_editor_base.ParameterValueEditorBase(index, editor_widgets, parent=None)
```

Bases: `PySide6.QtWidgets.QWidget`

Dialog for editing parameter values.

The dialog takes an index and shows a specialized editor corresponding to the value type in a stack widget. The user can change the value type by changing the specialized editor using a combo box. When the dialog is closed the value from the currently shown specialized editor is written back to the given index.

Parameters

- **index** (*QModelIndex*) – an index to a parameter_value in parent_model
- **editor_widgets** (*dict*) – a mapping from *ValueType* to *QWidget*
- **parent** (*QWidget*, *optional*) – a parent widget

accept()

Saves the parameter_value shown in the currently selected editor widget to the database manager.

_change_parameter_type(selector_index)

Handles switching between value types.

Does a rude conversion between fixed and variable resolution time series. In other cases, a default ‘empty’ value is used.

Parameters

selector_index (*int*) – an index to the selector combo box

_select_editor(value)

Shows the editor widget corresponding to the given value type on the editor stack.

_use_default_editor(message=None)

Opens the default editor widget. Optionally, displays a warning dialog indicating the problem.

Parameters

message (*str*, *optional*) –

_use_editor(*value*, *value_type*)

Sets a value to edit on an editor widget.

Parameters

- **value** (*object*) – value to edit
- **value_type** (*ValueType*) – type of value

abstract _set_data(*value*)

Writes parameter value back to the model.

Parameters

value (*object*) – value to write

Returns

True if the operation was successful, False otherwise

Return type

bool

`spinetoolbox.widgets.persistent_console_widget`

Module Contents

Classes

_CustomLineEdit

PersistentConsoleWidget

A widget to interact with a persistent process.

AnsiEscapeCodeHandler

Functions

_ansi_color(code[, bright])

class `spinetoolbox.widgets.persistent_console_widget._CustomLineEdit`(*console*)

Bases: `PySide6.QtWidgets.QPlainTextEdit`

property `min_pos`

property `new_line_indent`

reset(*current_prompt*)

new_line()

formatted_text()

raw_text()

set_raw_text(*text*)

_handle_text_changed()

Add indent to new lines.

_handle_cursor_position_changed()

Move cursor away from indent areas.

keyPressEvent(*ev*)

class spinetoolbox.widgets.persistent_console_widget.**PersistentConsoleWidget**(*toolbox, key, language, owner=None*)

Bases: PySide6.QtWidgets.QPlainTextEdit

A widget to interact with a persistent process.

Parameters

- **toolbox** (*ToolboxUI*) –
- **key** (*tuple*) – persistent process identifier
- **language** (*str*) – for syntax highlighting and prompting, etc.
- **owner** (*ProjectItemBase, optional*) – console owner

property prompt

property owner_names

property _input_start_pos

_command_checked

_msg_available

_command_finished

_history_item_available

_completions_available

_restarted

_killed

_flush_needed

_FLUSH_INTERVAL = 200

_MAX_LINES_PER_SECOND = 2000

_MAX_LINES_PER_CYCLE

_MAX_LINES_COUNT = 2000

closeEvent(*ev*)

name()

Returns console name for display purposes.

focusInEvent(*ev*)

mouseMoveEvent(*ev*)

mousePressEvent(*ev*)

mouseReleaseEvent(*ev*)

scrollContentsBy(*dx*, *dy*)

_handle_contents_changed()

_handle_selection_changed()

_handle_cursor_position_changed()

_handle_update_request(*_rect*, *_dy*)

Move line edit to input start pos.

resizeEvent(*ev*)

_move_and_resize_line_edit()

_update_user_input()

_start_flush_timer()

_flush_text_buffer()

Inserts all text from buffer.

_make_prompt()

_make_prompt_block(*prompt=""*)

_insert_prompt(*prompt=""*)

_insert_stdin_text(*cursor*, *text*)

Inserts highlighted text.

Parameters

- **cursor** (*QTextCursor*) –
- **text** (*str*) –

_do_insert_stdin_text(*cursor*, *text*)

_insert_stdout_text(*cursor*, *text*)

Inserts ansi highlighted text.

Parameters

- **cursor** (*QTextCursor*) –
- **text** (*str*) –

_insert_text_before_prompt(*text*, *with_prompt=False*)

Inserts given text before the prompt. Used when adding input and output from external execution.

Parameters

- **text** (*str*) –

_insert_text(*cursor*, *text*, *with_prompt*)

set_killed(*killed*)

Emits the killed signal.

Parameters

killed (*bool*) – if True, may the console rest in peace

_do_set_killed(*killed*)

Sets the console as killed or alive.

Parameters

killed (*bool*) – if True, may the console rest in peace

add_stdin(*data*)

Adds new prompt with data. Used when adding stdin from external execution.

Parameters

data (*str*) –

add_stdout(*data*)

Adds new line to stdout. Used when adding stdout from external execution.

Parameters

data (*str*) –

add_stderr(*data*)

Adds new line to stderr. Used when adding stderr from external execution.

Parameters

data (*str*) –

_get_current_text()

_get_prefix()

_highlight_current_input()

key_press_event(*ev*)

Handles key press event from line edit.

Returns

True if handled, False if not.

create_engine_manager()

Returns a new local or remote spine engine manager or an existing remote spine engine manager. Returns None if connecting to Spine Engine Server fails.

_issue_command(*text*)

Issues command.

Parameters

text (*str*) –

_do_check_command(*text*)

_handle_command_checked(*text*, *complete*)

Issues command.

Parameters

text (*str*) –

`_do_issue_command(text)`

`_handle_msg_available(msg_type, text)`

`_handle_command_finished()`

`_move_history(text, backwards)`

Moves history.

`_do_move_history(text, backwards)`

`_display_history_item(history_item, prefix)`

`_autocomplete(text)`

Autocompletes current text in the prompt (or output options if multiple matches).

Parameters

text (*str*) –

`_do_autocomplete(text)`

`_display_completions(text, prefix, completions)`

`_restart_persistent(_=False)`

Restarts underlying persistent process.

`_do_restart_persistent()`

`_handle_restarted()`

`_interrupt_persistent(_=False)`

Sends a task to executor which will interrupt the underlying persistent process.

`_do_interrupt_persistent()`

Interrupts the underlying persistent process.

`_kill_persistent(_=False)`

Sends a task to executor which will kill the underlying persistent process.

`_do_kill_persistent()`

Kills underlying persistent process.

`_extend_menu(menu)`

Appends two more actions: Restart, and Interrupt.

Parameters

menu (*QMenu*) – where to append

`contextMenuEvent(ev)`

Reimplemented to extend menu with custom actions.

`class spinetoolbox.widgets.persistent_console_widget.AnsiEscapeCodeHandler`(*fg_color*,
bg_color)

`_make_default_format()`

`endFormatScope()`

`setFormatScope(char_format)`

`parse_text(text)`

`spinetoolbox.widgets.persistent_console_widget._ansi_color(code, bright=False)`

`spinetoolbox.widgets.plain_parameter_value_editor`

An editor widget for editing plain number database (relationship) parameter values.

Module Contents

Classes

<i><code>PlainParameterValueEditor</code></i>	A widget to edit float or boolean type parameter values.
---	--

class `spinetoolbox.widgets.plain_parameter_value_editor.PlainParameterValueEditor`(*parent_widget=None*)

Bases: `PySide6.QtWidgets.QWidget`

A widget to edit float or boolean type parameter values.

Parameters

parent_widget (*QWidget*) – a parent widget

_set_number_or_string_enabled(*on*)

_set_string_enabled(*on*)

set_value(*value*)

Sets the value to be edited in this widget.

value()

Returns the value currently being edited.

`spinetoolbox.widgets.plot_canvas`

A Qt widget to use as a matplotlib backend.

Module Contents

Classes

<i><code>LegendPosition</code></i>	Generic enumeration.
<i><code>PlotCanvas</code></i>	A widget for plotting with matplotlib.

class `spinetoolbox.widgets.plot_canvas.LegendPosition`

Bases: `enum.Enum`

Generic enumeration.

Derive from this class to define new enumerations.

BOTTOM

RIGHT

```
class spinetoolbox.widgets.plot_canvas.PlotCanvas(parent=None,  
                                                  legend_axes_position=LegendPosition.BOTTOM)
```

Bases: matplotlib.backends.backend_qt5agg.FigureCanvasQTAgg

A widget for plotting with matplotlib.

Parameters

- **legend_axes_position** ([LegendPosition](#)) – legend axes position relative to plot axes
- **parent** (*QWidget*, *optional*) – a parent widget

property axes

figure's axes

Type

matplotlib.axes.Axes

property legend_axes

figure's legend axes

Type

matplotlib.axes.Axes

has_twinned_axes()

Checks whether the axes have been twinned.

Returns

True if axes have been twinned, False otherwise

Return type

bool

twinned_axes()

Returns twinned axes.

Returns

twinned axes

Return type

list of Axes

spinetoolbox.widgets.plot_widget

A Qt widget showing a toolbar and a matplotlib plotting canvas.

Module Contents

Classes

<i>PlotWidget</i>	A widget that contains a toolbar and a plotting canvas.
<i>_PlotDataView</i>	Custom QTableView class with copy and paste methods.
<i>_PlotDataWidget</i>	

Functions

<i>prepare_plot_in_window_menu</i> (menu)	Fills a given menu with available plot window names.
---	--

class `spinetoolbox.widgets.plot_widget.PlotWidget`(parent=None,
legend_axes_position=LegendPosition.BOTTOM)

Bases: PySide6.QtWidgets.QWidget

A widget that contains a toolbar and a plotting canvas.

canvas

the plotting canvas

Type

[*PlotCanvas*](#)

original_xy_data

unmodified data on which the plots are based

Type

list of XYData

Parameters

- **parent** (*QWidget*, *optional*) – parent widget
- **legend_axes_position** ([*LegendPosition*](#)) – legend axes position relative to plot axes

plot_windows

A global list of plot windows.

closeEvent(*event*)

Removes the window from plot_windows and closes.

contextMenuEvent(*event*)

Shows plot context menu.

_get_plot_data()

Gathers plot data into a table.

Returns

data as table

Return type

list of list

copy_plot_data()

Copies plot data to clipboard.

show_plot_data()

Opens a separate window that shows the plot data.

add_legend(handles)

Adds a legend to the plot's legend axes.

Parameters

handles (*list*) – legend handles

use_as_window(parent_window, document_name)

Prepares the widget to be used as a window and adds it to plot_windows list.

Parameters

- **parent_window** (*QWidget*) – a parent window
- **document_name** (*str*) – a string to add to the window title

static _unique_window_name(document_name)

Returns an unique identifier for a new plot window.

class spinetoolbox.widgets.plot_widget._PlotDataView(*parent=None*)

Bases: *spinetoolbox.widgets.custom_qtableview.CopyPasteTableView*

Custom QTableView class with copy and paste methods.

contextMenuEvent(event)

class spinetoolbox.widgets.plot_widget._PlotDataWidget(*rows, parent=None*)

Bases: *PySide6.QtWidgets.QWidget*

set_size_according_to_parent()

Sets the size of the widget according to the parent widget's dimensions and the data in the table

spinetoolbox.widgets.plot_widget.prepare_plot_in_window_menu(menu)

Fills a given menu with available plot window names.

Parameters

menu (*QMenu*) – menu to modify

spinetoolbox.widgets.plugin_manager_widgets

Contains PluginManager dialogs and widgets.

Module Contents

Classes

_InstallPluginModel

_ManagePluginsModel

<i>InstallPluginDialog</i>	Initialize class
----------------------------	------------------

<i>ManagePluginsDialog</i>	Initialize class
----------------------------	------------------

```
class spinetoolbox.widgets.plugin_manager_widgets._InstallPluginModel
    Bases: PySide6.QtGui.QStandardItemModel
    data(index, role=None)

class spinetoolbox.widgets.plugin_manager_widgets._ManagePluginsModel
    Bases: _InstallPluginModel
    flags(index)

class spinetoolbox.widgets.plugin_manager_widgets.InstallPluginDialog(parent)
    Bases: PySide6.QtWidgets.QDialog
    Initialize class
    item_selected
    populate_list(names)
    _handle_search_text_changed(_text)
    _filter_model()
    _handle_ok_clicked(_=False)
    _emit_item_selected(index)
    _update_ok_button_enabled(_selected, _deselected)

class spinetoolbox.widgets.plugin_manager_widgets.ManagePluginsDialog(parent)
    Bases: PySide6.QtWidgets.QDialog
    Initialize class
    item_removed
    item_updated
    populate_list(names)
    _create_plugin_widget(plugin_name, can_update)
    _emit_item_removed(plugin_name)
    _emit_item_updated(plugin_name)
```

spinetoolbox.widgets.project_item_drag

Classes for custom QListView.

Module Contents

Classes

<i>ProjectItemDragMixin</i>	Custom class with dragging support.
<i>NiceButton</i>	
<i>ProjectItemButtonBase</i>	Custom class with dragging support.
<i>ProjectItemButton</i>	Custom class with dragging support.
<i>ProjectItemSpecButton</i>	Custom class with dragging support.
<i>ShadeMixin</i>	
<i>ShadeProjectItemSpecButton</i>	Custom class with dragging support.
<i>ShadeButton</i>	
<i>_ChoppedIcon</i>	
<i>_ChoppedIconEngine</i>	
<i>ProjectItemSpecArray</i>	An array of <i>ProjectItemSpecButton</i> that can be expanded/collapsed.

```
class spinetoolbox.widgets.project_item_drag.ProjectItemDragMixin(*args, **kwargs)
    Custom class with dragging support.
    drag_about_to_start
    mouseMoveEvent(event)
        Start dragging action if needed
    mouseReleaseEvent(event)
        Forget drag start position
class spinetoolbox.widgets.project_item_drag.NiceButton(*args, **kwargs)
    Bases: PySide6.QtWidgets.QToolButton
    setText(text)
    set_orientation(orientation)
class spinetoolbox.widgets.project_item_drag.ProjectItemButtonBase(toolbox, item_type, icon,
                                                                    parent=None)
    Bases: ProjectItemDragMixin, NiceButton
    Custom class with dragging support.
    set_colored_icons(colored)
    _handle_drag_about_to_start()
    mousePressEvent(event)
        Register drag start position
    abstract _make_mime_data_text()
```

```

class spinetoolbox.widgets.project_item_drag.ProjectItemButton(toolbox, item_type, icon,
                                                                parent=None)

    Bases: ProjectItemButtonBase
    Custom class with dragging support.
    double_clicked
    _make_mime_data_text()
    mouseDoubleClickEvent(event)

class spinetoolbox.widgets.project_item_drag.ProjectItemSpecButton(toolbox, item_type, icon,
                                                                    spec_name="",
                                                                    parent=None)

    Bases: ProjectItemButtonBase
    Custom class with dragging support.
    property spec_name
    _make_mime_data_text()
    contextMenuEvent(event)
    mouseDoubleClickEvent(event)

class spinetoolbox.widgets.project_item_drag.ShadeMixin

    paintEvent(ev)

class spinetoolbox.widgets.project_item_drag.ShadeProjectItemSpecButton(toolbox, item_type,
                                                                           icon, spec_name="",
                                                                           parent=None)

    Bases: ShadeMixin, ProjectItemSpecButton
    Custom class with dragging support.
    clone()

class spinetoolbox.widgets.project_item_drag.ShadeButton(*args, **kwargs)
    Bases: ShadeMixin, NiceButton

class spinetoolbox.widgets.project_item_drag._ChoppedIcon(icon, size)
    Bases: PySide6.QtGui.QIcon
    update()

class spinetoolbox.widgets.project_item_drag._ChoppedIconEngine(icon, size)
    Bases: PySide6.QtGui.QIconEngine
    update()
    pixmap(size, mode, state)

class spinetoolbox.widgets.project_item_drag.ProjectItemSpecArray(toolbox, model, item_type,
                                                                    icon)

    Bases: PySide6.QtWidgets.QToolBar
    An array of ProjectItemSpecButton that can be expanded/collapsed.

```

Parameters

- **toolbox** (`ToolboxUI`) –
- **model** (`FilteredSpecificationModel`) –
- **item_type** (`str`) –
- **icon** (`ColoredIcon`) –

set_colored_icons(*colored*)

update()

_update_button_visible_icon_color()

set_color(*color*)

paintEvent(*ev*)

_get_first_chopped_index()

Returns the index of the first chopped action (chopped = not drawn because of space).

Returns

list(QAction) int or NoneType

_add_filling(*actions*, *ind*)

Adds a button to fill empty space after the last visible action.

Parameters

- **actions** (`list(QAction)`) – actions
- **ind** (`int` or `NoneType`) – index of the first chopped one or None if all are visible

_get_filling(*previous*)

Returns the position and size of the filling widget.

Parameters

previous (`QWidget`) – last visible widget

Returns

position x int: position y int: width int: height

Return type

int

_populate_extension_menu(*actions*, *ind*)

Populates extension menu with chopped actions.

Parameters

- **actions** (`list(QAction)`) – actions
- **ind** (`int` or `NoneType`) – index of the first chopped one or None if all are visible

showEvent(*ev*)

_update_button_geom(*orientation=None*)

Updates geometry of buttons given the orientation

Parameters

orientation (`Qt.Orientation`) –

```
_show_spec_form(_checked=False)
toggle_visibility(_checked=False)
set_visible(visible)
_insert_specs(parent, first, last)
_remove_specs(parent, first, last)
_remove_spec(row)
_reset_specs()
_add_spec(row)
```

`spinetoolbox.widgets.properties_widget`

Contains PropertiesWidgetBase.

Module Contents

Classes

PropertiesWidgetBase

Properties widget base class.

class `spinetoolbox.widgets.properties_widget.PropertiesWidgetBase(toolbox, base_color=None)`

Bases: `PySide6.QtWidgets.QWidget`

Properties widget base class.

property `fg_color`

set_item(*project_item*)

Sets the active project item.

Parameters

project_item (`ProjectItem`) – active project item

unset_item()

Unsets the active project item.

set_color_and_icon(*base_color, icon=None*)

eventFilter(*obj, ev*)

paintEvent(*ev*)

Paints background

spinetoolbox.widgets.report_plotting_failure

Functions to report failures in plotting to the user.

Module Contents

Functions

<i>report_plotting_failure</i> (error, parent_widget)	Reports a PlottingError exception to the user.
---	--

spinetoolbox.widgets.report_plotting_failure.**report_plotting_failure**(*error, parent_widget*)
 Reports a PlottingError exception to the user.

Parameters

- **error** ([*PlottingError*](#)) – exception to report
- **parent_widget** ([*QWidget*](#)) – parent widget

spinetoolbox.widgets.select_database_items

A widget and utilities to select database items.

Module Contents

Classes

<i>SelectDatabaseItems</i>	Widget that allows selecting database items.
--	--

Functions

<i>add_check_boxes</i> (check_boxes, checked_states, ...)	Adds check boxes to grid layout.
<i>batch_set_check_state</i> (boxes, checked)	Sets the checked state of multiple check boxes.

spinetoolbox.widgets.select_database_items.**add_check_boxes**(*check_boxes, checked_states, select_all_button, deselect_all_button, state_changed_slot, layout*)

Adds check boxes to grid layout.

Parameters

- **check_boxes** (*dict*) – mapping from label to QCheckBox
- **checked_states** (*dict*) – mapping from label to checked state boolean
- **select_all_button** ([*QPushButton*](#)) – the Select all button
- **deselect_all_button** ([*QPushButton*](#)) – the Deselect all button
- **state_changed_slot** (*Callable*) – slot to call when any checked state changes

- **layout** (*QGridLayout*) – target layout

`spinetoolbox.widgets.select_database_items.batch_set_check_state(boxes, checked)`

Sets the checked state of multiple check boxes.

Parameters

- **boxes** (*Iterable of QCheckBox*) – check boxes
- **checked** (*bool*) – checked state

class `spinetoolbox.widgets.select_database_items.SelectDatabaseItems(checked_states=None, parent=None)`

Bases: `PySide6.QtWidgets.QWidget`

Widget that allows selecting database items.

Parameters

- **checked_states** (*dict, optional*) – mapping from item name to check state boolean
- **parent** (*QWidget*) – parent widget

checked_state_changed

COLUMN_COUNT = 3

_DATA_ITEMS = ('object', 'relationship', 'entity_group', 'parameter_value', 'entity_metadata', ...)

_SCENARIO_ITEMS = ('alternative', 'scenario', 'scenario_alternative')

checked_states()

Collects the checked states of database items.

Returns

mapping from item name to checked state boolean

Return type

dict

any_checked()

Checks if any of the checkboxes is checked.

Returns

True if any check box is checked, False otherwise

Return type

bool

any_structural_item_checked()

_select_data_items(*_=False*)

Checks all data items.

_select_scenario_items(*_=False*)

Checks all scenario items.

spinetoolbox.widgets.set_description_dialog

A widget for editing project description

Module Contents

Classes

<i>SetDescriptionDialog</i>	Dialog for setting a description for a project.
-----------------------------	---

class spinetoolbox.widgets.set_description_dialog.**SetDescriptionDialog**(*toolbox*, *project*)

Bases: PySide6.QtWidgets.QDialog

Dialog for setting a description for a project.

Parameters

- **toolbox** (*ToolboxUI*) – QMainWindow instance
- **project** (*SpineToolboxProject*) –

property description

_set_ok_enabled()

accept()

spinetoolbox.widgets.settings_widget

Widget for controlling user settings.

Module Contents

Classes

<i>SettingsWidgetBase</i>	param qsettings Toolbox settings
<i>SpineDBEditorSettingsMixin</i>	
<i>SpineDBEditorSettingsWidget</i>	A widget to change user's preferred settings, but only for the Spine db editor.
<i>SettingsWidget</i>	A widget to change user's preferred settings.

Functions

<code>_get_kernel_name_by_exe(p, kernel_model)</code>	Returns the kernel name corresponding to given executable or an empty string if not found.
<code>_selected_project_matches_kernel_project(...)</code>	Checks if given Julia kernel's project matches the given Julia project.
<code>_samefile(a, b)</code>	

class `spinetoolbox.widgets.settings_widget.SettingsWidgetBase(qsettings)`

Bases: `PySide6.QtWidgets.QWidget`

Parameters

qsettings (`QSettings`) – Toolbox settings

property `qsettings`

connect_signals()

Connect signals.

keyPressEvent(*e*)

Close settings form when escape key is pressed.

Parameters

e (`QKeyEvent`) – Received key press event.

mousePressEvent(*e*)

Save mouse position at the start of dragging.

Parameters

e (`QMouseEvent`) – Mouse event

mouseReleaseEvent(*e*)

Save mouse position at the end of dragging.

Parameters

e (`QMouseEvent`) – Mouse event

mouseMoveEvent(*e*)

Moves the window when mouse button is pressed and mouse cursor is moved.

Parameters

e (`QMouseEvent`) – Mouse event

update_ui()

Updates UI to reflect current settings. Called when the user choses to cancel their changes. Undoes all temporary UI changes that resulted from the user playing with certain settings.

save_settings()

Gets selections and saves them to persistent memory.

update_ui_and_close()

Updates UI to reflect current settings and close.

save_and_close()

Saves settings and close.

class spinetoolbox.widgets.settings_widget.SpineDBEditorSettingsMixin

connect_signals()

Connect signals.

read_settings()

Read saved settings from app QSettings instance and update UI to display them.

save_settings()

Get selections and save them to persistent memory.

update_ui()

set_auto_expand_objects(*checked=False*)

set_merge_dbs(*checked=False*)

class spinetoolbox.widgets.settings_widget.SpineDBEditorSettingsWidget(*multi_db_editor*)

Bases: [SpineDBEditorSettingsMixin](#), [SettingsWidgetBase](#)

A widget to change user's preferred settings, but only for the Spine db editor.

Initialize class.

property db_mgr

show()

class spinetoolbox.widgets.settings_widget.SettingsWidget(*toolbox*)

Bases: [SpineDBEditorSettingsMixin](#), [SettingsWidgetBase](#)

A widget to change user's preferred settings.

Parameters

toolbox ([ToolboxUI](#)) – Parent widget.

property db_mgr

connect_signals()

Connect signals.

_make_python_kernel_context_menu()

Returns a context-menu for Python kernel comboBox.

_make_julia_kernel_context_menu()

Returns a context-menu for Julia kernel comboBox.

eventFilter(*o, event*)

Event filter that catches mouse right button release events. This event typically closes the context-menu, but we want to prevent this and show a context-menu instead.

Parameters

- **o** (*QObject*) – Watcher
- **event** (*QEvent*) – Event

Returns

True when event is caught, False otherwise

Return type

bool

`_update_python_widgets_enabled(state)`

Enables or disables some widgets based on given boolean state.

`_update_julia_widgets_enabled(state)`

Enables or disables some widgets based on given boolean state.

`_update_remote_execution_page_widget_status(state)`

Enables or disables widgets on Remote Execution page, based on the state of remote execution enabled check box.

`_show_install_julia_wizard(_=False)`

Opens Install Julia Wizard.

`_show_add_up_spine_opt_wizard(_=False)`

Opens the add/update SpineOpt wizard.

`browse_gams_button_clicked(_=False)`

Calls static method that shows a file browser for selecting a Gams executable.

`browse_julia_button_clicked(_=False)`

Calls static method that shows a file browser for selecting a Julia path.

`browse_julia_project_button_clicked(_=False)`

Calls static method that shows a folder browser for selecting a Julia project.

`browse_python_button_clicked(_=False)`

Calls static method that shows a file browser for selecting a Python interpreter.

`browse_conda_button_clicked(_=False)`

Calls static method that shows a file browser for selecting a Conda executable.

`browse_certificate_directory_clicked(_=False)`

Calls static method that shows a file browser for selecting the security folder for Engine Server.

`make_python_kernel(_=False)`

Makes a Python kernel for Jupyter Console based on selected Python interpreter. If a kernel using this Python interpreter already exists, sets that kernel selected in the comboBox.

`make_julia_kernel(_=False)`

Makes a Julia kernel for Jupyter Console based on selected Julia executable and Julia project. If a kernel using the selected Julia executable and project already exists, sets that kernel selected in the comboBox.

`show_python_kernel_context_menu_on_combobox(pos)`

Shows the context-menu on Python kernels combobox.

`show_julia_kernel_context_menu_on_combobox(pos)`

Shows the context-menu on Julia kernels combobox.

`show_python_kernel_context_menu_on_combobox_list(pos)`

Shows the context-menu on Python kernels combobox popup list.

`show_julia_kernel_context_menu_on_combobox_list(pos)`

Shows the context-menu on Julia kernels combobox popup list.

`_open_python_kernel_resource_dir(_=False)`

Opens Python kernels resource dir.

`_open_julia_kernel_resource_dir(_=False)`

Opens Julia kernels resource dir.

`open_rsc_dir(item)`

Open path hidden in given item's tooltip in file browser.

`browse_work_path(_=False)`

Open file browser where user can select the path to wanted work directory.

`show_color_dialog(_=False)`

Lets user pick the background color from a color dialog.

`update_bg_color()`

Set tool button icon as the selected color and update Design View scene background color.

`update_scene_bg(_=False)`

Draw background on scene depending on radiobutton states.

`update_links_geometry(checked=False)`

`update_items_path(checked=False)`

`set_toolbar_colored_icons(checked=False)`

`_update_properties_widget(_checked=False)`

`read_settings()`

Read saved settings from app QSettings instance and update UI to display them.

`_read_engine_settings()`

Reads Engine settings and sets the corresponding UI elements.

`save_settings()`

Get selections and save them to persistent memory. Note: On Linux, True and False are saved as boolean values into QSettings. On Windows, booleans and integers are saved as strings. To make it consistent, we should use strings.

`_save_engine_settings()`

Stores Engine settings to application settings.

Returns

True if settings were stored successfully, False otherwise

Return type

bool

`_get_julia_settings()`

`set_work_directory(new_work_dir)`

Sets new work directory.

Parameters

`new_work_dir` (*str*) – Possibly a new work directory

`update_ui()`

Updates UI to reflect current settings. Called when the user choses to cancel their changes. Undoes all temporary UI changes that resulted from the user playing with certain settings.

_edit_remote_host(*new_text*)

Prepends host line edit with the protocol for user convenience.

Parameters

new_text (*str*) – Text in the line edit after user has entered a character

start_fetching_julia_kernels()

Starts a thread for fetching Julia kernels.

stop_fetching_julia_kernels()

Terminates the kernel fetcher thread.

add_julia_kernel(*kernel_name, resource_dir, conda, icon, deats*)

Adds a kernel entry as an item to Julia kernels comboBox.

restore_saved_julia_kernel()

Sets saved or given julia kernel selected after kernels have been loaded.

start_fetching_python_kernels()

Starts a thread for fetching Python kernels.

stop_fetching_python_kernels()

Terminates the kernel fetcher thread.

add_python_kernel(*kernel_name, resource_dir, conda, icon, deats*)

Adds a kernel entry as an item to Python kernels comboBox.

restore_saved_python_kernel()

Sets saved or given python kernel selected after kernels have been loaded.

closeEvent(*ev*)

`spinetoolbox.widgets.settings_widget._get_kernel_name_by_exe(p, kernel_model)`

Returns the kernel name corresponding to given executable or an empty string if not found.

Parameters

- **p** (*str*) – Absolute path to an executable
- **kernel_model** (*QStandardItemModel*) – Model containing items, which contain kernel spec details as item data

Returns

Kernel name or an empty string

Return type

str

`spinetoolbox.widgets.settings_widget._selected_project_matches_kernel_project(julia_kernel_name, julia_project, kernel_model)`

Checks if given Julia kernel's project matches the given Julia project.

Parameters

- **julia_kernel_name** (*str*) – Kernel name
- **julia_project** (*str*) – Path or some other string (e.g. '@.') to denote the Julia project
- **kernel_model** (*QStandardItemModel*) – Model containing kernels

Returns

True if projects match, False otherwise

Return type

bool

`spinetoolbox.widgets.settings_widget._samefile(a, b)`

`spinetoolbox.widgets.statusbars`

Functions to make and handle QStatusBars.

Module Contents

Classes

<i>MainStatusBar</i>	A status bar for the main toolbox window.
----------------------	---

class `spinetoolbox.widgets.statusbars.MainStatusBar(toolbox)`

Bases: `PySide6.QtWidgets.QStatusBar`

A status bar for the main toolbox window.

Parameters

toolbox (`ToolboxUI`) –

`_ALL_RUNS = 'All executions'`

`_populate_executions_menu()`

`reset_executions_button_text()`

`_select_execution(action)`

`spinetoolbox.widgets.time_pattern_editor`

An editor widget for editing a time pattern type (relationship) parameter values.

Module Contents

Classes

<i>TimePatternEditor</i>	A widget for editing time patterns.
--------------------------	-------------------------------------

class `spinetoolbox.widgets.time_pattern_editor.TimePatternEditor(parent=None)`

Bases: `PySide6.QtWidgets.QWidget`

A widget for editing time patterns.

Parameters

parent (`QWidget`) – parent widget

`_show_table_context_menu(position)`

Opens the table's context menu.

Parameters

`position` (*QPoint*) – menu's position on the table

`set_value(value)`

Sets the parameter_value to be edited.

`value()`

Returns the parameter_value currently being edited.

`_open_header_editor(column)`

`spinetoolbox.widgets.time_series_fixed_resolution_editor`

Contains logic for the fixed step time series editor widget.

Module Contents

Classes

<i><code>TimeSeriesFixedResolutionEditor</code></i>	A widget for editing time series data with a fixed time step.
---	---

Functions

<i><code>_resolution_to_text(resolution)</code></i>	Converts a list of durations into a string of comma-separated durations.
<i><code>_text_to_resolution(text)</code></i>	Converts a comma-separated string of durations into a resolution array.

`spinetoolbox.widgets.time_series_fixed_resolution_editor._resolution_to_text(resolution)`

Converts a list of durations into a string of comma-separated durations.

`spinetoolbox.widgets.time_series_fixed_resolution_editor._text_to_resolution(text)`

Converts a comma-separated string of durations into a resolution array.

class `spinetoolbox.widgets.time_series_fixed_resolution_editor.TimeSeriesFixedResolutionEditor` (*parent=None*)

Bases: `PySide6.QtWidgets.QWidget`

A widget for editing time series data with a fixed time step.

Parameters

`parent` (*QWidget*) – a parent widget

`_resolution_changed()`

Updates the models after resolution change.

_show_table_context_menu(*position*)

Shows the table's context menu.

Parameters

position (*QPoint*) – menu's position in table view's coordinates

_select_date(*selected_date*)

set_value(*value*)

Sets the parameter_value for editing in this widget.

_show_calendar()

_start_time_changed()

Updates the model due to start time change.

_update_plot(*topLeft=None, bottomRight=None, roles=None*)

Updated the plot.

value()

Returns the parameter_value currently being edited.

_open_header_editor(*column*)

spinetoolbox.widgets.time_series_variable_resolution_editor

Contains logic for the variable resolution time series editor widget.

Module Contents

Classes

<i>TimeSeriesVariableResolutionEditor</i>	A widget for editing variable resolution time series data.
---	--

class spinetoolbox.widgets.time_series_variable_resolution_editor.**TimeSeriesVariableResolutionEditor**(*parent*)

Bases: PySide6.QtWidgets.QWidget

A widget for editing variable resolution time series data.

Parameters

parent (*QWidget*) – a parent widget

_show_table_context_menu(*position*)

Shows the table's context menu.

Parameters

position (*QPoint*) – menu's position on the table

set_value(*value*)

Sets the time series being edited.

_update_plot(*topLeft=None, bottomRight=None, roles=None*)

Updates the plot widget.

value()

Return the time series currently being edited.

_open_header_editor(*column*)

spinetoolbox.widgets.toolbars

Functions to make and handle QToolBars.

Module Contents

Classes

<i>ToolBar</i>	Base class for Toolbox toolbars.
<i>PluginToolBar</i>	A plugin toolbar.
<i>MainToolBar</i>	The main application toolbar: Items Execute
<i>PaddingLabel</i>	

class spinetoolbox.widgets.toolbars.**ToolBar**(*name*, *toolbox*)

Bases: PySide6.QtWidgets.QToolBar

Base class for Toolbox toolbars.

Parameters

- **name** (*str*) – toolbar’s name
- **toolbox** (*ToolboxUI*) – Toolbox main window

abstract **set_color**(*color*)

Sets toolbar’s background color.

Parameters

color (*QColor*) – background color

set_project_actions_enabled(*enabled*)

Enables or disables project related actions.

Parameters

enabled (*bool*) – True to enable actions, False to disable

class spinetoolbox.widgets.toolbars.**PluginToolBar**(*name*, *parent*)

Bases: *ToolBar*

A plugin toolbar.

Parameters

parent (*ToolboxUI*) – QMainWindow instance

setup(*plugin_specs*, *disabled_names*)

Sets up the toolbar.

Parameters

- **plugin_specs** (*dict*) – mapping from specification name to specification

- **disabled_names** (*Iterable of str*) – specifications that should be disabled

set_color(*color*)

Sets toolbar's background color.

Parameters

color (*QColor*) – background color

_update_spec_button_name(*old_name, new_name*)

class spinetoolbox.widgets.toolbars.**MainToolBar**(*execute_project_action, execute_selection_action, stop_execution_action, parent*)

Bases: [ToolBar](#)

The main application toolbar: Items | Execute

Parameters

- **execute_project_action** (*QAction*) – action to execute project
- **execute_selection_action** (*QAction*) – action to execute selected items
- **stop_execution_action** (*QAction*) – action to stop execution
- **parent** ([ToolboxUI](#)) – QMainWindow instance

_SEPARATOR = ';;'

set_project_actions_enabled(*enabled*)

Enables or disables project related actions.

Parameters

enabled (*bool*) – True to enable actions, False to disable

set_color(*color*)

Sets toolbar's background color.

Parameters

color (*QColor*) – background color

setup()

add_project_item_buttons()

_add_project_item_button(*item_type, factory, colored*)

set_colored_icons(*colored*)

_make_tool_button(*icon, text, slot, tip=None*)

Makes a new tool button and adds it to the toolbar.

Parameters

- **icon** (*QIcon*) – button's icon
- **text** (*str*) – button's text
- **slot** (*Callable*) – slot where to connect button's clicked signal
- **tip** (*str*) – button's tooltip

Returns

created button

Return type`QToolButton`**`_add_tool_button(button)`**

Adds a button to the toolbar.

Parameters**`button`** (`QToolButton`) – button to add**`add_execute_buttons()`**

Adds project execution buttons to the toolbar.

`dragLeaveEvent(event)`**`dragEnterEvent(event)`****`dragMoveEvent(event)`****`dropEvent(event)`****`_update_drop_actions(event)`**

Updates source and target actions for drop operation:

Parameters**`event`** (`QDragMoveEvent`) –**`paintEvent(ev)`**

Draw a line as drop indicator.

`_drop_line()`**`icon_ordering()`****`class` `spinetoolbox.widgets.toolbars.PaddingLabel(*args, **kwargs)`**Bases: `PySide6.QtWidgets.QLabel`

20.1.2 Submodules

`spinetoolbox.__main__`

Spine Toolbox application main file.

Module Contents

`spinetoolbox.__main__.return_code`

`spinetoolbox._version`

`spinetoolbox.config`

Application constants and style sheets

Module Contents

```
spinetoolbox.config.LATEST_PROJECT_VERSION = 11
spinetoolbox.config.REQUIRED_SPINE_OPT_VERSION = '0.6.9'
spinetoolbox.config.INVALID_CHARS = ['<', '>', ':', '"', '/', '\\', '|', '?', '*', '.']
spinetoolbox.config.INVALID_FILENAME_CHARS = ['<', '>', ':', '"', '/', '\\', '|', '?', '*']
spinetoolbox.config._frozen
spinetoolbox.config._path_to_executable
spinetoolbox.config.APPLICATION_PATH
spinetoolbox.config._program_root
spinetoolbox.config.DEFAULT_WORK_DIR
spinetoolbox.config.DOCUMENTATION_PATH
spinetoolbox.config.ONLINE_DOCUMENTATION_URL =
'https://spine-toolbox.readthedocs.io/en/master/'
spinetoolbox.config.PLUGINS_PATH
spinetoolbox.config.PLUGIN_REGISTRY_URL =
'https://spine-tools.github.io/PluginRegistry/registry.json'
spinetoolbox.config.JUPYTER_KERNEL_TIME_TO_DEAD = 20
spinetoolbox.config.PROJECT_FILENAME = 'project.json'
spinetoolbox.config.PROJECT_LOCAL_DATA_DIR_NAME = 'local'
spinetoolbox.config.PROJECT_LOCAL_DATA_FILENAME = 'project_local_data.json'
spinetoolbox.config.SPECIFICATION_LOCAL_DATA_FILENAME = 'specification_local_data.json'
spinetoolbox.config.PROJECT_ZIP_FILENAME = 'project_package'
spinetoolbox.config.STATUSBAR_SS = 'QStatusBar{background-color: #EBEBE0; border-width:
1px; border-color: gray; border-style: groove;}'
spinetoolbox.config.BG_COLOR = '#19232D'
spinetoolbox.config.FG_COLOR = '#F0F0F0'
spinetoolbox.config.SETTINGS_SS
```

spinetoolbox.config.TEXTBROWSER_SS

spinetoolbox.config.MAINWINDOW_SS

spinetoolbox.execution_managers

Classes to manage tool instance execution in various forms.

Module Contents

Classes

<i>ExecutionManager</i>	Base class for all tool instance execution managers.
<i>QProcessExecutionManager</i>	Class to manage tool instance execution using a PySide6 QProcess.

class spinetoolbox.execution_managers.**ExecutionManager**(*logger*)

Bases: PySide6.QtCore.QObject

Base class for all tool instance execution managers.

Class constructor.

Parameters

logger (*LoggerInterface*) – a logger instance

execution_finished

abstract start_execution(*workdir=None*)

Starts the execution.

Parameters

workdir (*str*) – Work directory

abstract stop_execution()

Stops the execution.

class spinetoolbox.execution_managers.**QProcessExecutionManager**(*logger, program="", args=None, silent=False, semisilent=False*)

Bases: *ExecutionManager*

Class to manage tool instance execution using a PySide6 QProcess.

Class constructor.

Parameters

- **logger** (*LoggerInterface*) – a logger instance
- **program** (*str*) – Path to program to run in the subprocess (e.g. julia.exe)
- **args** (*list, optional*) – List of argument for the program (e.g. path to script file)
- **silent** (*bool*) – Whether or not to emit logger msg signals
- **semisilent** (*bool*) – If True, show Process Log messages

program()

Program getter method.

args()

Program argument getter method.

start_execution(*workdir=None*)

Starts the execution of a command in a QProcess.

Parameters

workdir (*str*, *optional*) – Work directory

wait_for_process_finished(*msecs=30000*)

Wait for subprocess to finish.

Parameters

msecs (*int*) – Timeout in milliseconds

Returns

True if process finished successfully, False otherwise

process_started()

Run when subprocess has started.

on_state_changed(*new_state*)

Runs when QProcess state changes.

Parameters

new_state (*int*) – Process state number (QProcess::ProcessState)

on_process_error(*process_error*)

Runs if there is an error in the running QProcess.

Parameters

process_error (*int*) – Process error number (QProcess::ProcessError)

teardown_process()

Tears down the QProcess in case a QProcess.ProcessError occurred. Emits execution_finished signal.

stop_execution()

See base class.

on_process_finished(*exit_code*, *exit_status*)

Runs when subprocess has finished.

Parameters

- **exit_code** (*int*) – Return code from external program (only valid for normal exits)
- **exit_status** (*int*) – Crash or normal exit (QProcess::ExitStatus)

on_ready_stdout()

Emit data from stdout.

on_ready_stderr()

Emit data from stderr.

spinetoolbox.fetch_parent

The FetchParent and FlexibleFetchParent classes.

Module Contents

Classes

<i>FetchParent</i>	Attrs:
<i>ItemTypeFetchParent</i>	Attrs:
<i>FlexibleFetchParent</i>	Attrs:

class spinetoolbox.fetch_parent.**FetchParent**(owner=None, chunk_size=1000)

Bases: PySide6.QtCore.QObject

Attrs:

fetch_token (int or None) will_have_children (bool or None): Whether this parent will have children if fetched. None means we don't know yet. Set to a boolean value whenever we find out.

Parameters

- **owner** (*object*) – somebody who owns this FetchParent. If it's a QObject instance, then this FetchParent
- **destroyed** (*becomes obsolete whenever the owner is*) –
- **chunk_size** (*int or None*) – the number of items this parent should be happy with fetching at a time.
- **None** (*If*) –
- **DB.** (*then no limit is imposed and the parent should fetch the entire contents of the*) –

abstract property fetch_item_type

Returns the type of item to fetch, e.g., "object_class".

Returns

str

property is_obsolete

property is_fetched

property is_busy

_changes_pending

reset_fetching(fetch_token)

Resets fetch parent as if nothing was ever fetched.

Parameters

fetch_token (*object*) – current fetch token

position(db_map)

increment_position(*db_map*)

_apply_pending_changes()

add_item(*item*, *db_map*)

update_item(*item*, *db_map*)

remove_item(*item*, *db_map*)

accepts_item(*item*, *db_map*)

Called by the associated SpineDBWorker whenever items are fetched and also added/updated/removed. Returns whether this parent accepts that item as a children.

In case of modifications, the SpineDBWorker will call one or more of `handle_items_added()`, `handle_items_updated()`, or `handle_items_removed()` with all the items that pass this test.

Parameters

- **item** (*dict*) – The item
- **db_map** (*DiffDatabaseMapping*) –

Returns

bool

shows_item(*item*, *db_map*)

Called by the associated SpineDBWorker whenever items are fetched and accepted. Returns whether this parent will show this item to the user.

Parameters

- **item** (*dict*) – The item
- **db_map** (*DiffDatabaseMapping*) –

Returns

bool

will_have_children_change()

Called when the `will_have_children` property changes.

set_obsolete(*obsolete*)

Sets the obsolete status.

Parameters

obsolete (*bool*) – whether parent has become obsolete

set_fetched(*fetched*)

Sets the fetched status.

Parameters

fetched (*bool*) – whether parent has been fetched completely

set_busy(*busy*)

Sets the busy status.

Parameters

busy (*bool*) – whether parent is busy fetching

abstract handle_items_added(*db_map_data*)

Called by SpineDBWorker when items are added to the DB.

Parameters

db_map_data (*dict*) – Mapping DiffDatabaseMapping instances to list of dict-items for which `accepts_item()` returns True.

abstract handle_items_removed(*db_map_data*)

Called by SpineDBWorker when items are removed from the DB.

Parameters

db_map_data (*dict*) – Mapping DiffDatabaseMapping instances to list of dict-items for which `accepts_item()` returns True.

abstract handle_items_updated(*db_map_data*)

Called by SpineDBWorker when items are updated in the DB.

Parameters

db_map_data (*dict*) – Mapping DiffDatabaseMapping instances to list of dict-items for which `accepts_item()` returns True.

class `spinetoolbox.fetch_parent.ItemTypeFetchParent`(*fetch_item_type*, *owner=None*,
chunk_size=1000)

Bases: [FetchParent](#)

Attrs:

`fetch_token` (int or None) `will_have_children` (bool or None): Whether this parent will have children if fetched. None means we don't know yet. Set to a boolean value whenever we find out.

Parameters

- **owner** (*object*) – somebody who owns this FetchParent. If it's a QObject instance, then this FetchParent
- **destroyed** (*becomes obsolete whenever the owner is*) –
- **chunk_size** (*int or None*) – the number of items this parent should be happy with fetching at a time.
- **None** (*If*) –
- **DB.** (*then no limit is imposed and the parent should fetch the entire contents of the*) –

property `fetch_item_type`

Returns the type of item to fetch, e.g., "object_class".

Returns

str

abstract handle_items_added(*db_map_data*)

Called by SpineDBWorker when items are added to the DB.

Parameters

db_map_data (*dict*) – Mapping DiffDatabaseMapping instances to list of dict-items for which `accepts_item()` returns True.

abstract handle_items_removed(*db_map_data*)

Called by SpineDBWorker when items are removed from the DB.

Parameters

db_map_data (*dict*) – Mapping DiffDatabaseMapping instances to list of dict-items for which `accepts_item()` returns True.

abstract handle_items_updated(*db_map_data*)

Called by SpineDBWorker when items are updated in the DB.

Parameters

db_map_data (*dict*) – Mapping DiffDatabaseMapping instances to list of dict-items for which `accepts_item()` returns True.

__str__()

```
class spinetoolbox.fetch_parent.FlexibleFetchParent(fetch_item_type, handle_items_added=None,
                                                    handle_items_removed=None,
                                                    handle_items_updated=None,
                                                    accepts_item=None, shows_item=None,
                                                    will_have_children_change=None,
                                                    owner=None, chunk_size=1000)
```

Bases: *ItemTypeFetchParent*

Attrs:

fetch_token (int or None) **will_have_children** (bool or None): Whether this parent will have children if fetched. None means we don't know yet. Set to a boolean value whenever we find out.

Parameters

- **owner** (*object*) – somebody who owns this FetchParent. If it's a QObject instance, then this FetchParent
- **destroyed** (*becomes obsolete whenever the owner is*) –
- **chunk_size** (*int or None*) – the number of items this parent should be happy with fetching at a time.
- **None** (*If*) –
- **DB.** (*then no limit is imposed and the parent should fetch the entire contents of the*) –

handle_items_added(*db_map_data*)

Called by SpineDBWorker when items are added to the DB.

Parameters

db_map_data (*dict*) – Mapping DiffDatabaseMapping instances to list of dict-items for which `accepts_item()` returns True.

handle_items_removed(*db_map_data*)

Called by SpineDBWorker when items are removed from the DB.

Parameters

db_map_data (*dict*) – Mapping DiffDatabaseMapping instances to list of dict-items for which `accepts_item()` returns True.

handle_items_updated(*db_map_data*)

Called by SpineDBWorker when items are updated in the DB.

Parameters

db_map_data (*dict*) – Mapping DiffDatabaseMapping instances to list of dict-items for which `accepts_item()` returns True.

accepts_item(*item*, *db_map*)

Called by the associated SpineDBWorker whenever items are fetched and also added/updated/removed. Returns whether this parent accepts that item as a children.

In case of modifications, the SpineDBWorker will call one or more of `handle_items_added()`, `handle_items_updated()`, or `handle_items_removed()` with all the items that pass this test.

Parameters

- **item** (*dict*) – The item
- **db_map** (*DiffDatabaseMapping*) –

Returns

bool

shows_item(*item*, *db_map*)

Called by the associated SpineDBWorker whenever items are fetched and accepted. Returns whether this parent will show this item to the user.

Parameters

- **item** (*dict*) – The item
- **db_map** (*DiffDatabaseMapping*) –

Returns

bool

will_have_children_change()

Called when the `will_have_children` property changes.

spinetoolbox.headless

Contains facilities to open and execute projects without GUI.

Module Contents**Classes**

<i>HeadlessLogger</i>	A <code>LoggerInterface</code> compliant logger that uses Python's standard logging facilities.
<i>ModifiableProject</i>	A simple project that is available for modification script.
<i>ActionsWithProject</i>	A 'task' which opens Toolbox project and operates on it.
<i>Status</i>	Status codes returned from headless execution.

Functions

<code>headless_main(args)</code>	Executes a project using <code>QCoreApplication</code> .
<code>open_project(project_dict, project_dir, logger)</code>	Opens a project.
<code>_specification_dicts(project_dict, project_dir, logger)</code>	Loads project item specification dictionaries.

class `spinetoolbox.headless.HeadlessLogger`

Bases: `PySide6.QtCore.QObject`

A `LoggerInterface` compliant logger that uses Python's standard logging facilities.

msg

Emits a notification message.

msg_success

Emits a message on success

msg_warning

Emits a warning message.

msg_error

Emits an error message.

msg_proc

Emits a message originating from a subprocess (usually something printed to stdout).

msg_proc_error

Emits an error message originating from a subprocess (usually something printed to stderr).

information_box

Requests an 'information message box' (e.g. a message window) to be opened with a given title and message.

error_box

Requests an 'error message box' to be opened with a given title and message.

_log_message(message)

Prints an information message.

_log_warning(message)

Prints a warning message.

_log_error(message)

Prints an error message.

_show_information_box(title, message)

Prints an information message with a title.

_show_error_box(title, message)

Prints an error message with a title.

_print(message, out_stream)

Filters HTML tags from message before printing it to given file.

class spinetoolbox.headless.**ModifiableProject**(*project_dir*, *items_dict*, *connection_dicts*)

A simple project that is available for modification script.

Parameters

- **project_dir** (*Path*) – project directory
- **items_dict** (*dict*) – project item dictionaries
- **connection_dicts** (*list of dict*) – connection dictionaries

property project_dir

find_connection(*source_name*, *destination_name*)

Searches for a connection between given items.

Parameters

- **source_name** (*str*) – source item's name
- **destination_name** (*str*) – destination item's name

Returns

connection instance or None if there is no connection

Return type

Connection

find_item(*name*)

Searches for a project item.

Parameters

name (*str*) – item's name

Returns

item dict or None if no such item exists

Return type

dict

items_to_dict()

Stores project items back to dictionaries.

Returns

item dictionaries

Return type

dict

connections_to_dict()

Stores connections back to dictionaries.

Returns

connection dictionaries

Return type

list of dict

class spinetoolbox.headless.**ActionsWithProject**(*args*, *startup_event_type*, *parent*)

Bases: PySide6.QtCore.QObject

A 'task' which opens Toolbox project and operates on it.

The execution of this task is triggered by sending it a 'startup' QEvent using e.g. QApplication.postEvent()

Parameters

- **args** (*argparse.Namespace*) – parsed command line arguments
- **startup_event_type** (*int*) – expected type id for the event that starts this task
- **parent** (*QObject*) – a parent object

_start

A private signal to actually start execution. Not to be used directly. Post a startup event instead.

_dags()**_execute()**

Executes this task.

_open_project()

Opens a project.

Returns

status code

Return type

Status

_check_project_version(*project_dict*)

Checks project dict version.

Parameters

project_dict (*dict*) – project dict

Returns

status code

Return type

Status

_exec_mod_script()

Executes project modification script given in command line arguments.

Returns

status code

Return type

Status

_execute_project()

Executes all DAGs in a project.

Returns

status code

Return type

Status

_process_engine_event(*event_type, data*)**event(*e*)****_handle_node_execution_started(*data*)**

Starts collecting messages from given node.

Parameters

data (*dict*) – execution start data

_handle_node_execution_finished(*data*)

Prints messages for finished nodes.

Parameters

data (*dict*) – execution end data

_handle_event_msg(*data*)

Stores event messages for later printing.

Parameters

data (*dict*) – event message data

_handle_process_msg(*data*)

Stores process messages for later printing.

Parameters

data (*dict*) – process message data

_handle_standard_execution_msg(*data*)

Handles standard execution messages.

Currently, these messages are ignored.

Parameters

data (*dict*) – execution message data

_handle_persistent_execution_msg(*data*)

Handles persistent execution messages.

Parameters

data (*dict*) – execution message data

_handle_kernel_execution_msg(*data*)

Handles kernel messages.

Currently, these messages are ignored.

Parameters

data (*dict*) – message data

spinetoolbox.headless.headless_main(*args*)

Executes a project using QApplication.

Parameters

args (*argparser.Namespace*) – parsed command line arguments.

Returns

exit status code; 0 for success, everything else for failure

Return type

int

spinetoolbox.headless.open_project(*project_dict*, *project_dir*, *logger*)

Opens a project.

Parameters

- **project_dict** (*dict*) – a serialized project dictionary
- **project_dir** (*Path*) – path to a directory containing the .spinetoolbox dir

- **logger** ([LoggerInterface](#)) – a logger

Returns

item dicts, specification dicts, connection dicts, jump dicts and a DagHandler object

Return type

tuple

`spinetoolbox.headless._specification_dicts(project_dict, project_dir, logger)`

Loads project item specification dictionaries.

Parameters

- **project_dict** (*dict*) – a serialized project dictionary
- **project_dir** (*str*) – path to a directory containing the `.spinetoolbox` dir
- **logger** ([LoggerInterface](#)) – a logger

Returns

a mapping from item type to a list of specification dicts

Return type

dict

class `spinetoolbox.headless.Status`

Bases: `enum.IntEnum`

Status codes returned from headless execution.

Initialize self. See `help(type(self))` for accurate signature.

OK = 0

ERROR = 1

ARGUMENT_ERROR = 2

`spinetoolbox.helpers`

General helper functions and classes.

Module Contents

Classes

<i>LinkType</i>	Graphics scene's link types.
<i>IconListManager</i>	A class to manage icons for icon list widgets.
<i>TransparentIconEngine</i>	Specialization of QIconEngine with transparent background.
<i>CharIconEngine</i>	Specialization of QIconEngine used to draw font-based icons.
<i>ColoredIcon</i>	
<i>ColoredIconEngine</i>	
<i>ProjectDirectoryIconProvider</i>	QFileIconProvider that provides a Spine icon to the
<i>ChildCyclingKeyPressFilter</i>	Event filter class for catching next and previous child key presses.
<i>QuietLogger</i>	
<i>SignalWaiter</i>	A 'traffic light' that allows waiting for a signal to be emitted in another thread.
<i>CustomSyntaxHighlighter</i>	
<i>HTMLTagFilter</i>	HTML tag filter.

Functions

<i>home_dir()</i>	Returns user's home dir
<i>format_log_message</i> (msg_type, message[, show_datetime])	Adds color tags and optional time stamp to message.
<i>busy_effect</i> (func)	Decorator to change the mouse cursor to 'busy' while a function is processed.
<i>create_dir</i> (base_path[, folder, verbosity])	Create (input/output) directories recursively.
<i>rename_dir</i> (old_dir, new_dir, toolbox, box_title)	Renames directory. Called by ProjectItemModel.set_item_name()
<i>open_url</i> (url)	Opens the given url in the appropriate Web browser for the user's desktop environment,
<i>set_taskbar_icon</i> ()	Set application icon to Windows taskbar.
<i>supported_img_formats</i> ()	Checks if reading .ico files is supported.
<i>pyside6_version_check</i> ()	Check that PySide6 version is at least 6.4.
<i>get_datetime</i> (show[, date])	Returns date and time string for appending into Event Log messages.
<i>copy_files</i> (src_dir, dst_dir[, includes, excludes])	Function for copying files. Does not copy folders.
<i>erase_dir</i> (path[, verbosity])	Deletes a directory and all its contents without prompt.
<i>recursive_overwrite</i> (logger, src, dst[, ignore, silent])	Copies everything from source directory to destination directory recursively.
<i>tuple_itemgetter</i> (itemgetter_func, num_indexes)	Change output of itemgetter to always be a tuple even for a single index.
<i>format_string_list</i> (str_list)	Returns a html unordered list from the given list of strings.

continues on next page

Table 1 – continued from previous page

<code>rows_to_row_count_tuples(rows)</code>	Breaks a list of rows into a list of (row, count) tuples to corresponding
<code>object_icon(display_icon)</code>	Creates and returns a QIcon corresponding to display_icon.
<code>color_pixmap(pixmap, color)</code>	
<code>make_icon_id(icon_code, color_code)</code>	Takes icon and color codes, and return equivalent integer.
<code>interpret_icon_id(display_icon)</code>	Takes a display icon id and returns an equivalent tuple of icon and color code.
<code>default_icon_id()</code>	Creates a default icon id.
<code>ensure_window_is_on_screen(window, size)</code>	Checks if window is on screen and if not, moves and resizes it to make it visible on the primary screen.
<code>first_non_null(s)</code>	Returns the first element in Iterable s that is not None.
<code>get_save_file_name_in_last_dir(qsettings, key, parent, ...)</code>	Calls QFileDialog.getSaveFileName in the directory that was selected last time the dialog was accepted.
<code>get_open_file_name_in_last_dir(qsettings, key, parent, ...)</code>	
<code>try_number_from_string(text)</code>	Tries to convert a string to integer or float.
<code>focused_widget_has_callable(parent, callable_name)</code>	Returns True if the currently focused widget or one of its ancestors has the given callable.
<code>call_on_focused_widget(parent, callable_name)</code>	Calls the given callable on the currently focused widget or one of its ancestors.
<code>select_gams_executable(parent, line_edit)</code>	Opens file browser where user can select a Gams executable (i.e. gams.exe on Windows).
<code>select_julia_executable(parent, line_edit)</code>	Opens file browser where user can select a Julia executable (i.e. julia.exe on Windows).
<code>select_julia_project(parent, line_edit)</code>	Shows file browser and inserts selected julia project dir to give line_edit.
<code>select_python_interpreter(parent, line_edit)</code>	Opens file browser where user can select a python interpreter (i.e. python.exe on Windows).
<code>select_conda_executable(parent, line_edit)</code>	Opens file browser where user can select a conda executable.
<code>select_certificate_directory(parent, line_edit)</code>	Shows file browser and inserts selected certificate directory to given line edit.
<code>file_is_valid(parent, file_path, msgbox_title[, ...])</code>	Checks that given path is not a directory and it's a file that actually exists.
<code>dir_is_valid(parent, dir_path, msgbox_title)</code>	Checks that given path is a directory. Needed in
<code>make_settings_dict_for_engine(app_settings)</code>	Converts Toolbox settings to a dictionary acceptable by Engine.
<code>make_icon_background(color)</code>	
<code>make_icon_toolbar_ss(color)</code>	
<code>color_from_index(i, count[, base_hue, saturation])</code>	
<code>unique_name(prefix, existing)</code>	Creates a unique name in the form <i>prefix (xx)</i> where xx is a counter value.
<code>get_upgrade_db_prompt_text(url, current, expected)</code>	
<code>parse_specification_file(spec_path, logger)</code>	Parses specification file.

continues on next page

Table 1 – continued from previous page

<code>load_specification_from_file(spec_path, ...)</code>	Returns an Item specification from a definition file.
<code>specification_from_dict(spec_dict, local_data_dict, ...)</code>	Returns item specification from a dictionary.
<code>plugins_dirs(app_settings)</code>	Loads plugins.
<code>load_plugin_dict(plugin_dir, logger)</code>	Loads plugin dict from plugin directory.
<code>load_plugin_specifications(plugin_dict, ...)</code>	Loads plugin's specifications.
<code>load_specification_local_data(config_dir)</code>	Loads specifications' project-specific data.
<code>parameter_identifier(database, parameter, names, ...)</code>	Concatenates given information into parameter value identifier string.
<code>disconnect(signal, *slots)</code>	Disconnects signal for the duration of a 'with' block.
<code>signal_waiter(signal[, condition, timeout])</code>	
<code>inquire_index_name(model, column, title, parent_widget)</code>	Asks for indexed parameter's index name and updates model accordingly.
<code>preferred_row_height(widget[, factor])</code>	
<code>restore_ui(window, app_settings, settings_group)</code>	Restores UI state from previous session.
<code>save_ui(window, app_settings, settings_group)</code>	Saves UI state for next session.
<code>bisect_chunks(current_data, new_data[, key])</code>	Finds insertion points for chunks of data using binary search.
<code>load_project_dict(project_config_dir, logger)</code>	Loads project dictionary from project directory.
<code>load_local_project_data(project_config_dir, logger)</code>	Loads local project data.
<code>merge_dicts(source, target)</code>	Merges two dictionaries that may contain nested dictionaries recursively.
<code>fix_lightness_color(color[, lightness])</code>	
<code>scrolling_to_bottom(widget[, tolerance])</code>	
<code>_is_metadata_item(item)</code>	Identifies a database metadata record.
<code>separate_metadata_and_item_metadata(db_map_data)</code>	Separates normal metadata items from item metadata items.
<code>same_path(path1, path2)</code>	Checks if two paths are equal.
<code>solve_connection_file(connection_file, ...)</code>	Returns the connection_file path, if it exists on this computer. If the path

Attributes

<code>_matplotlib_version</code>	
<code>DB_ITEM_SEPARATOR</code>	Display string to separate items such as entity names.

`spinetoolbox.helpers._matplotlib_version`

class `spinetoolbox.helpers.LinkType`

Bases: `enum.Enum`

Graphics scene's link types.

CONNECTION = **'connection'**

JUMP = 'jump'

`spinetoolbox.helpers.home_dir()`

Returns user's home dir

`spinetoolbox.helpers.format_log_message(msg_type, message, show_datetime=True)`

Adds color tags and optional time stamp to message.

Parameters

- **msg_type** (*str*) – message's type; accepts only 'msg', 'msg_success', 'msg_warning', or 'msg_error'
- **message** (*str*) – message to format
- **show_datetime** (*bool*) – True to add time stamp, False to omit it

Returns

formatted message

Return type

str

`spinetoolbox.helpers.busy_effect(func)`

Decorator to change the mouse cursor to 'busy' while a function is processed.

Parameters

func (*Callable*) – Decorated function.

`spinetoolbox.helpers.create_dir(base_path, folder="", verbosity=False)`

Create (input/output) directories recursively.

Parameters

- **base_path** (*str*) – Absolute path to wanted dir
- **folder** (*str*) – (Optional) Folder name. Usually short name of item.
- **verbosity** (*bool*) – True prints a message that tells if the directory already existed or if it was created.

Raises

OSError if operation failed. –

`spinetoolbox.helpers.rename_dir(old_dir, new_dir, toolbox, box_title)`

Renames directory. Called by `ProjectItemModel.set_item_name()`

Parameters

- **old_dir** (*str*) – Absolute path to directory that will be renamed
- **new_dir** (*str*) – Absolute path to new directory
- **toolbox** (`ToolboxUI`) – A toolbox to log messages and ask questions.
- **box_title** (*str*) – The title of the message boxes, (e.g. "Undoing 'rename DC1 to DC2'")

Returns

True if operation was successful, False otherwise

Return type

bool

`spinetoolbox.helpers.open_url(url)`

Opens the given url in the appropriate Web browser for the user's desktop environment, and returns true if successful; otherwise returns false.

If the URL is a reference to a local file (i.e., the URL scheme is "file") then it will be opened with a suitable application instead of a Web browser.

Handle return value on caller side.

Parameters

url (*str*) – URL to open

Returns

True if successful, False otherwise

Return type

bool

`spinetoolbox.helpers.set_taskbar_icon()`

Set application icon to Windows taskbar.

`spinetoolbox.helpers.supported_img_formats()`

Checks if reading .ico files is supported.

`spinetoolbox.helpers.pyside6_version_check()`

Check that PySide6 version is at least 6.4.

qt_version (*str*) is the Qt version used to compile PySide6. E.g. "6.4.1" qt_version_info (*tuple*) contains each version component separately e.g. (6, 4, 1)

`spinetoolbox.helpers.get_datetime(show, date=True)`

Returns date and time string for appending into Event Log messages.

Parameters

- **show** (*bool*) – True returns date and time string. False returns empty string.
- **date** (*bool*) – Whether or not the date should be included in the result

Returns

datetime string or empty string if show is False

Return type

str

`spinetoolbox.helpers.copy_files(src_dir, dst_dir, includes=None, excludes=None)`

Function for copying files. Does not copy folders.

Parameters

- **src_dir** (*str*) – Source directory
- **dst_dir** (*str*) – Destination directory
- **includes** (*list*, *optional*) – Included files (wildcards accepted)
- **excludes** (*list*, *optional*) – Excluded files (wildcards accepted)

Returns

Number of files copied

Return type

count (int)

`spinetoolbox.helpers.erase_dir(path, verbosity=False)`

Deletes a directory and all its contents without prompt.

Parameters

- **path** (*str*) – Path to directory
- **verbosity** (*bool*) – Print logging messages or not

Returns

True if operation was successful, False otherwise

Return type

bool

`spinetoolbox.helpers.recursive_overwrite(logger, src, dst, ignore=None, silent=True)`

Copies everything from source directory to destination directory recursively. Overwrites existing files.

Parameters

- **logger** ([LoggerInterface](#)) – Enables e.g. printing to Event Log
- **src** (*str*) – Source directory
- **dst** (*str*) – Destination directory
- **ignore** (*Callable, optional*) – Ignore function
- **silent** (*bool*) – If False, messages are sent to Event Log, If True, copying is done in silence

`spinetoolbox.helpers.tuple_itemgetter(itemgetter_func, num_indexes)`

Change output of itemgetter to always be a tuple even for a single index.

Parameters

- **itemgetter_func** (*Callable*) – item getter function
- **num_indexes** (*int*) – number of indexes

Returns

getter function that works with a single index

Return type

Callable

`spinetoolbox.helpers.format_string_list(str_list)`

Returns a html unordered list from the given list of strings. Intended to print error logs as returned by spinedb_api.

Parameters

str_list (*list of str*) – list of strings to format

Returns

formatted list

Return type

str

`spinetoolbox.helpers.rows_to_row_count_tuples(rows)`

Breaks a list of rows into a list of (row, count) tuples to corresponding chunks of successive rows.

Parameters

rows (*Iterable of int*) – rows

Returns

row count tuples

Return type

list of tuple

class spinetoolbox.helpers.**IconListManager**(*icon_size*)

A class to manage icons for icon list widgets.

Parameters**icon_size** (*QSize*) – icon’s size**init_model**()

Init model that can be used to display all icons in a list.

_model_data(*index*, *role*)

Creates pixmaps as they’re requested by the data() method, to reduce loading time.

Parameters

- **index** (*QModelIndex*) – index to the model
- **role** (*int*) – data role

Returns

role-dependent model data

Return type

Any

spinetoolbox.helpers.**object_icon**(*display_icon*)

Creates and returns a QIcon corresponding to display_icon.

Parameters**display_icon** (*int*) – icon id**Returns**

requested icon

Return type

QIcon

class spinetoolbox.helpers.**TransparentIconEngine**

Bases: PySide6.QtGui.QIconEngine

Specialization of QIconEngine with transparent background.

pixmap(*size=QSize(512, 512)*, *mode=None*, *state=None*)**class** spinetoolbox.helpers.**CharIconEngine**(*char*, *color=None*)Bases: [TransparentIconEngine](#)

Specialization of QIconEngine used to draw font-based icons.

Parameters

- **char** (*str*) – character to use as the icon
- **color** (*QColor*, *optional*) –

paint(*painter*, *rect*, *mode=None*, *state=None*)**class** spinetoolbox.helpers.**ColoredIcon**(*icon_file_name*, *icon_color*, *icon_size*, *colored=None*)

Bases: PySide6.QtGui.QIcon

set_colored(*colored*)

color(*mode=QIcon.Normal*)

class spinetoolbox.helpers.**ColoredIconEngine**(*icon_file_name, icon_color, icon_size, colored=None*)

Bases: PySide6.QtGui.QIconEngine

color(*mode=QIcon.Normal*)

set_colored(*colored*)

_do_make_pixmap(*mode, state*)

_make_pixmap(*mode, state*)

pixmap(*size, mode, state*)

spinetoolbox.helpers.**color_pixmap**(*pixmap, color*)

spinetoolbox.helpers.**make_icon_id**(*icon_code, color_code*)

Takes icon and color codes, and return equivalent integer.

Parameters

- **icon_code** (*int*) – icon’s code
- **color_code** (*int*) – color code

Returns

icon id

Return type

int

spinetoolbox.helpers.**interpret_icon_id**(*display_icon*)

Takes a display icon id and returns an equivalent tuple of icon and color code.

Parameters

display_icon (*int, optional*) – icon id

Returns

icon’s code, color code

Return type

tuple

spinetoolbox.helpers.**default_icon_id**()

Creates a default icon id.

Returns

default icon’s id

Return type

int

class spinetoolbox.helpers.**ProjectDirectoryIconProvider**

Bases: PySide6.QtWidgets.QFileIconProvider

QFileIconProvider that provides a Spine icon to the Open Project Dialog when a Spine Toolbox project directory is encountered.

icon(*info*)

Returns an icon for the file described by *info*.

Parameters

info (*QFileInfo*) – File (or directory) info

Returns

Icon for a file system resource with the given info

Return type

QIcon

spinetoolbox.helpers.ensure_window_is_on_screen(*window*, *size*)

Checks if window is on screen and if not, moves and resizes it to make it visible on the primary screen.

Parameters

- **window** (*QWidget*) – a window to check
- **size** (*QSize*) – desired window size if the window is moved

spinetoolbox.helpers.first_non_null(*s*)

Returns the first element in Iterable *s* that is not None.

spinetoolbox.helpers.get_save_file_name_in_last_dir(*qsettings*, *key*, *parent*, *caption*, *given_dir*, *filter_*=")

Calls QFileDialog.getSaveFileName in the directory that was selected last time the dialog was accepted.

Parameters

- **qsettings** (*QSettings*) – A QSettings object where the last directory is stored
- **key** (*string*) – The name of the entry in the above QSettings
- **parent** – Args passed to QFileDialog.getSaveFileName
- **caption** – Args passed to QFileDialog.getSaveFileName
- **given_dir** – Args passed to QFileDialog.getSaveFileName
- **filter** – Args passed to QFileDialog.getSaveFileName

Returns

filename str: selected filter

Return type

str

spinetoolbox.helpers.get_open_file_name_in_last_dir(*qsettings*, *key*, *parent*, *caption*, *given_dir*, *filter_*=")**spinetoolbox.helpers.try_number_from_string(*text*)**

Tries to convert a string to integer or float.

Parameters

text (*str*) – string to convert

Returns

converted value or text if conversion failed

Return type

int or float or str

`spinetoolbox.helpers.focused_widget_has_callable(parent, callable_name)`

Returns True if the currently focused widget or one of its ancestors has the given callable.

`spinetoolbox.helpers.call_on_focused_widget(parent, callable_name)`

Calls the given callable on the currently focused widget or one of its ancestors.

class `spinetoolbox.helpers.ChildCyclingKeyPressFilter`

Bases: `PySide6.QtCore.QObject`

Event filter class for catching next and previous child key presses. Used in filtering the Ctrl+Tab and Ctrl+Shift+Tab key presses in the Item Properties tab widget.

eventFilter(*obj, event*)

`spinetoolbox.helpers.select_gams_executable(parent, line_edit)`

Opens file browser where user can select a Gams executable (i.e. gams.exe on Windows).

Parameters

- **parent** (*QWidget, optional*) – Parent widget for the file dialog and message boxes
- **line_edit** (*QLineEdit*) – Line edit where the selected path will be inserted

`spinetoolbox.helpers.select_julia_executable(parent, line_edit)`

Opens file browser where user can select a Julia executable (i.e. julia.exe on Windows). Used in SettingsWidget and KernelEditor.

Parameters

- **parent** (*QWidget, optional*) – Parent widget for the file dialog and message boxes
- **line_edit** (*QLineEdit*) – Line edit where the selected path will be inserted

`spinetoolbox.helpers.select_julia_project(parent, line_edit)`

Shows file browser and inserts selected julia project dir to give line_edit. Used in SettingsWidget and KernelEditor.

Parameters

- **parent** (*QWidget, optional*) – Parent of QFileDialog
- **line_edit** (*QLineEdit*) – Line edit where the selected path will be inserted

`spinetoolbox.helpers.select_python_interpreter(parent, line_edit)`

Opens file browser where user can select a python interpreter (i.e. python.exe on Windows). Used in SettingsWidget and KernelEditor.

Parameters

- **parent** (*QWidget*) – Parent widget for the file dialog and message boxes
- **line_edit** (*QLineEdit*) – Line edit where the selected path will be inserted

`spinetoolbox.helpers.select_conda_executable(parent, line_edit)`

Opens file browser where user can select a conda executable.

Parameters

- **parent** (*QWidget*) – Parent widget for the file dialog and message boxes
- **line_edit** (*QLineEdit*) – Line edit where the selected path will be inserted

`spinetoolbox.helpers.select_certificate_directory(parent, line_edit)`

Shows file browser and inserts selected certificate directory to given line edit.

Parameters

- **parent** (*QWidget*, *optional*) – Parent of *QFileDialog*
- **line_edit** (*QLineEdit*) – Line edit where the selected dir path will be inserted

`spinetoolbox.helpers.file_is_valid(parent, file_path, msgbox_title, extra_check=None)`

Checks that given path is not a directory and it's a file that actually exists. In addition, can be used to check if the file name in given file path starts with the given *extra_check* string. Needed in *SettingsWidget* and *KernelEditor* because the *QLineEdit*s are editable. Returns *True* when *file_path* is an empty string so that we can use default values (e.g. from line edit place holder text). Returns also *True* when *file_path* is just 'python' or 'julia' so that user's can use the python or julia in *PATH*.

Parameters

- **parent** (*QWidget*) – Parent widget for the message boxes
- **file_path** (*str*) – Path to check
- **msgbox_title** (*str*) – Title for message boxes
- **extra_check** (*str*, *optional*) – String that must match the file name of the given *file_path* (without extension)

Returns

True if given path is an empty string or if path is valid, *False* otherwise

Return type

bool

`spinetoolbox.helpers.dir_is_valid(parent, dir_path, msgbox_title)`

Checks that given path is a directory. Needed in *SettingsWidget* and *KernelEditor* because the *QLineEdit*s are editable. Returns *True* when *dir_path* is an empty string so that we can use default values (e.g. from line edit place holder text)

Parameters

- **parent** (*QWidget*) – Parent widget for the message box
- **dir_path** (*str*) – Directory path to check
- **msgbox_title** (*str*) – Message box title

Returns

True if given path is an empty string or if path is an existing directory, *False* otherwise

Return type

bool

`class spinetoolbox.helpers.QuietLogger`

`__getattr__(_)`

`__call__(*args, **kwargs)`

`spinetoolbox.helpers.make_settings_dict_for_engine(app_settings)`

Converts Toolbox settings to a dictionary acceptable by Engine.

Parameters

app_settings (*QSettings*) – Toolbox settings

Returns

Engine-compatible settings

Return type

dict

`spinetoolbox.helpers.make_icon_background(color)`

`spinetoolbox.helpers.make_icon_toolbar_ss(color)`

`spinetoolbox.helpers.color_from_index(i, count, base_hue=0.0, saturation=1.0)`

`spinetoolbox.helpers.unique_name(prefix, existing)`

Creates a unique name in the form *prefix* (*xx*) where *xx* is a counter value. When *prefix* already contains a counter (*xx*), the value *xx* is updated.

Parameters

- **prefix** (*str*) – name prefix
- **existing** (*Iterable of str*) – existing names

Returns

unique name

Return type

str

`spinetoolbox.helpers.get_upgrade_db_prompt_text(url, current, expected)`

`spinetoolbox.helpers.parse_specification_file(spec_path, logger)`

Parses specification file.

Parameters

- **spec_path** (*str*) – path to specification file
- **logger** ([LoggerInterface](#)) – a logger

Returns

specification dict or None if the operation failed

Return type

dict

`spinetoolbox.helpers.load_specification_from_file(spec_path, local_data_dict, spec_factories, app_settings, logger)`

Returns an Item specification from a definition file.

Parameters

- **spec_path** (*str*) – Path of the specification definition file
- **local_data_dict** (*dict*) – specifications local data dict
- **spec_factories** (*dict*) – Dictionary mapping specification type to ProjectItemSpecificationFactory
- **app_settings** (*QSettings*) – Toolbox settings
- **logger** ([LoggerInterface](#)) – a logger

Returns

item specification or None if reading the file failed

Return type

ProjectItemSpecification

`spinetoolbox.helpers.specification_from_dict(spec_dict, local_data_dict, spec_factories, app_settings, logger)`

Returns item specification from a dictionary.

Parameters

- **spec_dict** (*dict*) – Dictionary with the specification
- **local_data_dict** (*dict*) – specifications local data
- **spec_factories** (*dict*) – Dictionary mapping specification name to ProjectItemSpecificationFactory
- **app_settings** (*QSettings*) – Toolbox settings
- **logger** (*LoggerInterface*) – a logger

Returns

specification or None if factory isn't found.

Return type

ProjectItemSpecType or NoneType

`spinetoolbox.helpers.plugins_dirs(app_settings)`

Loads plugins.

Parameters

app_settings (*QSettings*) – Toolbox settings

Returns

plugin directories

Return type

list of str

`spinetoolbox.helpers.load_plugin_dict(plugin_dir, logger)`

Loads plugin dict from plugin directory.

Parameters

- **plugin_dir** (*str*) – path of plugin dir with “plugin.json” in it
- **logger** (*LoggerInterface*) – a logger

Returns

plugin dict or None if the operation failed

Return type

dict

`spinetoolbox.helpers.load_plugin_specifications(plugin_dict, local_data_dict, spec_factories, app_settings, logger)`

Loads plugin's specifications.

Parameters

- **plugin_dict** (*dict*) – plugin dict
- **local_data_dict** (*dict*) – specifications local data dictionary
- **spec_factories** (*dict*) – Dictionary mapping specification name to ProjectItemSpecificationFactory

- **app_settings** (*QSettings*) – Toolbox settings
- **logger** (*LoggerInterface*) – a logger

Returns

mapping from plugin name to list of specifications or None if the operation failed

Return type

dict

`spinetoolbox.helpers.load_specification_local_data(config_dir)`

Loads specifications' project-specific data.

Parameters

config_dir (*str or Path*) – project config dir

Returns

specifications local data

Return type

dict

`spinetoolbox.helpers.DB_ITEM_SEPARATOR = ' '`

Display string to separate items such as entity names.

`spinetoolbox.helpers.parameter_identifier(database, parameter, names, alternative)`

Concatenates given information into parameter value identifier string.

Parameters

- **database** (*str, optional*) – database's code name
- **parameter** (*str*) – parameter's name
- **names** (*list of str*) – name of the entity or class that holds the value
- **alternative** (*str or NoneType*) – name of the value's alternative

`spinetoolbox.helpers.disconnect(signal, *slots)`

Disconnects signal for the duration of a 'with' block.

Parameters

- **signal** (*Signal*) – signal to disconnect
- ***slots** – slots to disconnect from

`class spinetoolbox.helpers.SignalWaiter(condition=None, timeout=None)`

Bases: `PySide6.QtCore.QObject`

A 'traffic light' that allows waiting for a signal to be emitted in another thread.

Parameters

- **condition** (*function, optional*) – receiving the self.args and returning whether to stop waiting.
- **timeout** (*float, optional*) – timeout in seconds; wait will raise after timeout

trigger(**args*)

Signal receiving slot.

wait()

Wait for signal to be received.

`spinetoolbox.helpers.signal_waiter(signal, condition=None, timeout=None)`

class `spinetoolbox.helpers.CustomSyntaxHighlighter(*arg, **kwargs)`

Bases: `PySide6.QtGui.QSyntaxHighlighter`

property `formats`

set_style(*style*)

yield_formats(*text*)

highlightBlock(*text*)

`spinetoolbox.helpers.inquire_index_name(model, column, title, parent_widget)`

Asks for indexed parameter's index name and updates model accordingly.

Parameters

- **model** (`IndexedValueTableModel` or `ArrayModel`) – a model with header that contains index names
- **column** (`int`) – column index
- **title** (`str`) – input dialog's title
- **parent_widget** (`QWidget`) – dialog's parent widget

`spinetoolbox.helpers.preferred_row_height(widget, factor=1.5)`

`spinetoolbox.helpers.restore_ui(window, app_settings, settings_group)`

Restores UI state from previous session.

Parameters

- **window** (`QMainWindow`) –
- **app_settings** (`QSettings`) –
- **settings_group** (`str`) –

`spinetoolbox.helpers.save_ui(window, app_settings, settings_group)`

Saves UI state for next session.

Parameters

- **window** (`QMainWindow`) –
- **app_settings** (`QSettings`) –
- **settings_group** (`str`) –

`spinetoolbox.helpers.bisect_chunks(current_data, new_data, key=None)`

Finds insertion points for chunks of data using binary search.

Parameters

- **current_data** (`list`) – sorted list where to insert new data
- **new_data** (`list`) – data to insert
- **key** (`Callable`, *optional*) – sort key

Returns

sorted chunk of new data, insertion position

Return type

tuple

`spinetoolbox.helpers.load_project_dict(project_config_dir, logger)`

Loads project dictionary from project directory.

Parameters

- **project_config_dir** (*str*) – project’s .spinetoolbox directory
- **logger** ([LoggerInterface](#)) – a logger

Returns

project dictionary

Return type

dict

`spinetoolbox.helpers.load_local_project_data(project_config_dir, logger)`

Loads local project data.

Parameters

- **project_config_dir** (*Path or str*) – project’s .spinetoolbox directory
- **logger** ([LoggerInterface](#)) – a logger

Returns

project’s local data

Return type

dict

`spinetoolbox.helpers.merge_dicts(source, target)`

Merges two dictionaries that may contain nested dictionaries recursively.

Parameters

- **source** (*dict*) – dictionary that will be merged to **target**
- **target** (*dict*) – target dictionary

`spinetoolbox.helpers.fix_lightness_color(color, lightness=240)`

`spinetoolbox.helpers.scrolling_to_bottom(widget, tolerance=1)`

`spinetoolbox.helpers._is_metadata_item(item)`

Identifies a database metadata record.

Parameters

item (*dict*) – database item

Returns

True if item is metadata item, False otherwise

Return type

bool

`spinetoolbox.helpers.separate_metadata_and_item_metadata(db_map_data)`

Separates normal metadata items from item metadata items.

Parameters

db_map_data (*dict*) – database records

Returns

item metadata records and metadata records

Return type

tuple

class `spinetoolbox.helpers.HTMLTagFilter`

Bases: `html.parser.HTMLParser`

HTML tag filter.

Initialize and reset this instance.

If `convert_charrefs` is `True` (the default), all character references are automatically converted to the corresponding Unicode characters.

drain()

handle_data(*data*)

handle_starttag(*tag*, *attrs*)

`spinetoolbox.helpers.same_path`(*path1*, *path2*)

Checks if two paths are equal.

This is a lightweight version of `os.path.samefile()`: it doesn't check if the paths point to the same file system object but rather takes into account file system case-sensitivity and such.

Parameters

- **path1** (*str*) – a path
- **path2** (*str*) – a path

Returns

True if paths point to the same

Return type

bool

`spinetoolbox.helpers.solve_connection_file`(*connection_file*, *connection_file_dict*)

Returns the `connection_file` path, if it exists on this computer. If the path doesn't exist, assume that it points to a path on another computer, in which case store the contents of `connection_file_dict` into a tempfile.

Parameters

- **connection_file** (*str*) – Path to a connection file
- **connection_file_dict** (*dict*) –

Returns

Path to a connection file on this computer.

Return type

str

spinetoolbox.kernel_fetcher

Contains a class for fetching kernel specs in a thread.

Module Contents

Classes

<i>KernelFetcher</i>	Worker class for retrieving local kernels.
----------------------	--

class spinetoolbox.kernel_fetcher.**KernelFetcher**(conda_path, fetch_mode=1)

Bases: PySide6.QtCore.QThread

Worker class for retrieving local kernels.

Parameters

- **conda_path** (*str*) – Path to (mini)conda executable
- **fetch_mode** (*int*) – 1: Fetch all kernels, 2: Fetch regular and Conda Python kernels, 3: Fetch only regular Python kernels, 4: Fetch only regular Julia kernels, 5: Fetch kernels that are neither Python nor Julia

kernel_found

stop_fetcher

stop_thread()

Slot for handling a request to stop the thread.

get_all_regular_kernels()

Finds all kernel specs as quickly as possible.

get_all_conda_kernels()

Finds auto-generated Conda kernels.

run()

Finds kernel specs based on selected fetch mode. Sends found kernels one-by-one via signals.

static get_icon(p)

Retrieves the kernel's icon. First tries to find the .svg icon then .png's.

Parameters

p (*str*) – Path to Kernel's resource directory

Returns

Kernel's icon or a null icon if icon was not found.

Return type

QIcon

static get_kernel_deats(kernel_path)

Reads kernel.json from given kernel's resource dir and returns the details in a dictionary.

Parameters

kernel_path (*str*) – Full path to kernel resource directory

Returns

language (str), path to executable (str), display name (str), project (str) (NA for Python kernels)

Return type

dict

spinetoolbox.link

Classes for drawing graphics items on QGraphicsScene.

Module Contents**Classes**

<i>LinkBase</i>	Base class for Link and LinkDrawer.
<i>_IconBase</i>	Base class for icons to show over a Link.
<i>_SvgIcon</i>	A svg icon to show over a Link.
<i>_TextIcon</i>	A font awesome icon to show over a Link.
<i>_WarningTextIcon</i>	A font awesome icon to show over a Link.
<i>JumpOrLink</i>	Base class for Jump and Link.
<i>Link</i>	A graphics item to represent the connection between two project items.
<i>JumpLink</i>	A graphics icon to represent a jump connection between items.
<i>LinkDrawerBase</i>	A base class for items intended for drawing links between project items.
<i>ConnectionLinkDrawer</i>	An item for drawing connection links between project items.
<i>JumpLinkDrawer</i>	An item for drawing jump connections between project items.

Functions

<i>_regular_poligon_points(n, side[, initial_angle])</i>
--

Attributes

<i>LINK_COLOR</i>
<i>JUMP_COLOR</i>

spinetoolbox.link.LINK_COLOR

spinetoolbox.link.JUMP_COLOR

class `spinetoolbox.link.LinkBase(toolbox, src_connector, dst_connector)`

Bases: `PySide6.QtWidgets.QGraphicsPathItem`

Base class for Link and LinkDrawer.

Mainly provides the `update_geometry` method for ‘drawing’ the link on the scene.

Parameters

- **toolbox** (`ToolboxUI`) – main UI class instance
- **src_connector** (`ConnectorButton`, *optional*) – Source connector button
- **dst_connector** (`ConnectorButton`) – Destination connector button

property `outline_color`

property `magic_number`

property `src_rect`

Returns the scene rectangle of the source connector.

property `src_center`

Returns the center point of the source rectangle.

property `dst_rect`

Returns the scene rectangle of the destination connector.

property `dst_center`

Returns the center point of the destination rectangle.

_COLOR

shape()

moveBy(_dx, _dy)

Does nothing. This item is not moved the regular way, but follows the `ConnectorButtons` it connects.

update_geometry(*curved_links=None*)

Updates geometry.

guide_path()

For tests.

_do_update_geometry()

Sets the path for this item.

_add_ellipse_path(*path*)

Adds an ellipse for the link’s base.

Parameters

QPainterPath –

_get_joint_angle()

_add_arrow_path(*path*)

Returns an arrow path for the link’s tip.

Parameters

QPainterPath –

static `_get_offset(button)`

`_get_src_offset()`

`_get_dst_offset()`

`_find_new_point(points, target)`

Finds a new point that approximates points to target in a smooth trajectory. Returns the new point, or None if no need for approximation.

Parameters

- **points** (*list(QPointF)*) –
- **target** (*QPointF*) –

Returns

QPointF or None

`_close_enough(p1, p2)`

`_make_guide_path(curved_links=False)`

Returns a ‘narrow’ path connecting this item’s source and destination.

Parameters

curved_links (*bool*) – Whether the path should follow a curved line or just a straight line

Returns

QPainterPath

`itemChange(change, value)`

Wipes out the link when removed from scene.

`wipe_out()`

Removes any trace of this item from the system.

class `spinetoolbox.link._IconBase(x, y, w, h, parent, tooltip=None, active=True)`

Bases: `PySide6.QtWidgets.QGraphicsEllipseItem`

Base class for icons to show over a Link.

hoverEnterEvent(event)

hoverLeaveEvent(event)

class `spinetoolbox.link._SvgIcon(parent, extent, path, tooltip=None, active=False)`

Bases: `_IconBase`

A svg icon to show over a Link.

wipe_out()

Cleans up icon’s resources.

class `spinetoolbox.link._TextIcon(parent, extent, char, tooltip=None, active=False)`

Bases: `_IconBase`

A font awesome icon to show over a Link.

wipe_out()

Cleans up icon’s resources.

class `spinetoolbox.link._WarningTextIcon`(*parent, extent, char, tooltip*)

Bases: `_TextIcon`

A font awesome icon to show over a Link.

class `spinetoolbox.link.JumpOrLink`(*toolbox, src_connector, dst_connector*)

Bases: `LinkBase`

Base class for Jump and Link.

Parameters

- **toolbox** (`ToolboxUI`) – main UI class instance
- **src_connector** (`ConnectorButton`, *optional*) – Source connector button
- **dst_connector** (`ConnectorButton`) – Destination connector button

abstract property item

`_do_update_geometry()`

See base class.

`_place_icons()`

`mousePressEvent(e)`

Ignores event if there's a connector button underneath, to allow creation of new links.

Parameters

e (`QGraphicsSceneMouseEvent`) – Mouse event

`contextMenuEvent(e)`

Selects the link and shows context menu.

Parameters

e (`QGraphicsSceneMouseEvent`) – Mouse event

`paint(painter, option, widget=None)`

Sets a dashed pen if selected.

`shape()`

`wipe_out()`

Removes any trace of this item from the system.

`_make_execution_animation()`

Returns an animation to play when execution 'passes' through this link.

Returns

`QVariantAnimation`

`run_execution_animation()`

Runs execution animation.

`_handle_execution_animation_value_changed(step)`

class `spinetoolbox.link.Link`(*toolbox, src_connector, dst_connector, connection*)

Bases: `JumpOrLink`

A graphics item to represent the connection between two project items.

Parameters

- **toolbox** ([ToolboxUI](#)) – main UI class instance
- **src_connector** ([ConnectorButton](#)) – Source connector button
- **dst_connector** ([ConnectorButton](#)) – Destination connector button
- **connection** ([LoggingConnection](#)) – connection this link represents

property name

property connection

property item

_COLOR

_MEMORY = '\uf538'

_FILTERS = '\uf0b0'

_PURGE = '\uf0e7'

_WARNING = '\uf06a'

_DATAPACKAGE = ':/icons/datapkg.svg'

update_icons()

itemChange(*change, value*)

Brings selected link to top.

class spinetoolbox.link.**JumpLink**(*toolbox, src_connector, dst_connector, jump*)

Bases: [JumpOrLink](#)

A graphics icon to represent a jump connection between items.

Parameters

- **toolbox** ([ToolboxUI](#)) – main UI class instance
- **src_connector** ([ConnectorButton](#)) – Source connector button
- **dst_connector** ([ConnectorButton](#)) – Destination connector button
- **jump** (*spine_engine.project_item.connection.Jump*) – connection this link represents

property **jump**

property **item**

property **name**

_COLOR

_ISSUE = '\uf071'

issues()

Checks if jump is well-defined.

Returns

issues regarding the jump

Return type

list of str

update_icons()

class spinetoolbox.link.LinkDrawerBase(*toolbox*)

Bases: [LinkBase](#)

A base class for items intended for drawing links between project items.

Parameters

toolbox ([ToolboxUI](#)) – main UI class instance

property src_rect

Returns the scene rectangle of the source connector.

property dst_rect

Returns the scene rectangle of the destination connector.

property dst_center

Returns the center point of the destination rectangle.

_get_dst_offset()

abstract add_link()

Makes link between source and destination connectors.

wake_up(*src_connector*)

Sets the source connector, shows this item and adds it to the scene. After calling this, the scene is in link drawing mode.

Parameters

src_connector ([ConnectorButton](#)) – source connector

sleep()

Removes this drawer from the scene, clears its source and destination connectors, and hides it. After calling this, the scene is no longer in link drawing mode.

class spinetoolbox.link.ConnectionLinkDrawer(*toolbox*)

Bases: [LinkDrawerBase](#)

An item for drawing connection links between project items.

Parameters

toolbox ([ToolboxUI](#)) – main UI class instance

_COLOR

add_link()

Makes link between source and destination connectors.

wake_up(*src_connector*)

Sets the source connector, shows this item and adds it to the scene. After calling this, the scene is in link drawing mode.

Parameters

src_connector ([ConnectorButton](#)) – source connector

sleep()

Removes this drawer from the scene, clears its source and destination connectors, and hides it. After calling this, the scene is no longer in link drawing mode.

class `spinetoolbox.link.JumpLinkDrawer(toolbox)`

Bases: `LinkDrawerBase`

An item for drawing jump connections between project items.

Parameters

toolbox (`ToolboxUI`) – main UI class instance

_COLOR

add_link()

Makes link between source and destination connectors.

`spinetoolbox.link._regular_poligon_points(n, side, initial_angle=0)`

spinetoolbox.load_project_items

Functions to load project item modules.

Module Contents

Functions

<code>load_project_items(items_package_name)</code>	Loads project item modules.
<code>_find_module_material(module)</code>	

`spinetoolbox.load_project_items.load_project_items(items_package_name)`

Loads project item modules.

Parameters

items_package_name (*str*) – name of the package that contains the project items

Returns

two dictionaries; first maps item type to its category
while second maps item type to item factory

Return type

tuple of dict

`spinetoolbox.load_project_items._find_module_material(module)`

spinetoolbox.log_mixin

Contains LogMixin.

Module Contents

Classes

LogMixin

class spinetoolbox.log_mixin.**LogMixin**

add_log_message(*filter_id*, *message*)

Adds a message to the log document.

Parameters

- **filter_id** (*str*) – filter identifier
- **message** (*str*) – formatted message

add_event_message(*filter_id*, *msg_type*, *msg_text*)

Adds a message to the log document.

Parameters

- **filter_id** (*str*) – filter identifier
- **msg_type** (*str*) – message type
- **msg_text** (*str*) – message text

add_process_message(*filter_id*, *msg_type*, *msg_text*)

Adds a message to the log document.

Parameters

- **filter_id** (*str*) – filter identifier
- **msg_type** (*str*) – message type
- **msg_text** (*str*) – message text

spinetoolbox.logger_interface

A logger interface.

Module Contents

Classes

LoggerInterface

Placeholder for signals that can be emitted to send messages to an output device.

class `spinetoolbox.logger_interface.LoggerInterface`

Bases: `PySide6.QtCore.QObject`

Placeholder for signals that can be emitted to send messages to an output device.

The signals should be connected to a concrete logging system.

Currently, this is just a ‘model interface’. `ToolboxUI` contains the same signals so it can be used as a drop-in replacement for this class.

msg

Emits a notification message.

msg_success

Emits a message on success

msg_warning

Emits a warning message.

msg_error

Emits an error message.

msg_proc

Emits a message originating from a subprocess (usually something printed to stdout).

msg_proc_error

Emits an error message originating from a subprocess (usually something printed to stderr).

information_box

Requests an ‘information message box’ (e.g. a message window) to be opened with a given title and message.

error_box

Requests an ‘error message box’ to be opened with a given title and message.

`spinetoolbox.main`

Provides the `main()` function.

Module Contents**Functions**

<code>main()</code>	Creates main window GUI and starts main event loop.
<code>_make_argument_parser()</code>	Returns a command line argument parser configured for Toolbox use.
<code>_add_pywin32_system32_to_path()</code>	Adds a directory to PATH on Windows that is required to make pywin32 work

Attributes

dirname

plugin_path

`spinetoolbox.main.dirname`

`spinetoolbox.main.plugin_path`

`spinetoolbox.main.main()`

Creates main window GUI and starts main event loop.

`spinetoolbox.main._make_argument_parser()`

Returns a command line argument parser configured for Toolbox use.

Returns

Toolbox' command line argument parser

Return type

ArgumentParser

`spinetoolbox.main._add_pywin32_system32_to_path()`

Adds a directory to PATH on Windows that is required to make pywin32 work on (Conda) Python 3.8. See <https://github.com/spine-tools/Spine-Toolbox/issues/1230>.

`spinetoolbox.metaobject`

MetaObject class.

Module Contents

Classes

MetaObject

Class for an object which has a name, type, and some description.

class `spinetoolbox.metaobject.MetaObject(name, description)`

Bases: `PySide6.QtCore.QObject`

Class for an object which has a name, type, and some description.

Parameters

- **name** (*str*) – Object name
- **description** (*str*) – Object description

set_name(*name*)

Set object name and short name. Note: Check conflicts (e.g. name already exists) before calling this method.

Parameters**name** (*str*) – New (long) name for this object**set_description**(*description*)

Set object description.

Parameters**description** (*str*) – Object description**spinetoolbox.plotting**

Functions for plotting on PlotWidget.

Module Contents**Classes**

<i>PlotType</i>	Generic enumeration.
<i>IndexName</i>	
<i>XYData</i>	Two-dimensional data for plotting.
<i>TreeNode</i>	A labeled node in tree structure.
<i>ParameterTableHeaderSection</i>	Header section info for Database editor's parameter tables.
<i>_PlotStackedBars</i>	

Functions

<i>convert_indexed_value_to_tree</i> (value)	Converts indexed values to tree nodes recursively.
<i>turn_node_to_xy_data</i> (root_node, y_label_position[, ...])	Constructs plottable data and indexes recursively.
<i>raise_if_not_common_x_labels</i> (data_list)	Raises an exception if data has different x axis labels.
<i>raise_if_incompatible_x</i> (data_list)	Raises an exception if the types of x data don't match.
<i>reduce_indexes</i> (data_list)	Removes redundant indexes from given XYData.
<i>combine_data_with_same_indexes</i> (data_list)	Combines data with same data indexes into the same x axis.
<i>_always_single_y_axis</i> (plot_type)	Returns True if a single y-axis should be used.
<i>plot_data</i> (data_list[, plot_widget, plot_type])	Returns a plot widget with plots of the given data.
<i>_plot_single_y_axis</i> (data_list, y_label, axes, plot_type)	Plots all data on single y-axis.
<i>_plot_stacked_line</i> (data_list, y_label, axes)	Plots all data as stacked lines.
<i>_plot_bar</i> (data_list, y_label, axes)	Plots all data as bars.
<i>_plot_double_y_axis</i> (data_list, y_labels, plot_widget, ...)	Plots all data on two y-axes.
<i>_make_x_plottable</i> (xs)	Converts x-axis values to something matplotlib can handle.

continues on next page

Table 2 – continued from previous page

<code>_make_plot_function(plot_type, x_data_type, axes)</code>	Decides plot method and default keyword arguments based on XYData.
<code>_plot_or_step(x_data_type, axes)</code>	Makes choice between Axes.plot() and Axes.step().
<code>_bar(x, y, axes, **kwargs)</code>	Plots bar chart on axes but returns patches instead of bar container.
<code>_group_bars(data_list)</code>	Gives data with same x small offsets to prevent bar stacking.
<code>_clear_plot(plot_widget)</code>	Removes plots and legend from plot widget.
<code>_limit_string_x_tick_labels(data, plot_widget)</code>	Limits the number of x tick labels in case x-axis consists of strings.
<code>_table_display_row(row)</code>	Calculates a human-readable row number.
<code>plot_parameter_table_selection(model, model_indexes, ...)</code>	Returns a plot widget with plots of the selected indexes.
<code>plot_value_editor_table_selection(model, model_indexes)</code>	Returns a plot widget with plots of the selected indexes.
<code>plot_pivot_table_selection(model, model_indexes[, ...])</code>	Returns a plot widget with plots of the selected indexes.
<code>plot_db_mgr_items(items, db_maps[, plot_widget])</code>	Returns a plot widget with plots of database manager parameter value items.
<code>_has_x_column(model, source_model)</code>	Checks if pivot source model has x column.
<code>_set_default_node(root_node, key, label)</code>	Gets node from the contents of root_node adding a new node if necessary.
<code>_get_parsed_value(model_index, display_row)</code>	Gets parsed value from model.
<code>_pivot_index_names(indexes)</code>	Gathers index names from pivot table.
<code>_pivot_display_row(row, source_model)</code>	Calculates display row for pivot table.
<code>_convert_to_leaf(y)</code>	Converts parameter value to leaf TreeElement.
<code>add_row_to_exception(row, display_row)</code>	Adds row information to PlottingError if it is raised in the with block.
<code>add_array_plot(plot_widget, value)</code>	Adds an array plot to a plot widget.
<code>add_time_series_plot(plot_widget, value)</code>	Adds a time series step plot to a plot widget.

Attributes

<code>LEGEND_PLACEMENT_THRESHOLD</code>
<code>_BASE_SETTINGS</code>
<code>_SCATTER_PLOT_SETTINGS</code>
<code>_LINE_PLOT_SETTINGS</code>
<code>_SCATTER_LINE_PLOT_SETTINGS</code>
<code>_TIME_SERIES_PLOT_SETTINGS</code>

```
spinetoolbox.plotting.LEGEND_PLACEMENT_THRESHOLD = 8
```

```
class spinetoolbox.plotting.PlotType
```

```
Bases: enum.Enum
```

Generic enumeration.

Derive from this class to define new enumerations.

SCATTER

SCATTER_LINE

LINE

STACKED_LINE

BAR

STACKED_BAR

`spinetoolbox.plotting._BASE_SETTINGS`

`spinetoolbox.plotting._SCATTER_PLOT_SETTINGS`

`spinetoolbox.plotting._LINE_PLOT_SETTINGS`

`spinetoolbox.plotting._SCATTER_LINE_PLOT_SETTINGS`

`spinetoolbox.plotting._TIME_SERIES_PLOT_SETTINGS`

exception `spinetoolbox.plotting.PlottingError`

Bases: `Exception`

An exception signalling failure in plotting.

Initialize self. See `help(type(self))` for accurate signature.

class `spinetoolbox.plotting.IndexName`

label: `str`

id: `int`

class `spinetoolbox.plotting.XYData`

Two-dimensional data for plotting.

x: `List[Union[float, int, str, numpy.datetime64]]`

y: `List[Union[float, int]]`

x_label: `IndexName`

y_label: `str`

data_index: `List[str]`

index_names: `List[IndexName]`

class `spinetoolbox.plotting.TreeNode`

A labeled node in tree structure.

label: `Union[str, IndexName]`

content: `Dict`

class `spinetoolbox.plotting.ParameterTableHeaderSection`

Header section info for Database editor's parameter tables.

label: `str`

separator: `Optional[str]`

`spinetoolbox.plotting.convert_indexed_value_to_tree(value)`

Converts indexed values to tree nodes recursively.

Parameters

value (*IndexedValue*) – value to convert

Returns

root node of the converted tree

Return type

TreeNode

Raises

ValueError – raised when leaf value couldn't be converted to float

`spinetoolbox.plotting.turn_node_to_xy_data(root_node, y_label_position, index_names=None, indexes=None)`

Constructs plottable data and indexes recursively.

Parameters

- **root_node** (*TreeNode*) – root node
- **y_label_position** (*int*, *optional*) – position of y label in indexes
- **index_names** (*list of IndexName*, *optional*) – list of current index names
- **indexes** (*list*) – list of current indexes

Yields

XYData – plot data

`spinetoolbox.plotting.raise_if_not_common_x_labels(data_list)`

Raises an exception if data has different x axis labels.

Parameters

data_list (*list of XYData*) – data to check

Raises

PlottingError – raised if x axis labels don't match.

`spinetoolbox.plotting.raise_if_incompatible_x(data_list)`

Raises an exception if the types of x data don't match.

Parameters

data_list (*list of XYData*) – data to check

Raises

PlottingError – raised if x data types don't match.

`spinetoolbox.plotting.reduce_indexes(data_list)`

Removes redundant indexes from given XYData.

Parameters

data_list (*list of XYData*) – data to reduce

Returns

reduced data list and list of common data indexes

Return type

tuple

`spinetoolbox.plotting.combine_data_with_same_indexes(data_list)`

Combines data with same data indexes into the same x axis.

Parameters

data_list (*list of XYData*) – data to combine

Returns

combined data

Return type

list of XYData

`spinetoolbox.plotting._always_single_y_axis(plot_type)`

Returns True if a single y-axis should be used.

Parameters

plot_type (*PlotType*) – plot type

Returns

True if single y-axis is required, False otherwise

Return type

bool

`spinetoolbox.plotting.plot_data(data_list, plot_widget=None, plot_type=None)`

Returns a plot widget with plots of the given data.

Parameters

- **data_list** (*list of XYData*) – data to plot
- **plot_widget** (*PlotWidget*, *optional*) – an existing plot widget to draw into or None to create a new widget
- **plot_type** (*PlotType*, *optional*) – plot type

Returns

a PlotWidget object

`spinetoolbox.plotting._plot_single_y_axis(data_list, y_label, axes, plot_type)`

Plots all data on single y-axis.

Parameters

- **data_list** (*list of XYData*) – data to plot
- **y_label** (*str*) – y-axis label
- **axes** (*Axes*) – plot axes
- **plot_type** (*PlotType*) – plot type

Returns

legend handles

Return type

list

`spinetoolbox.plotting._plot_stacked_line(data_list, y_label, axes)`

Plots all data as stacked lines.

Parameters

- **data_list** (*list of XYData*) – data to plot
- **y_label** (*str*) – y-axis label
- **axes** (*Axes*) – plot axes

Returns

legend handles

Return type

list

`spinetoolbox.plotting._plot_bar(data_list, y_label, axes)`

Plots all data as bars.

Parameters

- **data_list** (*list of XYData*) – data to plot
- **y_label** (*str*) – y-axis label
- **axes** (*Axes*) – plot axes

Returns

legend handles

Return type

list

`spinetoolbox.plotting._plot_double_y_axis(data_list, y_labels, plot_widget, plot_type)`

Plots all data on two y-axes.

Parameters

- **data_list** (*list of XYData*) – data to plot
- **y_labels** (*list of str*) – y-axis labels
- **plot_widget** (*PlotWidget*) – plot widget
- **plot_type** (*PlotType*) – plot type

Returns

legend handles

Return type

list

`spinetoolbox.plotting._make_x_plottable(xs)`

Converts x-axis values to something matplotlib can handle.

Parameters

xs (*list*) – x values

Returns

x values

Return type

list

class spinetoolbox.plotting._PlotStackedBars(*axes*)

 __call__(*x*, *height*, ****kwargs**)

spinetoolbox.plotting._make_plot_function(*plot_type*, *x_data_type*, *axes*)

Decides plot method and default keyword arguments based on XYData.

Parameters

- **plot_type** (*PlotType*) – plot type
- **x_data_type** (*Type*) – data type of x-axis
- **axes** (*Axes*) – plot axes

Returns

plot method

Return type

Callable

spinetoolbox.plotting._plot_or_step(*x_data_type*, *axes*)

Makes choice between Axes.plot() and Axes.step().

Parameters

- **x_data_type** (*Type*) – data type of x-axis
- **axes** (*Axes*) – plot axes

spinetoolbox.plotting._bar(*x*, *y*, *axes*, ****kwargs**)

Plots bar chart on axes but returns patches instead of bar container.

Parameters

- **x** (*Any*) – x data
- **y** (*Any*) – y data
- **axes** (*Axes*) – plot axes
- ****kwargs** – keyword arguments passed to bar()

Returns

patches

Return type

list of Patch

spinetoolbox.plotting._group_bars(*data_list*)

Gives data with same x small offsets to prevent bar stacking.

Parameters

data_list (*List of XYData*) – squeezed data

Returns

grouped data, bar width and x ticks

Return type

tuple

spinetoolbox.plotting._clear_plot(*plot_widget*)

Removes plots and legend from plot widget.

Parameters

plot_widget ([PlotWidget](#)) – plot widget

`spinetoolbox.plotting._limit_string_x_tick_labels(data, plot_widget)`

Limits the number of x tick labels in case x-axis consists of strings.

Matplotlib tries to plot every single x tick label if they are strings. This can become very slow if the labels are numerous.

Parameters

- **data** (*list of XYData*) – plot data
- **plot_widget** ([PlotWidget](#)) – plot widget

`spinetoolbox.plotting._table_display_row(row)`

Calculates a human-readable row number.

Parameters

row (*int*) – model row

Returns

row number

Return type

int

`spinetoolbox.plotting.plot_parameter_table_selection(model, model_indexes, table_header_sections, value_section_label, plot_widget=None)`

Returns a plot widget with plots of the selected indexes.

Parameters

- **model** (*QAbstractTableModel*) – a model
- **model_indexes** (*Iterable of QModelIndex*) – a list of QModelIndex objects for plotting
- **table_header_sections** (*list of ParameterTableHeaderSection*) – table header labels
- **value_section_label** (*str*) – value column's header label
- **plot_widget** ([PlotWidget](#), *optional*) – an existing plot widget to draw into or None to create a new widget

Returns

a PlotWidget object

Return type

[PlotWidget](#)

`spinetoolbox.plotting.plot_value_editor_table_selection(model, model_indexes, plot_widget=None)`

Returns a plot widget with plots of the selected indexes.

Parameters

- **model** (*QAbstractTableModel*) – a model
- **model_indexes** (*Iterable of QModelIndex*) – a list of QModelIndex objects for plotting
- **plot_widget** ([PlotWidget](#), *optional*) – an existing plot widget to draw into or None to create a new widget

Returns

a PlotWidget object

Return type

PlotWidget

`spinetoolbox.plotting.plot_pivot_table_selection(model, model_indexes, plot_widget=None)`

Returns a plot widget with plots of the selected indexes.

Parameters

- **model** (*QAbstractTableModel*) – a model
- **model_indexes** (*Iterable of QModelIndex*) – a list of QModelIndex objects for plotting
- **plot_widget** (*PlotWidget, optional*) – an existing plot widget to draw into or None to create a new widget

Returns

a PlotWidget object

Return type

PlotWidget

`spinetoolbox.plotting.plot_db_mgr_items(items, db_maps, plot_widget=None)`

Returns a plot widget with plots of database manager parameter value items.

Parameters

- **items** (*list of dict*) – parameter value items
- **db_maps** (*list of DatabaseMappingBase*) – database mappings corresponding to items
- **plot_widget** (*PlotWidget, optional*) – widget to add plots to

`spinetoolbox.plotting._has_x_column(model, source_model)`

Checks if pivot source model has x column.

Parameters

- **model** (*PivotTableSortFilterProxy*) – proxy pivot model
- **source_model** (*PivotTableModelBase*) – pivot table model

Returns

True if x pivot table has column, False otherwise

Return type

bool

`spinetoolbox.plotting._set_default_node(root_node, key, label)`

Gets node from the contents of root_node adding a new node if necessary.

Parameters

- **root_node** (*TreeNode*) – root node
- **key** (*Hashable*) – key to root_node contents
- **label** (*str*) – label of possible new node

Returns

node at given key

Return type

TreeNode

`spinetoolbox.plotting._get_parsed_value(model_index, display_row)`

Gets parsed value from model.

Parameters

- **model_index** (*QModelIndex*) – model index
- **display_row** (*Callable*) – callable that returns a display row

Returns

parsed value

Return type

Any

Raises

PlottingError – raised if parsing of value failed

`spinetoolbox.plotting._pivot_index_names(indexes)`

Gathers index names from pivot table.

Parameters

indexes (*tuple of str*) – “path” of indexes

Returns

names corresponding to given indexes

Return type

tuple of str

`spinetoolbox.plotting._pivot_display_row(row, source_model)`

Calculates display row for pivot table.

Parameters

- **row** (*int*) – row in source table model
- **source_model** (*QAbstractItemModel*) – pivot model

Returns

human-readable row number

Return type

int

`spinetoolbox.plotting._convert_to_leaf(y)`

Converts parameter value to leaf TreeElement.

Parameters

y (*Any*) – parameter value

Returns

leaf element

Return type

float or datetime or *TreeNode*

`spinetoolbox.plotting.add_row_to_exception(row, display_row)`

Adds row information to PlottingError if it is raised in the with block.

Parameters

- **row** (*int*) – row
- **display_row** (*Callable*) – function to convert row to display row

`spinetoolbox.plotting.add_array_plot(plot_widget, value)`

Adds an array plot to a plot widget.

Parameters

- **plot_widget** (*PlotWidget*) – a plot widget to modify
- **value** (*Array*) – the array to plot

`spinetoolbox.plotting.add_time_series_plot(plot_widget, value)`

Adds a time series step plot to a plot widget.

Parameters

- **plot_widget** (*PlotWidget*) – a plot widget to modify
- **value** (*TimeSeries*) – the time series to plot

`spinetoolbox.plugin_manager`

Contains PluginManager class.

Module Contents

Classes

<code>PluginManager</code>	Class for managing plugins.
<code>_PluginWorker</code>	

Functions

<code>_download_file(remote, local)</code>
<code>_download_plugin(plugin, plugin_local_dir)</code>

`spinetoolbox.plugin_manager._download_file(remote, local)`

`spinetoolbox.plugin_manager._download_plugin(plugin, plugin_local_dir)`

class `spinetoolbox.plugin_manager.PluginManager(toolbox)`

Class for managing plugins.

Parameters

toolbox (*ToolboxUI*) – Toolbox instance.

property `plugin_toolbars`

property `plugin_specs`

load_installed_plugins()

Loads installed plugins and adds their specifications to toolbars.

reload_plugins_with_local_data()

Reloads plugins that have project specific local data.

load_individual_plugin(*plugin_dir*, *specification_local_data*)

Loads plugin from directory.

Parameters

- **plugin_dir** (*str*) – path of plugin dir with “plugin.json” in it.
- **specification_local_data** (*dict*) – specification local data

_create_worker()

_clean_up_worker(*worker*)

_load_registry()

show_install_plugin_dialog(*_=False*)

_do_show_install_plugin_dialog()

_install_plugin(*plugin_name*)

Installs plugin from the registry and loads it.

Parameters

plugin_name (*str*) – plugin name

_load_installed_plugin(*plugin_local_dir*)

show_manage_plugins_dialog(*_=False*)

_do_show_manage_plugins_dialog()

_remove_plugin(*plugin_name*)

Removes installed plugin.

Parameters

plugin_name (*str*) – plugin name

_update_plugin(*plugin_name*)

exception `spinetoolbox.plugin_manager.PluginWorkFailed`

Bases: `Exception`

Exception to signal plugin worker that something failed.

Initialize self. See `help(type(self))` for accurate signature.

class `spinetoolbox.plugin_manager._PluginWorker`

Bases: `PySide6.QtCore.QObject`

failed

finished

succeeded

start(*function*, **args*, ***kwargs*)

_do_work()

clean_up()

spinetoolbox.project

Spine Toolbox project class.

Module Contents

Classes

<i>ItemNameStatus</i>	Generic enumeration.
<i>SpineToolboxProject</i>	Class for Spine Toolbox projects.

Functions

<i>node_successors</i> (<i>g</i>)	Returns a dict mapping nodes in topological order to a list of successors.
<i>_edges_causing_loops</i> (<i>g</i>)	Returns a list of edges whose removal from <i>g</i> results in it becoming acyclic.
<i>_ranks</i> (<i>node_successors</i>)	Calculates node ranks.

class spinetoolbox.project.**ItemNameStatus**

Bases: `enum.Enum`

Generic enumeration.

Derive from this class to define new enumerations.

OK

INVALID

EXISTS

SHORT_NAME_EXISTS

class spinetoolbox.project.**SpineToolboxProject**(*toolbox*, *p_dir*, *plugin_specs*, *app_settings*, *settings*, *logger*)

Bases: `spinetoolbox.metaobject.MetaObject`

Class for Spine Toolbox projects.

Parameters

- **toolbox** (`ToolboxUI`) – toolbox of this project
- **p_dir** (`str`) – Project directory

- **plugin_specs** (*Iterable of ProjectItemSpecification*) – specifications available as plugins
- **app_settings** (*QSettings*) – Toolbox settings
- **settings** (*ProjectSettings*) – project settings
- **logger** (*LoggerInterface*) – a logger instance

property all_item_names

property settings

property connections

property app_settings

project_about_to_be_torn_down

Emitted before project is being torn down.

project_execution_about_to_start

Emitted just before the entire project is executed.

project_execution_finished

Emitted after the entire project execution finishes.

connection_established

Emitted after new connection has been added to project.

connection_about_to_be_removed

Emitted before connection removal.

connection_updated

Emitted after a connection has been updated.

jump_added

Emitted after a jump has been added.

jump_about_to_be_removed

Emitted before a jump is removed.

jump_updated

Emitted after a jump has been replaced by another.

item_added

Emitted after a project item has been added.

item_about_to_be_removed

Emitted before project item removal.

item_renamed

Emitted after project item has been renamed.

specification_added

Emitted after a specification has been added.

specification_about_to_be_removed

Emitted before a specification will be removed.

specification_replaced

Emitted after a specification has been replaced.

specification_saved

Emitted after a specification has been saved.

LOCAL_EXECUTION_JOB_ID = '1'

toolbox()

Returns Toolbox main window.

Returns

main window

Return type

ToolboxUI

_create_project_structure(directory)

Makes the given directory a Spine Toolbox project directory. Creates directories and files that are common to all projects.

Parameters

directory (*str*) – Abs. path to a directory that should be made into a project directory

Returns

True if project structure was created successfully, False otherwise

Return type

bool

call_set_description(description)**set_description(description)**

Set object description.

Parameters

description (*str*) – Object description

save()

Collects project information and objects into a dictionary and writes it to a JSON file.

_save_all_specifications(local_path)

Writes all specifications except plugins to disk.

Local specification data is also written to disk, including local data from plugins.

Parameters

local_path (*Path*) –

Returns

specification local data that is supposed to be stored in a project specific place

Return type

dict

_pop_local_data_from_items_dict(items_dict)

Pops local data from project items dict.

Parameters

items_dict (*dict*) – items dict

Returns

local project item data

Return type

dict

static `_dump(target_dict, out_stream)`

Dumps given dict into output stream.

Parameters

- **target_dict** (*dict*) – dictionary to dump
- **out_stream** (*IOBase*) – output stream

load(*spec_factories, item_factories*)

Loads project from its project directory.

Parameters

- **spec_factories** (*dict*) – Dictionary mapping specification name to ProjectItemSpecificationFactory
- **item_factories** (*dict*) – mapping from item type to ProjectItemFactory

Returns

True if the operation was successful, False otherwise

Return type

bool

static `_merge_local_data_to_project_info(local_data_dict, project_info)`

Merges local data into project info.

Parameters

- **local_data_dict** (*dict*) – local data
- **project_info** (*dict*) – project dict

connection_from_dict(*connection_dict*)

jump_from_dict(*jump_dict*)

add_specification(*specification, save_to_disk=True*)

Adds a specification to the project.

Parameters

- **specification** (*ProjectItemSpecification*) – specification to add
- **save_to_disk** (*bool*) – if True, save the specification to disk

Returns

A unique identifier for the specification or None if the operation was unsuccessful

Return type

int

is_specification_name_reserved(*name*)

Checks if specification exists.

Parameters

name (*str*) – specification's name

Returns

True if project has given specification, False otherwise

Return type

bool

specifications()

Yields project's specifications.

Yields

ProjectItemSpecification – specification

_specification_id()

Creates an id for specification.

Returns

new id

Return type

int

get_specification(*name_or_id*)

Returns project item specification.

Parameters

name_or_id (*str* or *int*) – specification's name or id

Returns

specification or None if specification was not found

Return type

ProjectItemSpecification

specification_name_to_id(*name*)

Returns identifier for named specification.

Parameters

name (*str*) – specification's name

Returns

specification's id or None if no such specification exists

Return type

int

remove_specification(*id_or_name*)

Removes a specification from project.

Parameters

id_or_name (*int* or *str*) – specification's id or name

replace_specification(*name*, *specification*, *save_to_disk=True*)

Replaces an existing specification.

Refreshes the spec in all items that use it.

Parameters

- **name** (*str*) – name of the specification to replace
- **specification** (*ProjectItemSpecification*) – a specification
- **save_to_disk** (*bool*) – If True, saves the given specification to disk

Returns

True if operation was successful, False otherwise

Return type

bool

save_specification_file(*specification*, *previous_name=None*)

Saves the given project item specification.

Save path is determined by specification directory and specification's name.

Parameters

- **specification** (*ProjectItemSpecification*) – specification to save
- **previous_name** (*str*, *optional*) – specification's earlier name if it has been re-named/replaced

Returns

True if operation was successful, False otherwise

Return type

bool

_update_specification_local_data_store(*specification*, *local_data*, *previous_name*)

Updates the file containing local data of project's specifications.

Parameters

- **specification** (*ProjectItemSpecification*) – specification
- **local_data** (*dict*) – local data serialized into dict
- **previous_name** (*str*, *optional*) – specification's earlier name if it has been re-named/replaced

_default_specification_file_path(*specification*)

Determines a path inside project directory to save a specification.

Parameters

specification (*ProjectItemSpecification*) – specification

Returns

valid path or None if operation failed

Return type

str

add_item(*item*, *silent=True*)

Adds a project to item project.

Parameters

- **item** (*ProjectItem*) – item to add
- **silent** (*bool*) – if True, don't log messages

has_items()

Returns True if project has project items.

Returns

True if project has items, False otherwise

Return type

bool

get_item(*name*)

Returns project item.

Parameters

name (*str*) – item’s name

Returns

project item

Return type

ProjectItem

get_items()

Returns all project items.

Returns

all project items

Return type

list of *ProjectItem*

rename_item(*previous_name*, *new_name*, *rename_data_dir_message*)

Renames a project item

Parameters

- **previous_name** (*str*) – item’s current name
- **new_name** (*str*) – item’s new name
- **rename_data_dir_message** (*str*) – message to show when renaming item’s data directory

Returns

True if item was renamed successfully, False otherwise

Return type

bool

validate_project_item_name(*name*)

Validates item name.

Parameters

name (*str*) – proposed project item’s name

Returns

validation result

Return type

ItemNameStatus

find_connection(*source_name*, *destination_name*)

Searches for a connection between given items.

Parameters

- **source_name** (*str*) – source item’s name
- **destination_name** (*str*) – destination item’s name

Returns

connection instance or None if there is no connection

Return type

Connection

connections_for_item(*item_name*)

Returns connections that have given item as source or destination.

Parameters**item_name** (*str*) – item’s name**Returns**

connections connected to item

Return type

list of Connection

add_connection(**args*, *silent=False*)

Adds a connection to the project.

A single argument is expected to be the `LoggingConnection` instance. Optionally, source name and position, and destination name and position can be provided instead.

Parameters

- ***args** – connection to add
- **silent** (*bool*) – If False, prints ‘Link establ...’ msg to Event Log

Returns

True if connection was added successfully, False otherwise

Return type

bool

remove_connection(*connection*)

Removes a connection from the project.

Parameters**connection** (`LoggingConnection`) – connection to remove**update_connection**(*connection*, *source_position*, *destination_position*)

Updates existing connection between items.

Updating does not trigger any updates to the DAG or project items.

Parameters

- **connection** (`LoggingConnection`) – connection to update
- **source_position** (*str*) – link’s position on source item’s icon
- **destination_position** (*str*) – link’s position on destination item’s icon

jumps_for_item(*item_name*)

Returns jumps that have given item as source or destination.

Parameters**item_name** (*str*) – item’s name**Returns**

jumps connected to item

Return type

list of Jump

add_jump(*jump*, *silent=False*)

Adds a jump to project.

Parameters

- **jump** (*Jump*) – jump to add
- **silent** (*bool*) – if True, don't log messages

find_jump(*source_name*, *destination_name*)

Searches for a jump between given items.

Parameters

- **source_name** (*str*) – source item's name
- **destination_name** (*str*) – destination item's name

Returns

connection instance or None if there is no jump

Return type

Jump

remove_jump(*jump*)

Removes a jump from the project.

Parameters

jump (*Jump*) – jump to remove

update_jump(*jump*, *source_position*, *destination_position*)

Updates an existing jump between items.

Parameters

- **jump** (*LoggingJump*) – jump to update
- **source_position** (*str*) – link's position on source item's icon
- **destination_position** (*str*) – link's position on destination item's icon

_update_jump_icons()

Updates icons for all jumps in the project.

jump_issues(*jump*)

Checks if jump is OK.

Parameters

jump (*Jump*) – jump to check

Returns

list of issues, if any

Return type

list of str

_dag_iterator()

Iterates directed graphs in the project.

Yields

DiGraph

dag_with_node(*node*)

Returns the DiGraph that contains the given node (project item) name (str).

restore_project_items(*items_dict*, *item_factories*, *silent*)

Restores project items from dictionary.

Parameters

- **items_dict** (*dict*) – a mapping from item name to item dict
- **item_factories** (*dict*) – a mapping from item type to ProjectItemFactory
- **silent** (*bool*) – if True, suppress a log messages

remove_item_by_name(*item_name*, *delete_data=False*)

Removes project item by its name.

Parameters

- **item_name** (*str*) – Item’s name
- **delete_data** (*bool*) – If set to True, deletes the directories and data associated with the item

execute_dags(*dags*, *execution_permits_list*, *msg*)

Executes given dags.

Parameters

- **dags** (*list of DiGraph*) – List of DAGs
- **execution_permits_list** (*list of dict*) –
- **msg** (*str*) – Message to log before execution

_execute_dags(*dags*, *execution_permits_list*)

create_engine_worker(*dag*, *execution_permits*, *dag_identifier*, *settings*, *job_id*)

Creates and returns a SpineEngineWorker to execute given *validated* dag.

Parameters

- **dag** (*DiGraph*) – The dag
- **execution_permits** (*dict*) – mapping item names to a boolean indicating whether to execute it or skip it
- **dag_identifier** (*str*) – A string identifying the dag, for logging
- **settings** (*dict*) – project and app settings to send to the spine engine.
- **job_id** (*str*) – job id

Returns

SpineEngineWorker

_handle_engine_worker_finished(*worker*)

execute_selected(*names*)

Executes DAGs corresponding to given project items.

Parameters

names (*Iterable of str*) – Names of selected items

_split_to_subdags(*dag*, *selected_items*)

Checks if given dag contains weakly connected components. If it does, the weakly connected components are split into their own (sub)dags. If not, the dag is returned unaltered.

Parameters

- **dag** (*DiGraph*) – Dag that is checked if it needs to be split into subdags
- **selected_items** (*list*) – Names of selected items

Returns

List of dags, ready for execution

Return type

list of DiGraph

execute_project()

Executes all dags in the project.

_validate_dags(*dags*)

Validates dags and logs error messages.

Parameters

dags (*Iterable*) – dags to validate

Returns

validated dag

Return type

list

stop()

Stops execution.

notify_resource_changes_to_predecessors(*item*)

Updates resources for direct predecessors of given item.

Parameters

item (*ProjectItem*) – item whose resources have changed

_update_incoming_connection_and_jump_resources(*item_name*, *trigger_resources*)**notify_resource_changes_to_successors(*item*)**

Updates resources for direct successors and outgoing connections of given item.

Parameters

item (*ProjectItem*) – item whose resources have changed

_update_outgoing_connection_and_jump_resources(*item_name*, *trigger_resources*)**_notify_resource_changes(*trigger_name*, *target_names*, *provider_connections*, *update_resources*, *trigger_resources*)**

Updates resources in given direction for immediate neighbours of an item.

Parameters

- **trigger_name** (*str*) – item whose resources have changed
- **target_names** (*Iterable of str*) – items to be notified
- **provider_connections** (*Callable*) – function that receives a target item name and returns a list of Connections from resource providers
- **update_resources** (*Callable*) – function that takes an item name, a list of provider names, and a dictionary of resources, and does the updating
- **trigger_resources** (*list of ProjectItemResource*) – resources from the trigger item

notify_resource_replacement_to_successors(*item, old, new*)

Replaces resources for direct successors and outgoing connections of given item.

Parameters

- **item** (*ProjectItem*) – item whose resources have changed
- **old** (*list of ProjectItemResource*) – old resource
- **new** (*list of ProjectItemResource*) – new resource

notify_resource_replacement_to_predecessors(*item, old, new*)

Replaces resources for direct predecessors.

Parameters

- **item** (*ProjectItem*) – item whose resources have changed
- **old** (*list of ProjectItemResource*) – old resources
- **new** (*list of ProjectItemResource*) – new resources

_update_item_resources(*target_item, direction*)

Updates up or downstream resources for a single project item. Called in both directions after removing a Connection.

Parameters

- **target_item** (*ProjectItem*) – item whose resource need update
- **direction** (*ExecutionDirection*) – FORWARD updates resources from upstream, BACKWARD from downstream

predecessor_names(*name*)

Collects direct predecessor item names.

Parameters

name (*str*) – name of the project item whose predecessors to collect

Returns

direct predecessor names

Return type

set of str

successor_names(*name*)

Collects direct successor item names.

Parameters

name (*str*) – name of the project item whose successors to collect

Returns

direct successor names

Return type

set of str

descendant_names(*name*)

Yields descendant item names.

Parameters

name (*str*) – name of the project item whose descendants to collect

Yields

str – descendant name

outgoing_connections(*name*)

Collects outgoing connections.

Parameters

name (*str*) – name of the project item whose connections to collect

Returns

outgoing connections

Return type

set of Connection

_outgoing_jumps(*name*)

Collects outgoing jumps.

Parameters

name (*str*) – name of the project item whose jumps to collect

Returns

outgoing jumps

Return type

set of Jump

_outgoing_connections_and_jumps(*name*)

Collects outgoing connections and jumps.

Parameters

name (*str*) – name of the project item whose connections and jumps to collect

Returns

outgoing connections and jumps

Return type

set of Connection/Jump

incoming_connections(*name*)

Collects incoming connections.

Parameters

name (*str*) – name of the project item whose connections to collect

Returns

incoming connections

Return type

set of Connection

_incoming_jumps(*name*)

Collects incoming jumps.

Parameters

name (*str*) – name of the project item whose jumps to collect

Returns

incoming jumps

Return type

set of Jump

_incoming_connections_and_jumps(*name*)

Collects incoming connections and jumps.

Parameters

name (*str*) – name of the project item whose connections and jumps to collect

Returns

incoming connections

Return type

set of Connection/Jump

_update_successor(*successor, incoming_connections, resource_cache*)

_update_predecessor(*predecessor, outgoing_connections, resource_cache*)

_is_dag_valid(*dag*)

_update_ranks(*dag*)

prepare_remote_execution()

Pings the server and sends the project as a zip-file to server.

Returns

Job Id if server is ready for remote execution, empty string if something went wrong
or LOCAL_EXECUTION_JOB_ID if local execution is enabled.

Return type

str

finalize_remote_execution(*job_id*)

Sends a request to server to remove the project directory and removes the project ZIP file from client.y

Parameters

job_id (*str*) – job id

tear_down()

Cleans up project.

spinetoolbox.project.node_successors(*g*)

Returns a dict mapping nodes in topological order to a list of successors.

Parameters

g (*DiGraph*) –

Returns

dict

spinetoolbox.project._edges_causing_loops(*g*)

Returns a list of edges whose removal from g results in it becoming acyclic.

Parameters

g (*DiGraph*) –

Returns

list

spinetoolbox.project._ranks(*node_successors*)

Calculates node ranks.

Parameters

node_successors (*dict*) – a mapping from successor name to a list of predecessor names

Returns

a mapping from node name to rank

Return type

dict

spinetoolbox.project_commands

QUndoCommand subclasses for modifying the project.

Module Contents**Classes**

<i>SpineToolboxCommand</i>	
<i>SetItemSpecificationCommand</i>	Command to set the specification for a Tool.
<i>MoveIconCommand</i>	Command to move icons in the Design view.
<i>SetProjectDescriptionCommand</i>	Command to set the project description.
<i>AddProjectItemsCommand</i>	Command to add items.
<i>RemoveAllProjectItemsCommand</i>	Command to remove all items from project.
<i>RemoveProjectItemsCommand</i>	Command to remove items.
<i>RenameProjectItemCommand</i>	Command to rename project items.
<i>AddConnectionCommand</i>	Command to add connection between project items.
<i>RemoveConnectionsCommand</i>	Command to remove links.
<i>AddJumpCommand</i>	Command to add a jump between project items.
<i>RemoveJumpsCommand</i>	Command to remove jumps.
<i>SetJumpConditionCommand</i>	Command to set jump condition.
<i>UpdateJumpCmdLineArgsCommand</i>	Command to update Jump command line args.
<i>SetFiltersOnlineCommand</i>	Command to toggle filter value.
<i>SetConnectionDefaultFilterOnlineStatus</i>	Command to set connection's default filter online status.
<i>SetConnectionOptionsCommand</i>	Command to set connection options.
<i>AddSpecificationCommand</i>	Command to add item specification to a project.
<i>ReplaceSpecificationCommand</i>	Command to replace item specification in project.
<i>RemoveSpecificationCommand</i>	Command to remove specs from a project.
<i>SaveSpecificationAsCommand</i>	Command to remove item specs from a project.

class spinetoolbox.project_commands.SpineToolboxCommand

Bases: PySide6.QtGui.QUndoCommand

property is_critical

Returns True if this command needs to be undone before closing the project without saving changes.

successfully_undone = False

Flag to register the outcome of undoing a critical command, so toolbox can react afterwards.

class spinetoolbox.project_commands.SetItemSpecificationCommand(item, spec, old_spec)

Bases: *SpineToolboxCommand*

Command to set the specification for a Tool.

Parameters

- **item** ([ProjectItem](#)) – the Item
- **spec** ([ProjectItemSpecification](#)) – the new spec
- **old_spec** ([ProjectItemSpecification](#)) – the old spec

redo()

undo()

class `spinetoolbox.project_commands.MoveIconCommand`(*icon, project*)

Bases: [SpineToolboxCommand](#)

Command to move icons in the Design view.

Parameters

- **icon** ([ProjectItemIcon](#)) – the icon
- **project** ([SpineToolboxProject](#)) – project

redo()

undo()

_move_to(*positions*)

class `spinetoolbox.project_commands.SetProjectDescriptionCommand`(*project, description*)

Bases: [SpineToolboxCommand](#)

Command to set the project description.

Parameters

- **project** ([SpineToolboxProject](#)) – the project
- **description** (*str*) – The new description

redo()

undo()

class `spinetoolbox.project_commands.AddProjectItemsCommand`(*project, items_dict, item_factories, silent=True*)

Bases: [SpineToolboxCommand](#)

Command to add items.

Parameters

- **project** ([SpineToolboxProject](#)) – the project
- **items_dict** (*dict*) – a mapping from item name to item dict
- **item_factories** (*dict*) – a mapping from item type to ProjectItemFactory
- **silent** (*bool*) – If True, suppress messages

redo()

undo()


```
class spinetoolbox.project_commands.RemoveAllProjectItemsCommand(project, item_factories,  
                                                                delete_data=False)
```

Bases: *SpineToolboxCommand*

Command to remove all items from project.

Parameters

- **project** (*SpineToolboxProject*) – the project
- **item_factories** (*dict*) – a mapping from item type to ProjectItemFactory
- **delete_data** (*bool*) – If True, deletes the directories and data associated with the items

redo()

undo()

```
class spinetoolbox.project_commands.RemoveProjectItemsCommand(project, item_factories,  
                                                             item_names, delete_data=False)
```

Bases: *SpineToolboxCommand*

Command to remove items.

Parameters

- **project** (*SpineToolboxProject*) – The project
- **item_factories** (*dict*) – a mapping from item type to ProjectItemFactory
- **item_names** (*list of str*) – Item names
- **delete_data** (*bool*) – If True, deletes the directories and data associated with the item

redo()

undo()

```
class spinetoolbox.project_commands.RenameProjectItemCommand(project, previous_name, new_name)
```

Bases: *SpineToolboxCommand*

Command to rename project items.

Parameters

- **project** (*SpineToolboxProject*) – the project
- **previous_name** (*str*) – item's previous name
- **new_name** (*str*) – the new name

property is_critical

Returns True if this command needs to be undone before closing the project without saving changes.

redo()

undo()

```
class spinetoolbox.project_commands.AddConnectionCommand(project, source_name, source_position,  
                                                         destination_name, destination_position)
```

Bases: *SpineToolboxCommand*

Command to add connection between project items.

Parameters

- **project** (*SpineToolboxProject*) – project
- **source_name** (*str*) – source item’s name
- **source_position** (*str*) – link’s position on source item’s icon
- **destination_name** (*str*) – destination item’s name
- **destination_position** (*str*) – link’s position on destination item’s icon

redo()

undo()

class spinetoolbox.project_commands.**RemoveConnectionsCommand**(*project, connections*)

Bases: *SpineToolboxCommand*

Command to remove links.

Parameters

- **project** (*SpineToolboxProject*) – project
- **connections** (*list of LoggingConnection*) – the connections

redo()

undo()

class spinetoolbox.project_commands.**AddJumpCommand**(*project, source_name, source_position, destination_name, destination_position*)

Bases: *SpineToolboxCommand*

Command to add a jump between project items.

Parameters

- **project** (*SpineToolboxProject*) – project
- **source_name** (*str*) – source item’s name
- **source_position** (*str*) – link’s position on source item’s icon
- **destination_name** (*str*) – destination item’s name
- **destination_position** (*str*) – link’s position on destination item’s icon

redo()

undo()

class spinetoolbox.project_commands.**RemoveJumpsCommand**(*project, jumps*)

Bases: *SpineToolboxCommand*

Command to remove jumps.

Parameters

- **project** (*SpineToolboxProject*) – project
- **jumps** (*list of LoggingJump*) – the jumps

redo()

undo()

```
class spinetoolbox.project_commands.SetJumpConditionCommand(jump_properties, jump, condition)
```

Bases: [SpineToolboxCommand](#)

Command to set jump condition.

Parameters

- **jump_properties** ([JumpPropertiesWidget](#)) – jump’s properties tab
- **jump** (*Jump*) – target jump
- **condition** (*str*) – jump condition

redo()

undo()

```
class spinetoolbox.project_commands.UpdateJumpCmdLineArgsCommand(jump_properties, jump,
                                                                    cmd_line_args)
```

Bases: [SpineToolboxCommand](#)

Command to update Jump command line args.

Parameters

- **jump_properties** ([JumpPropertiesWidget](#)) – the item
- **cmd_line_args** (*list*) – list of command line args

redo()

undo()

```
class spinetoolbox.project_commands.SetFiltersOnlineCommand(project, connection, resource,
                                                             filter_type, online)
```

Bases: [SpineToolboxCommand](#)

Command to toggle filter value.

Parameters

- **project** ([SpineToolboxProject](#)) – project
- **connection** (*Connection*) – connection
- **resource** (*str*) – resource label
- **filter_type** (*str*) – filter type identifier
- **online** (*dict*) – mapping from scenario/tool id to online flag

redo()

undo()

```
class spinetoolbox.project_commands.SetConnectionDefaultFilterOnlineStatus(project,
                                                                              connection,
                                                                              default_status)
```

Bases: [SpineToolboxCommand](#)

Command to set connection’s default filter online status.

Parameters

- **project** ([SpineToolboxProject](#)) – project

- **connection** ([LoggingConnection](#)) – connection
- **default_status** (*bool*) – default filter online status

redo()

undo()

class `spinetoolbox.project_commands.SetConnectionOptionsCommand`(*project, connection, options*)

Bases: [SpineToolboxCommand](#)

Command to set connection options.

Parameters

- **project** ([SpineToolboxProject](#)) – project
- **connection** ([LoggingConnection](#)) – project
- **options** (*dict*) – containing options to be set

redo()

undo()

class `spinetoolbox.project_commands.AddSpecificationCommand`(*project, specification, save_to_disk*)

Bases: [SpineToolboxCommand](#)

Command to add item specification to a project.

Parameters

- **project** ([ToolboxUI](#)) – the toolbox
- **specification** (*ProjectItemSpecification*) – the spec
- **save_to_disk** (*bool*) – If True, save the specification to disk

redo()

undo()

class `spinetoolbox.project_commands.ReplaceSpecificationCommand`(*project, name, specification*)

Bases: [SpineToolboxCommand](#)

Command to replace item specification in project.

Parameters

- **project** ([ToolboxUI](#)) – the toolbox
- **name** (*str*) – the name of the spec to be replaced
- **specification** (*ProjectItemSpecification*) – the new spec

property is_critical

Returns True if this command needs to be undone before closing the project without saving changes.

redo()

undo()

class `spinetoolbox.project_commands.RemoveSpecificationCommand(project, name)`

Bases: *SpineToolboxCommand*

Command to remove specs from a project.

Parameters

- **project** (*SpineToolboxProject*) – the project
- **name** (*str*) – specification’s name

redo()

undo()

class `spinetoolbox.project_commands.SaveSpecificationAsCommand(project, name, path)`

Bases: *SpineToolboxCommand*

Command to remove item specs from a project.

Parameters

- **project** (*SpineToolboxProject*) – the project
- **name** (*str*) – specification’s name
- **path** (*str*) – new specification file location

redo()

undo()

spinetoolbox.project_item_icon

Classes for drawing graphics items on QGraphicsScene.

Module Contents

Classes

<i>ProjectItemIcon</i>	Base class for project item icons drawn in Design View.
<i>ConnectorButton</i>	Connector button graphics item. Used for Link drawing between project items.
<i>ExecutionIcon</i>	An icon to show information about the item's execution.
<i>ExclamationIcon</i>	An icon to notify that a ProjectItem is missing some configuration.
<i>RankIcon</i>	An icon to show the rank of a ProjectItem within its DAG.

class `spinetoolbox.project_item_icon.ProjectItemIcon(toolbox, icon_file, icon_color)`

Bases: `PySide6.QtWidgets.QGraphicsPathItem`

Base class for project item icons drawn in Design View.

Parameters

- **toolbox** (*ToolboxUI*) – QMainWindow instance

- **icon_file** (*str*) – Path to icon resource
- **icon_color** (*QColor*) – Icon’s color

ITEM_EXTENT = 64

FONT_SIZE_PIXELS = 12

rect()

_update_path()

update_path(*rounded*)

_do_update_path(*rounded*)

finalize(*name*, *x*, *y*)

Names the icon and moves it by given amount.

Parameters

- **name** (*str*) – icon’s name
- **x** (*int*) – horizontal offset
- **y** (*int*) – vertical offset

_setup()

Setup item’s attributes.

name()

Returns name of the item that is represented by this icon.

Returns

icon’s name

Return type

str

update_name_item(*new_name*)

Set a new text to name item.

Parameters

new_name (*str*) – icon’s name

set_name_attributes()

Set name QGraphicsSimpleTextItem attributes (font, size, position, etc.)

_reposition_name_item()

Set name item position (centered on top of the master icon).

conn_button(*position*='left')

Returns item’s connector button.

Parameters

position (*str*) – “left”, “right” or “bottom”

Returns

connector button

Return type

QWidget

outgoing_connection_links()

Collects outgoing connection links.

Returns

outgoing links

Return type

list of LinkBase

incoming_links()

Collects incoming connection links.

Returns

outgoing links

Return type

list of LinkBase

_closest_connector(*pos*)

Returns the closest connector button to given scene pos.

_update_link_drawer_destination(*pos=None*)

Updates link drawer destination. If pos is None, then the link drawer would have no destination. Otherwise, the destination would be the connector button closest to pos.

hoverEnterEvent(*event*)

Sets a drop shadow effect to icon when mouse enters its boundaries.

Parameters

event (*QGraphicsSceneMouseEvent*) – Event

hoverMoveEvent(*event*)**hoverLeaveEvent(*event*)**

Disables the drop shadow when mouse leaves icon boundaries.

Parameters

event (*QGraphicsSceneMouseEvent*) – Event

mousePressEvent(*event*)

Updates scene's icon group.

update_links_geometry()

Updates geometry of connected links to reflect this item's most recent position.

mouseReleaseEvent(*event*)

Clears pre-bump rects, and pushes a move icon command if necessary.

notify_item_move()**contextMenuEvent(*event*)**

Show item context menu.

Parameters

event (*QGraphicsSceneMouseEvent*) – Mouse event

itemChange(*change, value*)

Reacts to item removal and position changes.

In particular, destroys the drop shadow effect when the items is removed from a scene and keeps track of item's movements on the scene.

Parameters

- **change** (*GraphicsItemChange*) – a flag signalling the type of the change
- **value** – a value related to the change

Returns

Whatever `super()` does with the value parameter

set_pos_without_bumping(*pos*)

Sets position without bumping other items. Needed for undoing move operations.

Parameters

pos (*QPointF*) –

_handle_collisions()

Handles collisions with other items.

make_room_for_item(*other*)

Makes room for another item.

Parameters

item (*ProjectItemIcon*) –

_reestablish_bumped_items()

Moves bumped items back to their original position if no collision would happen anymore.

select_item()

Update GUI to show the details of the selected item.

paint(*painter, option, widget=None*)

Sets a dashed pen if selected.

class `spinetoolbox.project_item_icon.ConnectorButton`(*toolbox, parent, position='left'*)

Bases: `PySide6.QtWidgets.QGraphicsPathItem`

Connector button graphics item. Used for Link drawing between project items.

Parameters

- **toolbox** (*ToolboxUI*) – `QMainWindow` instance
- **parent** (*ProjectItemIcon*) – parent graphics item
- **position** (*str*) – Either “top”, “left”, “bottom”, or “right”

property `parent`

brush

hover_brush

rect()

update_path(*parent_radius*)

outgoing_links()

incoming_links()

parent_name()

Returns project item name owning this connector button.

project_item()

Returns the project item this connector button is attached to.

Returns

project item

Return type

ProjectItem

mousePressEvent(event)

Connector button mouse press event.

Parameters

event (*QGraphicsSceneMouseEvent*) – Event

_start_link(event)**set_friend_connectors_enabled(enabled)**

Enables or disables all connectors in the parent.

This is called by LinkDrawer to disable invalid connectors while drawing and reenabling them back when done.

Parameters

enabled (*bool*) – True to enable connectors, False to disable

set_hover_brush()**set_normal_brush()****hoverEnterEvent(event)**

Sets a darker shade to connector button when mouse enters its boundaries.

Parameters

event (*QGraphicsSceneMouseEvent*) – Event

hoverLeaveEvent(event)

Restore original brush when mouse leaves connector button boundaries.

Parameters

event (*QGraphicsSceneMouseEvent*) – Event

itemChange(change, value)

If this is being removed from the scene while it's the origin of the link drawer, put the latter to sleep.

class spinetoolbox.project_item_icon.ExecutionIcon(parent)

Bases: PySide6.QtWidgets.QGraphicsEllipseItem

An icon to show information about the item's execution.

Parameters

parent (*ProjectItemIcon*) – the parent item

_CHECK = '\uf00c'

_CROSS = '\uf00d'

_CLOCK = '\uf017'

_SKIP = '\uf054'

```
item_name()
_repaint(text, color)
mark_execution_waiting()
mark_execution_ignored()
mark_execution_started()
mark_execution_finished(item_finish_state)
hoverEnterEvent(event)
hoverLeaveEvent(event)
```

```
class spinetoolbox.project_item_icon.ExclamationIcon(parent)
```

Bases: PySide6.QtWidgets.QGraphicsTextItem

An icon to notify that a ProjectItem is missing some configuration.

Parameters

parent ([ProjectItemIcon](#)) – the parent item

```
FONT_SIZE_PIXELS = 14
```

```
clear_notifications()
```

Clear all notifications.

```
add_notification(text)
```

Add a notification.

```
remove_notification(subtext)
```

Remove the first notification that includes given subtext.

```
hoverEnterEvent(event)
```

Shows notifications as tool tip.

Parameters

event ([QGraphicsSceneMouseEvent](#)) – Event

```
hoverLeaveEvent(event)
```

Hides tool tip.

Parameters

event ([QGraphicsSceneMouseEvent](#)) – Event

```
class spinetoolbox.project_item_icon.RankIcon(parent)
```

Bases: PySide6.QtWidgets.QGraphicsTextItem

An icon to show the rank of a ProjectItem within its DAG.

Parameters

parent ([ProjectItemIcon](#)) – the parent item

```
_make_path(radius)
```

```
update_path(radius)
```

```
set_rank(rank)
```

spinetoolbox.project_settings

Contains project-specific settings.

Module Contents

Classes

<i>ProjectSettings</i>	Spine Toolbox project settings.
------------------------	---------------------------------

class spinetoolbox.project_settings.**ProjectSettings**

Spine Toolbox project settings.

enable_execute_all: `bool = True`

to_dict()

Serializes the settings into a dictionary.

Returns

serialized settings

Return type

dict

static from_dict(settings_dict)

Deserializes settings from dictionary.

Parameters

settings_dict (*dict*) – serialized settings

Returns

deserialized settings

Return type

ProjectSettings

spinetoolbox.project_upgrader

Contains ProjectUpgrader class used in upgrading and converting projects and project dicts from earlier versions to the latest version.

Module Contents

Classes

<i>ProjectUpgrader</i>	Class to upgrade/convert projects from earlier versions to the current version.
------------------------	---

Functions

<code>_fix_1d_array_to_array(mappings)</code>	Replaces '1d array' with 'array' for parameter type in Importer mappings.
---	---

class `spinetoolbox.project_upgrader.ProjectUpgrader(toolbox)`

Class to upgrade/convert projects from earlier versions to the current version.

Parameters

toolbox (`ToolboxUI`) – App main window instance

upgrade(*project_dict*, *project_dir*)

Upgrades the project described in given project dictionary to the latest version.

Parameters

- **project_dict** (*dict*) – Project configuration dictionary
- **project_dir** (*str*) – Path to current project directory

Returns

Latest version of the project info dictionary

Return type

dict

upgrade_to_latest(*v*, *project_dict*, *project_dir*)

Upgrades the given project dictionary to the latest version.

Parameters

- **v** (*int*) – Current version of the project dictionary
- **project_dict** (*dict*) – Project dictionary (JSON) to be upgraded
- **project_dir** (*str*) – Path to current project directory

Returns

Upgraded project dictionary

Return type

dict

static upgrade_v1_to_v2(*old*, *factories*)

Upgrades version 1 project dictionary to version 2.

Changes:

objects -> items, tool_specifications -> specifications store project item dicts under [“items”][<project item name>] instead of using their categories as keys specifications must be a dict instead of a list Add specifications[“Tool”] that must be a dict Remove “short name” from all project items

Parameters

- **old** (*dict*) – Version 1 project dictionary
- **factories** (*dict*) – Mapping of item type to item factory

Returns

Version 2 project dictionary

Return type

dict

upgrade_v2_to_v3(*old, project_dir, factories*)

Upgrades version 2 project dictionary to version 3.

Changes:

1. Move “specifications” from “project” -> “Tool” to just “project”
2. The “mappings” from importer items are used to build Importer specifications

Parameters

- **old** (*dict*) – Version 2 project dictionary
- **project_dir** (*str*) – Path to current project directory
- **factories** (*dict*) – Mapping of item type to item factory

Returns

Version 3 project dictionary

Return type

dict

static upgrade_v3_to_v4(*old*)

Upgrades version 3 project dictionary to version 4.

Changes:

1. Rename “Exporter” item type to “GdxExporter”

Parameters

old (*dict*) – Version 3 project dictionary

Returns

Version 4 project dictionary

Return type

dict

static upgrade_v4_to_v5(*old*)

Upgrades version 4 project dictionary to version 5.

Changes:

1. Get rid of “Combiner” items.

Parameters

old (*dict*) – Version 4 project dictionary

Returns

Version 5 project dictionary

Return type

dict

static upgrade_v5_to_v6(*old, project_dir*)

Upgrades version 5 project dictionary to version 6.

Changes:

1. Data store URL labels do not have ‘{’ and ‘}’ anymore

2. Importer stores resource labels instead of serialized paths in “file_selection”.
3. Gimlet’s “selections” is now called “file_selection”
4. Gimlet stores resource labels instead of serialized paths in “file_selection”.
5. Gimlet and Tool store command line arguments as serialized CmdLineArg objects, not serialized paths

Parameters

- **old** (*dict*) – Version 5 project dictionary
- **project_dir** (*str*) – Path to current project directory

Returns

Version 6 project dictionary

Return type

dict

static upgrade_v6_to_v7 (*old*)

Upgrades version 6 project dictionary to version 7.

Changes:

1. Introduces Mergers in between DS -> DS links.

Parameters

old (*dict*) – Version 6 project dictionary

Returns

Version 7 project dictionary

Return type

dict

static upgrade_v7_to_v8 (*old*)

Upgrades version 7 project dictionary to version 8.

Changes:

1. Move purge settings from items to their outgoing connections.

Parameters

old (*dict*) – Version 7 project dictionary

Returns

Version 8 project dictionary

Return type

dict

static upgrade_v8_to_v9 (*old*)

Upgrades version 8 project dictionary to version 9.

Changes:

1. Remove [“project”][“name”] key

Parameters

old (*dict*) – Version 8 project dictionary

Returns

Version 9 project dictionary

Return type

dict

static upgrade_v9_to_v10(*old*)

Upgrades version 9 project dictionary to version 10.

Changes:

1. Remove connections from Gimlets and GDXExporters
2. Remove Gimlet items

Parameters

old (*dict*) – Version 9 project dictionary

Returns

Version 10 project dictionary

Return type

dict

static upgrade_v10_to_v11(*old*)

Upgrades version 10 project dictionary to version 11.

Changes:

1. Add [“project”][“settings”] key

Parameters

old (*dict*) – Version 10 project dictionary

Returns

Version 11 project dictionary

Return type

dict

static make_unique_importer_specification_name(*importer_name*, *label*, *k*)**get_project_directory()**

Asks the user to select a new project directory. If the selected directory is already a Spine Toolbox project directory, asks if overwrite is ok. Used when opening a project from an old style project file (.proj).

Returns

Path to project directory or an empty string if operation is canceled.

Return type

str

is_valid(*v*, *p*)

Checks given project dict if it is valid for given version.

Parameters

- **v** (*int*) – project version to validate against
- **p** (*dict*) – project dictionary

Returns

True if project is valid, False otherwise

Return type

bool

is_valid_v1(*p*)

Checks that the given project JSON dictionary contains a valid version 1 Spine Toolbox project. Valid meaning, that it contains all required keys and values are of the correct type.

Parameters

p (*dict*) – Project information JSON

Returns

True if project is a valid version 1 project, False if it is not

Return type

bool

is_valid_v2_to_v8(*p*, *v*)

Checks that the given project JSON dictionary contains a valid version 2 to 8 Spine Toolbox project. Valid meaning, that it contains all required keys and values are of the correct type.

Parameters

- **p** (*dict*) – Project information JSON
- **v** (*int*) – Version

Returns

True if project is a valid version 2 to version 8 project, False if it is not

Return type

bool

is_valid_v9_to_v10(*p*)

Checks that the given project JSON dictionary contains a valid version 9 or 10 Spine Toolbox project. Valid meaning, that it contains all required keys and values are of the correct type.

Parameters

p (*dict*) – Project information JSON

Returns

True if project is a valid version 9 and 10 project, False otherwise

Return type

bool

is_valid_v11(*p*)

Checks that the given project JSON dictionary contains a valid version 11 Spine Toolbox project. Valid meaning, that it contains all required keys and values are of the correct type.

Parameters

p (*dict*) – Project information JSON

Returns

True if project is a valid version 11 project, False otherwise

Return type

bool

backup_project_file(*project_dir*, *v*)

Makes a backup copy of project.json file.

force_save(*p*, *project_dir*)

Saves given project dictionary to project.json file. Used to force save project.json file when the project dictionary has been upgraded.

spinetoolbox.project_upgrader._fix_1d_array_to_array(*mappings*)

Replaces '1d array' with 'array' for parameter type in Importer mappings.

With spinedb_api >= 0.3, '1d array' parameter type was replaced by 'array'. Other settings in a mapping are backwards compatible except the name.

spinetoolbox.qthread_pool_executor

Qt-based thread pool executor.

Module Contents

Classes

<i>QtBasedQueue</i>	A Qt-based clone of queue.Queue.
<i>QtBasedFuture</i>	A Qt-based clone of concurrent.futures.Future.
<i>QtBasedThread</i>	A Qt-based clone of threading.Thread.
<i>QtBasedThreadPoolExecutor</i>	A Qt-based clone of concurrent.futures.ThreadPoolExecutor

exception spinetoolbox.qthread_pool_executor.**TimeoutError**

Bases: Exception

An exception to raise when a timeouts expire

Initialize self. See help(type(self)) for accurate signature.

class spinetoolbox.qthread_pool_executor.**QtBasedQueue**

A Qt-based clone of queue.Queue.

put(*item*)

get(*timeout=None*)

class spinetoolbox.qthread_pool_executor.**QtBasedFuture**

A Qt-based clone of concurrent.futures.Future.

set_result(*result*)

set_exception(*exc*)

result(*timeout=None*)

exception(*timeout=None*)

```
class spinetoolbox.qthread_pool_executor.QtBasedThread(target=None, args=())
```

Bases: PySide6.QtCore.QThread

A Qt-based clone of threading.Thread.

```
run()
```

```
class spinetoolbox.qthread_pool_executor.QtBasedThreadPoolExecutor(max_workers=None)
```

A Qt-based clone of concurrent.futures.ThreadPoolExecutor

```
submit(fn, *args, **kwargs)
```

```
_spawn_thread()
```

```
_do_work()
```

```
shutdown()
```

spinetoolbox.spine_db_commands

QUndoCommand subclasses for modifying the db.

Module Contents

Classes

AgedUndoStack

AgedUndoCommand

param parent

The parent command, used for defining macros.

SpineDBMacro

A command that just runs a series of SpineDBCommand's one after another, *waiting* for each one to finish

SpineDBCommand

Base class for all commands that modify a Spine DB.

AddItemsCommand

Base class for all commands that modify a Spine DB.

UpdateItemsCommand

Base class for all commands that modify a Spine DB.

RemoveItemsCommand

Base class for all commands that modify a Spine DB.

```
class spinetoolbox.spine_db_commands.AgedUndoStack(*args, **kwargs)
```

Bases: PySide6.QtGui.QUndoStack

```
property redo_age
```

```
property undo_age
```

```
commands()
```

```
push(cmd)
```

```
commit()
```

setClean()

class spinetoolbox.spine_db_commands.**AgedUndoCommand**(*parent=None*)

Bases: PySide6.QtGui.QUndoCommand

Parameters

parent (*QUndoCommand*, *optional*) – The parent command, used for defining macros.

property age

clone()

Clones the command.

Returns

cloned command

Return type

AgedUndoCommand

abstract **_do_clone()**

Clones the command.

Subclasses should reimplement this to clone their internal state.

Returns

cloned command

Return type

AgedUndoCommand

is_clone(*other*)

redo()

undo()

class spinetoolbox.spine_db_commands.**SpineDBMacro**(*cmd_iter*, *parent=None*)

Bases: *AgedUndoCommand*

A command that just runs a series of SpineDBCommand's one after another, *waiting* for each one to finish before starting the next.

Parameters

parent (*QUndoCommand*, *optional*) – The parent command, used for defining macros.

_do_clone()

Clones the command.

Subclasses should reimplement this to clone their internal state.

Returns

cloned command

Return type

AgedUndoCommand

redo()

_redo_next()

undo()

`_undo_next()`

class `spinetoolbox.spine_db_commands.SpineDBCommand`(*db_mgr*, *db_map*, *parent=None*)

Bases: [*AgedUndoCommand*](#)

Base class for all commands that modify a Spine DB.

Parameters

- **db_mgr** ([*SpineDBManager*](#)) – SpineDBManager instance
- **db_map** (*DiffDatabaseMapping*) – DiffDatabaseMapping instance
- **parent** (*QUndoCommand*, *optional*) – The parent command, used for defining macros.

handle_undo_complete(*db_map_data*)

Calls the undo complete callback with the data from `undo()`.

Subclasses need to pass this as the callback to the function that modifies the db in `undo()`.

Parameters

db_map_data (*dict*) – mapping from database map to list of original cache items

handle_redo_complete(*db_map_data*)

Calls the redo complete callback with the data from `redo()`.

Subclasses need to pass this as the callback to the function that modifies the db in `redo()`.

Parameters

db_map_data (*dict*) – mapping from database map to list of cache items

abstract _handle_first_redo_complete(*db_map_data*)

Reimplement in subclasses to do stuff with the data from running `redo()` the first time.

Parameters

db_map_data (*dict*) – mapping from database map to list of original cache items

class `spinetoolbox.spine_db_commands.AddItemsCommand`(*db_mgr*, *db_map*, *data*, *item_type*,
parent=None, *check=True*)

Bases: [*SpineDBCommand*](#)

Base class for all commands that modify a Spine DB.

Parameters

- **db_mgr** ([*SpineDBManager*](#)) – SpineDBManager instance
- **db_map** (*DiffDatabaseMapping*) – DiffDatabaseMapping instance
- **data** (*list*) – list of dict-items to add
- **item_type** (*str*) – the item type
- **parent** (*QUndoCommand*, *optional*) – The parent command, used for defining macros.

_add_command_name

_do_clone()

Clones the command.

Subclasses should reimplement this to clone their internal state.

Returns

cloned command

Return type*AgedUndoCommand***redo()****undo()****_handle_first_redo_complete**(*db_map_data*)

Reimplement in subclasses to do stuff with the data from running redo() the first time.

Parameters**db_map_data** (*dict*) – mapping from database map to list of original cache items

```
class spinetoolbox.spine_db_commands.UpdateItemsCommand(db_mgr, db_map, data, item_type,
                                                         parent=None, check=True)
```

Bases: *SpineDBCommand*

Base class for all commands that modify a Spine DB.

Parameters

- **db_mgr** (*SpineDBManager*) – SpineDBManager instance
- **db_map** (*DiffDatabaseMapping*) – DiffDatabaseMapping instance
- **data** (*list*) – list of dict-items to update
- **item_type** (*str*) – the item type
- **parent** (*QUndoCommand*, *optional*) – The parent command, used for defining macros.

_update_command_name**_do_clone()**

Clones the command.

Subclasses should reimplement this to clone their internal state.

Returns

cloned command

Return type*AgedUndoCommand***redo()****undo()****_handle_first_redo_complete**(*db_map_data*)

Reimplement in subclasses to do stuff with the data from running redo() the first time.

Parameters**db_map_data** (*dict*) – mapping from database map to list of original cache items

```
class spinetoolbox.spine_db_commands.RemoveItemsCommand(db_mgr, db_map, ids, item_type,
                                                         parent=None)
```

Bases: *SpineDBCommand*

Base class for all commands that modify a Spine DB.

Parameters

- **db_mgr** (*SpineDBManager*) – SpineDBManager instance

- **db_map** (*DiffDatabaseMapping*) – DiffDatabaseMapping instance
- **ids** (*set*) – set of ids to remove
- **item_type** (*str*) – the item type
- **parent** (*QUndoCommand*, *optional*) – The parent command, used for defining macros.

_do_clone()

Clones the command.

Subclasses should reimplement this to clone their internal state.

Returns

cloned command

Return type

AgedUndoCommand

redo()

undo()

_handle_first_redo_complete(db_map_data)

Reimplement in subclasses to do stuff with the data from running redo() the first time.

Parameters

db_map_data (*dict*) – mapping from database map to list of original cache items

_update_undo_data(db_map_data)

spinetoolbox.spine_db_icon_manager

Provides SpineDBIconManager.

Module Contents

Classes

<i>_SceneSvgRenderer</i>	
<i>SpineDBIconManager</i>	A class to manage object_class icons for spine db editors.
<i>SceneIconEngine</i>	Specialization of QIconEngine used to draw scene-based icons.

Functions

`_align_text_in_item(item)`

`_center_scene(scene)`

`spinetoolbox.spine_db_icon_manager._align_text_in_item(item)`

`spinetoolbox.spine_db_icon_manager._center_scene(scene)`

class `spinetoolbox.spine_db_icon_manager._SceneSvgRenderer`

Bases: `PySide6.QtSvg.QSvgRenderer`

scene

classmethod `from_scene(scene)`

class `spinetoolbox.spine_db_icon_manager.SpineDBIconManager`

A class to manage object_class icons for spine db editors.

update_icon_caches(classes)

Called after adding or updating entity classes. Stores `display_icons` and clears obsolete entries from the relationship class and entity group renderer caches.

_create_icon_renderer(icon_code, color_code)

icon_renderer(icon_code, color_code)

_create_class_renderer(class_name)

class_renderer(class_name)

_create_rel_cls_renderer(object_class_names)

relationship_class_renderer(rel_cls_name, object_class_name_list)

_create_group_renderer(class_name)

group_renderer(class_name)

static icon_from_renderer(renderer)

class `spinetoolbox.spine_db_icon_manager.SceneIconEngine(scene)`

Bases: `spinetoolbox.helpers.TransparentIconEngine`

Specialization of `QIconEngine` used to draw scene-based icons.

paint(painter, rect, mode=None, state=None)

spinetoolbox.spine_db_manager

The SpineDBManager class

Module Contents

Classes

<i>SpineDBManager</i>	Class to manage DBs within a project.
-----------------------	---------------------------------------

Functions

<i>do_create_new_spine_database(url)</i>	Creates a new spine database at the given url.
--	--

`spinetoolbox.spine_db_manager.do_create_new_spine_database(url)`

Creates a new spine database at the given url.

class `spinetoolbox.spine_db_manager.SpineDBManager(settings, parent)`

Bases: `PySide6.QtCore.QObject`

Class to manage DBs within a project.

Initializes the instance.

Parameters

- **settings** (*QSettings*) – Toolbox settings
- **parent** (*QObject*, *optional*) – parent object

property `db_maps`

property `db_urls`

error_msg

session_refreshed

session_committed

session_rolled_back

items_added

Emitted whenever items are added to a DB.

Parameters

- **str** – item type, such as “object_class”
- **dict** – mapping `DiffDatabaseMapping` to list of added dict-items.

items_updated

Emitted whenever items are updated in a DB.

Parameters

- **str** – item type, such as “object_class”

- **dict** – mapping DiffDatabaseMapping to list of updated dict-items.

items_removed

Emitted whenever items are removed from a DB.

Parameters

- **str** – item type, such as “object_class”
- **dict** – mapping DiffDatabaseMapping to list of updated dict-items.

waiting_for_fetcher**fetcher_waiting_over****_connect_signals()****receive_error_msg**(*db_map_error_log*)**receive_session_refreshed**(*db_maps*)**receive_session_committed**(*db_maps, cookie*)**receive_session_rolled_back**(*db_maps*)**_get_worker**(*db_map*)

Returns a worker.

Parameters

db_map (*DiffDatabaseMapping*) –

Returns

SpineDBWorker

can_fetch_more(*db_map, parent*)

Whether or not we can fetch more items of given type from given db.

Parameters

- **db_map** (*DiffDatabaseMapping*) –
- **parent** ([FetchParent](#)) – The object that requests the fetching and that might want to react to further DB modifications.

Returns

bool

fetch_more(*db_map, parent*)

Fetches more items of given type from given db.

Parameters

- **db_map** (*DiffDatabaseMapping*) –
- **parent** ([FetchParent](#)) – The object that requests the fetching.

add_items_to_cache(*item_type, db_map_data*)

Adds items to cache.

Parameters

- **item_type** (*str*) –
- **db_map_data** (*dict*) – lists of dictionary items keyed by DiffDatabaseMapping

Returns

mapping db_map to added cache items

Return type

dict

update_items_in_cache(*item_type*, *db_map_data*)

Updates items in cache.

Parameters

- **item_type** (*str*) –
- **db_map_data** (*dict*) – lists of dictionary items keyed by DiffDatabaseMapping

static remove_items_in_cache(*item_type*, *db_map_ids*)

Removes items in cache.

Parameters

- **item_type** (*str*) –
- **db_map_ids** (*dict*) – mapping db_map to ids to remove

Returns

mapping db_map to removed cache items

Return type

dict

get_db_map_cache(*db_map*, *fetch_item_types=None*, *include_descendants=False*,
include_ancestors=False)

get_icon_mgr(*db_map*)

Returns an icon manager for given db_map.

Parameters

db_map (*DiffDatabaseMapping*) –

Returns

SpineDBIconManager

update_icons(*db_map_data*)

Runs when object classes are added or updated. Setups icons for those classes.

Parameters

db_map_data (*dict*) – lists of dictionary items keyed by DiffDatabaseMapping

static db_map_key(*db_map*)

Creates an identifier for given db_map.

Parameters

db_map (*DiffDatabaseMapping*) – database mapping

Returns

identification key

Return type

int

db_map_from_key(*key*)

Returns database mapping that corresponds to given identification key.

Parameters

key (*int*) – identification key

Returns

database mapping

Return type

DiffDatabaseMapping

Raises

KeyError – raised if database map is not found

db_map(*url*)

Returns a database mapping for given URL.

Parameters

url (*str*) – a database URL

Returns

a database map or None if not found

Return type

DiffDatabaseMapping

create_new_spine_database(*url*, *logger*)**close_session(*url*)**

Pops any db map on the given url and closes its connection.

Parameters

url (*str*) –

close_all_sessions()

Closes connections to all database mappings.

get_db_map(*url*, *logger*, *codename=None*, *upgrade=False*, *create=False*, *ignore_version_error=False*)

Returns a DiffDatabaseMapping instance from url if possible, None otherwise. If needed, asks the user to upgrade to the latest db version.

Parameters

- **url** (*str*, *URL*) –
- **logger** (*LoggerInterface*) –
- **codename** (*str*, *NoneType*, *optional*) –
- **upgrade** (*bool*, *optional*) –
- **create** (*bool*, *optional*) –

Returns

DiffDatabaseMapping, NoneType

_do_get_db_map(*url*, *codename*, *upgrade*, *create*)

Returns a memorized DiffDatabaseMapping instance from url. Called by *get_db_map*.

Parameters

- **url** (*str*, *URL*) –
- **codename** (*str*, *NoneType*) –
- **upgrade** (*bool*) –

- **create** (*bool*) –

Returns

DiffDatabaseMapping

query(*db_map*, *sq_name*)

For tests.

add_db_map_listener(*db_map*, *listener*)

Adds listener for given db_map.

remove_db_map_listener(*db_map*, *listener*)

Removes db_map from the maps listener listens to.

db_map_listeners(*db_map*)

register_listener(*listener*, **db_maps*)

Register given listener for all given db_map's signals.

Parameters

- **listener** (*object*) –
- **db_maps** (*DiffDatabaseMapping*) –

unregister_listener(*listener*, **db_maps*, *dirty_db_maps=None*, *commit_dirty=False*, *commit_msg=""*)

Unregisters given listener from given db_map signals. If any of the db_maps becomes an orphan and is dirty, prompts user to commit or rollback.

Parameters

- **listener** (*object*) –
- ***db_maps** (*DiffDatabaseMapping*) –
- **commit_dirty** (*bool*) – True to commit dirty database mapping, False to roll back
- **commit_msg** (*str*) – commit message

Returns

All the db maps that failed to commit

Return type

failed_db_maps (list)

is_dirty(*db_map*)

Returns True if mapping has pending changes.

Parameters

db_map (*DiffDatabaseMapping*) – database mapping

Returns

True if db_map has pending changes, False otherwise

Return type

bool

dirty(**db_maps*)

Filters clean mappings from given database maps.

Parameters

***db_maps** – mappings to check

Returns

dirty mappings

Return type

list of DiffDatabaseMapping

dirty_and_without_editors(*listener*, **db_maps*)

Checks which of the given database mappings are dirty and have no editors.

Parameters

- **listener** (*Any*) – a listener object
- ***db_maps** – mappings to check

Returns

mappings that are dirty and don't have editors

Return type

list of DiffDatabaseMapping

clean_up()

refresh_session(**db_maps*)

commit_session(*commit_msg*, **dirty_db_maps*, *cookie=None*)

Commits the current session.

Parameters

- **commit_msg** (*str*) – commit message for all database maps
- ***dirty_db_maps** – dirty database maps to commit
- **cookie** (*object*, *optional*) – a free form identifier which will be forwarded to `session_committed` signal

Returns

list of the db maps that failed to commit

Return type

failed_db_maps (list)

notify_session_committed(*cookie*, **db_maps*)

Notifies manager and listeners when a commit has taken place by a third party.

Parameters

- **cookie** (*Any*) – commit cookie
- ***db_maps** – database maps that were committed

rollback_session(**dirty_db_maps*)

Rolls back the current session.

Parameters

***dirty_db_maps** – dirty database maps to commit

entity_class_renderer(*db_map*, *entity_type*, *entity_class_id*, *for_group=False*)

Returns an icon renderer for a given entity class.

Parameters

- **db_map** (*DiffDatabaseMapping*) – database map

- **entity_type** (*str*) – either ‘object_class’ or ‘relationship_class’
- **entity_class_id** (*int*) – entity class’ id
- **for_group** (*bool*) – if True, return the group object icon instead

Returns

requested renderer or None if no entity class was found

Return type

QSvgRenderer

entity_class_icon(*db_map*, *entity_type*, *entity_class_id*, *for_group=False*)

Returns an appropriate icon for a given entity class.

Parameters

- **db_map** (*DiffDatabaseMapping*) – database map
- **entity_type** (*str*) – either ‘object_class’ or ‘relationship_class’
- **entity_class_id** (*int*) – entity class’ id
- **for_group** (*bool*) – if True, return the group object icon instead

Returns

requested icon or None if no entity class was found

Return type

QIcon

get_item(*db_map*, *item_type*, *id_*, *only_visible=True*)

Returns the item of the given type in the given db map that has the given id, or an empty dict if not found.

Parameters

- **db_map** (*DiffDatabaseMapping*) –
- **item_type** (*str*) –
- **id** (*int*) –
- **only_visible** (*bool*, *optional*) – If True, only looks in items that have already made it into the cache.

Returns

cached item

Return type

CacheItem

get_field(*db_map*, *item_type*, *id_*, *field*, *only_visible=True*)

get_items(*db_map*, *item_type*, *only_visible=True*)

Returns a list of the items of the given type in the given db map.

Parameters

- **db_map** (*DiffDatabaseMapping*) –
- **item_type** (*str*) –
- **only_visible** (*bool*, *optional*) – If True, only returns items that have already made it into the cache.

Returns

list

get_items_by_field(*db_map*, *item_type*, *field*, *value*, *only_visible=True*)

Returns a list of items of the given type in the given db map that have the given value for the given field.

Parameters

- **db_map** (*DiffDatabaseMapping*) –
- **item_type** (*str*) –
- **field** (*str*) –
- **value** –

Returns

list

get_item_by_field(*db_map*, *item_type*, *field*, *value*, *only_visible=True*)

Returns the first item of the given type in the given db map that has the given value for the given field
Returns an empty dictionary if none found.

Parameters

- **db_map** (*DiffDatabaseMapping*) –
- **item_type** (*str*) –
- **field** (*str*) –
- **value** –

Returns

dict

static display_data_from_parsed(*parsed_data*)

Returns the value's database representation formatted for Qt.ItemDataRole.DisplayRole.

static tool_tip_data_from_parsed(*parsed_data*)

Returns the value's database representation formatted for Qt.ItemDataRole.ToolTipRole.

_format_list_value(*db_map*, *item_type*, *value*, *list_value_id*)

get_value(*db_map*, *item_type*, *id_*, *role=Qt.ItemDataRole.DisplayRole*)

Returns the value or default value of a parameter.

Parameters

- **db_map** (*DiffDatabaseMapping*) –
- **item_type** (*str*) – either “parameter_definition”, “parameter_value”, or “list_value”
- **id** (*int*) – The parameter_value or definition id
- **role** (*int*, *optional*) –

Returns

any

get_value_from_data(*data*, *role=Qt.ItemDataRole.DisplayRole*)

Returns the value or default value of a parameter directly from data. Used by EmptyParameterModel.
data().

Parameters

- **data** (*str*) – joined value and type
- **role** (*int*, *optional*) –

Returns

any

static `_parse_value(db_value, value_type=None)`

`_format_value(parsed_value, role=Qt.ItemDataRole.DisplayRole)`

Formats the given value for the given role.

Parameters

- **parsed_value** (*object*) – A python object as returned by `spinedb_api.from_database`
- **role** (*int*, *optional*) –

get_value_indexes(*db_map*, *item_type*, *id_*)

Returns the value or default value indexes of a parameter.

Parameters

- **db_map** (*DiffDatabaseMapping*) –
- **item_type** (*str*) – either “parameter_definition” or “parameter_value”
- **id** (*int*) – The parameter_value or definition id

get_value_index(*db_map*, *item_type*, *id_*, *index*, *role=Qt.ItemDataRole.DisplayRole*)

Returns the value or default value of a parameter for a given index.

Parameters

- **db_map** (*DiffDatabaseMapping*) –
- **item_type** (*str*) – either “parameter_definition” or “parameter_value”
- **id** (*int*) – The parameter_value or definition id
- **index** – The index to retrieve
- **role** (*int*, *optional*) –

get_value_list_item(*db_map*, *id_*, *index*, *role=Qt.ItemDataRole.DisplayRole*, *only_visible=True*)

Returns one value item of a parameter_value_list.

Parameters

- **db_map** (*DiffDatabaseMapping*) –
- **id** (*int*) – The parameter_value_list id
- **index** (*int*) – The value item index
- **role** (*int*, *optional*) –

get_parameter_value_list(*db_map*, *id_*, *role=Qt.ItemDataRole.DisplayRole*, *only_visible=True*)

Returns a parameter_value_list formatted for the given role.

Parameters

- **db_map** (*DiffDatabaseMapping*) –
- **id** (*int*) – The parameter_value_list id
- **role** (*int*, *optional*) –

get_scenario_alternative_id_list(*db_map*, *scen_id*, *only_visible=True*)

import_data(*db_map_data*, *command_text*='Import data')

Imports the given data into given db maps using the dedicated import functions from `spinedb_api`. Condenses all in a single command for undo/redo.

Parameters

- **db_map_data** (*dict*(*DiffDatabaseMapping*, *dict*())) – Maps dbs to data to be passed as keyword arguments to *get_data_for_import*
- **command_text** (*str*, *optional*) – What to call the command that condenses the operation.

import_data_commands(*db_map*, *data*)

Creates undo commands necessary to import given data.

Parameters

- **db_map** (*DatabaseMapping*) – target database mapping
- **data** (*dict*) – data to import

Yields

SpineDBCommand – import undo command

_import_data_cmds(*db_map*, *data_for_import*, *db_map_error_log*)

add_items(*db_map_data*, *item_type*, *readd*=False, *cascade*=True, *check*=True, *callback*=None)

Returns True if the adding of all items succeeded, False otherwise

update_items(*db_map_data*, *item_type*, *check*=True, *callback*=None)

Returns True if the updating of all items succeeded, False otherwise

add_alternatives(*db_map_data*)

Adds alternatives to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by *DiffDatabaseMapping*

add_scenarios(*db_map_data*)

Adds scenarios to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by *DiffDatabaseMapping*

add_object_classes(*db_map_data*)

Adds object classes to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by *DiffDatabaseMapping*

add_objects(*db_map_data*)

Adds objects to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by *DiffDatabaseMapping*

add_object_metadata(*db_map_data*)

Adds object metadata to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by *DiffDatabaseMapping*

add_relationship_classes(*db_map_data*)

Adds relationship classes to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_relationships(*db_map_data*)

Adds relationships to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_object_groups(*db_map_data*)

Adds object groups to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_entity_groups(*db_map_data*)

Adds entity groups to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_parameter_definitions(*db_map_data*)

Adds parameter definitions to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_parameter_values(*db_map_data*)

Adds parameter values to db without checking integrity.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_parameter_value_lists(*db_map_data*)

Adds parameter_value lists to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_list_values(*db_map_data*)

Adds parameter_value list values to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_features(*db_map_data*)

Adds features to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_tools(*db_map_data*)

Adds tools to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_tool_features(*db_map_data*)

Adds tool features to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_tool_feature_methods(*db_map_data*)

Adds tool feature methods to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_metadata(*db_map_data*)

Adds metadata to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_entity_metadata(*db_map_data*)

Adds entity metadata to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

add_parameter_value_metadata(*db_map_data*)

Adds parameter value metadata to db.

Parameters

db_map_data (*dict*) – lists of items to add keyed by DiffDatabaseMapping

update_alternatives(*db_map_data*)

Updates alternatives in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_scenarios(*db_map_data*)

Updates scenarios in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_object_classes(*db_map_data*)

Updates object classes in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_objects(*db_map_data*)

Updates objects in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_relationship_classes(*db_map_data*)

Updates relationship classes in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_relationships(*db_map_data*)

Updates relationships in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_parameter_definitions(*db_map_data*)

Updates parameter definitions in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_parameter_values(*db_map_data*)

Updates parameter values in db without checking integrity.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_expanded_parameter_values(*db_map_data*)

Updates expanded parameter values in db without checking integrity.

Parameters

db_map_data (*dict*) – lists of expanded items to update keyed by DiffDatabaseMapping

update_parameter_value_lists(*db_map_data*)

Updates parameter_value lists in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_list_values(*db_map_data*)

Updates parameter_value list values in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_features(*db_map_data*)

Updates features in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_tools(*db_map_data*)

Updates tools in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_tool_features(*db_map_data*)

Updates tools features in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_tool_feature_methods(*db_map_data*)

Updates tools feature methods in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_metadata(*db_map_data*)

Updates metadata in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_entity_metadata(*db_map_data*)

Updates entity metadata in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

update_parameter_value_metadata(*db_map_data*)

Updates parameter value metadata in db.

Parameters

db_map_data (*dict*) – lists of items to update keyed by DiffDatabaseMapping

set_scenario_alternatives(*db_map_data*)

Sets scenario alternatives in db.

Parameters

db_map_data (*dict*) – lists of items to set keyed by DiffDatabaseMapping

purge_items(*db_map_purgable_items*)

Purges selected items from given database.

Parameters

db_map_purgable_items (*dict*) – mapping from database map to list of purgable item types

remove_items(*db_map_typed_ids*)

Pushes a command to remove items to undo stack.

_remove_items_commands(*db_map, ids_per_type*)

do_remove_items(*item_type, db_map_ids, callback=None, committing_callback=None*)

Removes items from database.

Parameters

- **item_type** (*str*) – database item type
- **db_map_ids** (*dict*) – mapping DatabaseMapping to removable ids
- **callback** (*Callable, optional*) – function to call after removal is finished
- **committing_callback** (*Callable, optional*) – function to call after removal has been committed

static db_map_ids(*db_map_data*)

static db_map_class_ids(*db_map_data*)

find_cascading_relationship_classes(*db_map_ids, only_visible=True*)

Finds and returns cascading relationship classes for the given object_class ids.

find_cascading_relationships(*db_map_ids, only_visible=True*)

Finds and returns cascading relationships for the given object ids.

find_cascading_parameter_data(*db_map_ids*, *item_type*, *only_visible=True*)

Finds and returns cascading parameter definitions or values for the given entity_class ids.

find_cascading_parameter_values_by_entity(*db_map_ids*, *only_visible=True*)

Finds and returns cascading parameter values for the given entity ids.

find_cascading_parameter_values_by_definition(*db_map_ids*, *only_visible=True*)

Finds and returns cascading parameter values for the given parameter_definition ids.

find_cascading_scenario_alternatives_by_scenario(*db_map_ids*, *only_visible=True*)

Finds and returns cascading scenario alternatives for the given scenario ids.

find_groups_by_entity(*db_map_ids*, *only_visible=True*)

Finds and returns groups for the given entity ids.

duplicate_scenario(*scen_data*, *dup_name*, *db_map*)

duplicate_object(*object_data*, *orig_name*, *dup_name*, *db_maps*)

_get_data_for_export(*db_map_item_ids*)

export_data(*caller*, *db_map_item_ids*, *file_path*, *file_filter*)

_is_url_available(*url*, *logger*)

export_to_sqlite(*file_path*, *data_for_export*, *caller*)

Exports given data into SQLite file.

export_to_json(*file_path*, *data_for_export*, *caller*)

Exports given data into JSON file.

export_to_excel(*file_path*, *data_for_export*, *caller*)

Exports given data into Excel file.

get_items_for_commit(*db_map*, *commit_id*)

static get_all_multi_spine_db_editors()

Yields all instances of MultiSpineDBEditor currently open.

Yields

MultiSpineDBEditor

get_all_spine_db_editors()

Yields all instances of SpineDBEditor currently open.

Yields

SpineDBEditor

_get_existing_spine_db_editor(*db_url_codenames*)

open_db_editor(*db_url_codenames*)

Opens a SpineDBEditor with given urls. Uses an existing MultiSpineDBEditor if any. Also, if the same urls are open in an existing SpineDBEditor, just raises that one instead of creating another.

Parameters

db_url_codenames (*dict*) – mapping url to codename

spinetoolbox.spine_db_parcel

SpineDBParcel class.

Module Contents

Classes

*SpineDBParcel*A class to create parcels of data from a Spine db.

class spinetoolbox.spine_db_parcel.**SpineDBParcel**(*db_mgr*)

A class to create parcels of data from a Spine db. Mainly intended for the *Export selection* action in the Spine db editor:

- `push` methods push items with everything they need to live in a standalone db.
- `full_push` and `inner_push` methods do something more specific

Initializes the parcel object.

Parameters

db_mgr (*SpineDBManager*) –

property data

_get_fields(*db_map*, *item_type*, *field*, *ids*)

Returns a list of field values for items of given type, having given ids.

push_object_class_ids(*db_map_ids*)

Pushes object_class ids.

push_relationship_class_ids(*db_map_ids*)

Pushes relationship_class ids.

push_object_ids(*db_map_ids*)

Pushes object ids.

push_relationship_ids(*db_map_ids*)

Pushes relationship ids.

push_parameter_value_list_ids(*db_map_ids*)

Pushes parameter_value_list ids.

push_parameter_definition_ids(*db_map_ids*, *entity_type*)

Pushes parameter_definition ids.

push_parameter_value_ids(*db_map_ids*, *entity_type*)

Pushes parameter_value ids.

push_object_group_ids(*db_map_ids*)

Pushes object group ids.

push_alternative_ids(*db_map_ids*)

Pushes alternative ids.

push_scenario_ids(*db_map_ids*)

Pushes scenario ids.

push_scenario_alternative_ids(*db_map_ids*)

Pushes scenario_alternative ids.

push_feature_ids(*db_map_ids*)

Pushes feature ids.

push_tool_ids(*db_map_ids*)

Pushes tool ids.

push_tool_feature_ids(*db_map_ids*)

Pushes tool_feature ids.

push_tool_feature_method_ids(*db_map_ids*)

Pushes tool_feature_method ids.

full_push_object_class_ids(*db_map_ids*)

Pushes parameter definitions associated with given object classes. This essentially full_pushes the object classes and their parameter definitions.

full_push_relationship_class_ids(*db_map_ids*)

Pushes parameter definitions associated with given relationship classes. This essentially full_pushes the relationships classes, their parameter definitions, and their member object classes.

full_push_object_ids(*db_map_ids*)

Pushes parameter values associated with objects and with any relationships involving those objects. This essentially full_pushes objects, their relationships, all the parameter values, and all the necessary classes, definitions, and lists.

full_push_relationship_ids(*db_map_ids*)

Pushes parameter values associated with relationships. This essentially full_pushes relationships, their parameter values, and all the necessary classes, definitions, and lists.

full_push_scenario_ids(*db_map_ids*)

inner_push_object_ids(*db_map_ids*)

Pushes object ids, cascading relationship ids, and the associated parameter values, but not any entity classes or parameter definitions. Mainly intended for the *Duplicate object* action.

inner_push_relationship_ids(*db_map_ids*)

Pushes relationship ids, and the associated parameter values, but not any entity classes or parameter definitions.

inner_push_parameter_value_ids(*db_map_ids*, *entity_type*)

Pushes parameter_value ids.

_update_ids(*db_map_ids*, *key*)

Updates ids for given database item.

Parameters

- **db_map_ids** (*dict*) – mapping from DatabaseMappingBase to ids or Asterisk
- **key** (*str*) – the key

`_setdefault(db_map)`

Adds new id sets for given `db_map` or returns existing ones.

Parameters

`db_map` (*DatabaseMappingBase*) – a database map

Returns

mapping from item name to set of ids

Return type

dict

spinetoolbox.spine_db_worker

The SpineDBWorker class

Module Contents

Classes

<i>SpineDBWorker</i>	Does all the communication with a certain DB for SpineDBManager, in a non-GUI thread.
----------------------	---

Functions

<i>_db_map_lock</i> (func)	A wrapper for SpineDBWorker that locks the database for the duration of the wrapped method.
----------------------------	---

Attributes

<i>_CHUNK_SIZE</i>	
--------------------	--

`spinetoolbox.spine_db_worker._CHUNK_SIZE = 1000`

`spinetoolbox.spine_db_worker._db_map_lock(func)`

A wrapper for SpineDBWorker that locks the database for the duration of the wrapped method.

In case the locking fails, the wrapped method will not be invoked.

Parameters

`func` (*Callable*) – method to wrap

class `spinetoolbox.spine_db_worker.SpineDBWorker(db_mgr, db_url)`

Bases: `PySide6.QtCore.QObject`

Does all the communication with a certain DB for SpineDBManager, in a non-GUI thread.

`_more_available`

_will_have_children_change

_get_parents(*item_type*)

clean_up()

query(*sq_name*)

For tests.

_query(*sq_name*)

get_db_map(*args, **kwargs)

_get_db_map(*args, **kwargs)

reset_queries()

Resets queries and clears caches.

_reset_fetching_if_required(*parent*)

Sets fetch parent's token or resets the parent if fetch tokens don't match.

Parameters

parent ([FetchParent](#)) – fetch parent

advance_query(*item_type*)

Advances the DB query that fetches items of given type.

Parameters

item_type (*str*) –

Returns

bool

_advance_query(*item_type*, *callback=None*)

Schedules a progression of the DB query that fetches items of given type. Adds the given callback to the collection of callbacks to call when the query progresses.

Parameters

- **item_type** (*str*) –
- **callback** (*Function or None*) –

Returns

True if query is being advanced, False otherwise

Return type

bool

_do_advance_query(*item_type*)

Advances the DB query that fetches items of given type and caches the results.

Parameters

item_type (*str*) –

Returns

True if new items were fetched from the DB, False otherwise.

Return type

bool

`_register_fetch_parent(parent)`

Registers the given parent and starts checking whether it will have children if fetched.

Parameters

`parent` (`FetchParent`) –

`_update_parents_will_have_children(item_type)`

Schedules a restart of the process that checks whether parents associated to given type will have children.

Parameters

`item_type` (`str`) –

`_do_update_parents_will_have_children(item_type)`

Updates the `will_have_children` property for all parents associated to given type.

Parameters

`item_type` (`str`) –

`static _handle_will_have_children_change(parents)`

`_iterate_cache(parent)`

Iterates the cache for given parent while updating its `position` property. Iterated items are added to the parent if it accepts them.

Parameters

`parent` (`FetchParent`) – the parent.

Returns

Whether the parent can stop fetching from now

Return type

`bool`

`_bind_item(parent, item)`

`_add_item(parent, item)`

`_update_item(parent, item)`

`_remove_item(parent, item)`

`_make_add_item_callback(parent)`

`_make_update_item_callback(parent)`

`_make_remove_item_callback(parent)`

`can_fetch_more(parent)`

Returns whether more data can be fetched for parent. Also, registers the parent to notify it of any relevant DB modifications later on.

Parameters

`parent` (`FetchParent`) – fetch parent

Returns

True if more data is available, False otherwise

Return type

`bool`

fetch_more(*parent*)

Fetches items from the database.

Parameters

parent ([FetchParent](#)) – fetch parent

_handle_query_advanced(*parent*)

fetch_all(*fetch_item_types=None, include_descendants=False, include_ancestors=False*)

_fetch_all(*item_types*)

_populate_commit_cache(*item_type, items*)

close_db_map()

_close_db_map()

_split_items_by_type(*item_type, items*)

add_items(*orig_items, item_type, readd, cascade, check, cache, callback*)

Adds items to db.

Parameters

- **orig_items** (*dict*) – lists of items to add or update
- **item_type** (*str*) – item type
- **readd** (*bool*) – Whether to re-add items that were previously removed
- **cascade** (*bool*) – Whether to add items in cascade or just the root items
- **check** (*bool*) – Whether to check integrity
- **cache** (*dict*) – Cache
- **callback** (*None or function*) – something to call with the result

Returns

True if adding successful, False otherwise

Return type

bool

_rebind_recursively(*item*)

Rebinds a cache item and its referrers to fetch parents.

Parameters

item (*CacheItem*) – item to rebind

update_items(*orig_items, item_type, check, cache, callback*)

Updates items in db.

Parameters

- **orig_items** (*dict*) – lists of items to add or update
- **item_type** (*str*) – item type
- **check** (*bool*) – Whether or not to check integrity
- **cache** (*dict*) – Cache
- **callback** (*None or function*) – something to call with the result

Returns

True if update successful, False otherwise

Return type

bool

remove_items(*item_type*, *ids*, *callback*, *committing_callback*)

Removes items from database.

Parameters

- **item_type** (*str*) – item type
- **ids** (*Iterable of int*) – removable item ids
- **callback** (*Callable, optional*) – function to call after items have been removed
- **committing_callback** (*Callable, optional*) – function to call after remove operation has been committed only

commit_session(*commit_msg*, *cookie=None*)

Initiates commit session.

Parameters

- **commit_msg** (*str*) – commit message
- **cookie** (*Any*) – a cookie to include in session_committed signal

Returns

True if commit succeeded, False otherwise

Return type

success (bool)

_commit_session(*commit_msg*, *undo_stack*, *cookie=None*)

Commits session for given database maps.

Parameters

- **commit_msg** (*str*) – commit message
- **undo_stack** (*AgedUndoStack*) – undo stack that outlive the DB manager
- **cookie** (*Any*) – a cookie to include in session_committed signal

Returns

True if commit succeeded, False otherwise

Return type

bool

rollback_session()

Initiates rollback session in the worker thread.

_rollback_session(*undo_stack*)

Rolls back session.

Parameters

- undo_stack** (*AgedUndoStack*) – undo stack that outlive the DB manager

spinetoolbox.spine_engine_manager

Contains SpineEngineManagerBase.

Module Contents

Classes

SpineEngineManagerBase

LocalSpineEngineManager

<i>RemoteSpineEngineManager</i>	Responsible for remote project execution.
---------------------------------	---

Functions

<i>make_engine_manager</i> ([remote_execution_enabled, job_id])	Returns either a Local or a remote Spine Engine Manager based on settings.
---	--

class spinetoolbox.spine_engine_manager.**SpineEngineManagerBase**

abstract **run_engine**(*engine_data*)

Runs an engine with given data.

Parameters

engine_data (*dict*) – The engine data.

abstract **get_engine_event**()

Gets next event from a running engine.

Returns

two element tuple: event type identifier string, and event data dictionary

Return type

tuple(str,dict)

abstract **stop_engine**()

Stops a running engine.

abstract **answer_prompt**(*item_name*, *accepted*)

Answers prompt.

Parameters

- **item_name** (*str*) – The item that emitted the prompt
- **accepted** (*bool*) – The user's decision.

abstract **restart_kernel**(*connection_file*)

Restarts the jupyter kernel associated to given connection file.

Parameters

connection_file (*str*) – path of connection file

abstract shutdown_kernel(*connection_file*)

Shuts down the jupyter kernel associated to given connection file.

Parameters

connection_file (*str*) – path of connection file

abstract issue_persistent_command(*persistent_key, command*)

Issues a command to a persistent process.

Parameters

- **persistent_key** (*tuple*) – persistent identifier
- **command** (*str*) – command to issue

Returns

stdin, stdout, and stderr messages (dictionaries with two keys: type, and data)

Return type

generator

abstract is_persistent_command_complete(*persistent_key, command*)

Checks whether a command is complete.

Parameters

- **key** (*tuple*) – persistent identifier
- **cmd** (*str*) – command to issue

Returns

bool

abstract restart_persistent(*persistent_key*)

Restarts a persistent process.

Parameters

persistent_key (*tuple*) – persistent identifier

Returns

stdout and stderr messages (dictionaries with two keys: type, and data)

Return type

generator

abstract interrupt_persistent(*persistent_key*)

Interrupts a persistent process.

Parameters

persistent_key (*tuple*) – persistent identifier

abstract kill_persistent(*persistent_key*)

Kills a persistent process.

Parameters

persistent_key (*tuple*) – persistent identifier

abstract get_persistent_completions(*persistent_key, text*)

Returns a list of auto-completion options from given text.

Parameters

- **persistent_key** (*tuple*) – persistent identifier

- **text** (*str*) – text to complete

Returns

list of *str*

abstract get_persistent_history_item(*persistent_key, text, prefix, backwards*)

Returns an item from persistent history.

Parameters

persistent_key (*tuple*) – persistent identifier

Returns

history item or empty string if none

Return type

str

class `spinetoolbox.spine_engine_manager.LocalSpineEngineManager`

Bases: [*SpineEngineManagerBase*](#)

run_engine(*engine_data*)

Runs an engine with given data.

Parameters

engine_data (*dict*) – The engine data.

get_engine_event()

Gets next event from a running engine.

Returns

two element tuple: event type identifier string, and event data dictionary

Return type

tuple(*str*,*dict*)

stop_engine()

Stops a running engine.

answer_prompt(*item_name, accepted*)

Answers prompt.

Parameters

- **item_name** (*str*) – The item that emitted the prompt
- **accepted** (*bool*) – The user's decision.

restart_kernel(*connection_file*)

Restarts the jupyter kernel associated to given connection file.

Parameters

connection_file (*str*) – path of connection file

shutdown_kernel(*connection_file*)

Shuts down the jupyter kernel associated to given connection file.

Parameters

connection_file (*str*) – path of connection file

kernel_managers()

issue_persistent_command(*persistent_key*, *command*)

Issues a command to a persistent process.

Parameters

- **persistent_key** (*tuple*) – persistent identifier
- **command** (*str*) – command to issue

Returns

stdin, stdout, and stderr messages (dictionaries with two keys: type, and data)

Return type

generator

is_persistent_command_complete(*persistent_key*, *command*)

Checks whether a command is complete.

Parameters

- **key** (*tuple*) – persistent identifier
- **cmd** (*str*) – command to issue

Returns

bool

restart_persistent(*persistent_key*)

Restarts a persistent process.

Parameters

persistent_key (*tuple*) – persistent identifier

Returns

stdout and stderr messages (dictionaries with two keys: type, and data)

Return type

generator

interrupt_persistent(*persistent_key*)

Interrupts a persistent process.

Parameters

persistent_key (*tuple*) – persistent identifier

kill_persistent(*persistent_key*)

Kills a persistent process.

Parameters

persistent_key (*tuple*) – persistent identifier

get_persistent_completions(*persistent_key*, *text*)

Returns a list of auto-completion options from given text.

Parameters

- **persistent_key** (*tuple*) – persistent identifier
- **text** (*str*) – text to complete

Returns

list of str

get_persistent_history_item(*persistent_key, text, prefix, backwards*)

Returns an item from persistent history.

Parameters

persistent_key (*tuple*) – persistent identifier

Returns

history item or empty string if none

Return type

str

class `spinetoolbox.spine_engine_manager.RemoteSpineEngineManager`(*job_id=""*)

Bases: [*SpineEngineManagerBase*](#)

Responsible for remote project execution.

Initializer.

make_engine_client(*host, port, security, sec_folder, ping=True*)

Creates a client for connecting to Spine Engine Server.

run_engine(*engine_data*)

Makes an engine client for communicating with the engine server. Starts a thread for monitoring the DAG execution on server.

Parameters

engine_data (*dict*) – The engine data.

get_engine_event()

Returns the next engine execution event.

clean_up()

Closes EngineClient and joins _runner thread if still active.

stop_engine()

Sends a request to stop execution on Server then waits for _runner thread to end.

_run()

Sends a start execution request to server with the job Id. Sets up a subscribe socket according to the publish port received from server. Passes received events to SpineEngineWorker for processing. After execution has finished, downloads new files from server.

answer_prompt(*item_name, accepted*)

See base class.

restart_kernel(*connection_file*)

See base class.

shutdown_kernel(*connection_file*)

See base class.

is_persistent_command_complete(*persistent_key, command*)

Checks whether a command is complete.

Parameters

- **key** (*tuple*) – persistent identifier
- **cmd** (*str*) – command to issue

Returns

bool

issue_persistent_command(*persistent_key*, *command*)

Issues a command to a persistent process.

Parameters

- **persistent_key** (*tuple*) – persistent identifier
- **command** (*str*) – command to issue

Returns

stdin, stdout, and stderr messages (dictionaries with two keys: type, and data)

Return type

generator

restart_persistent(*persistent_key*)

See base class.

interrupt_persistent(*persistent_key*)

See base class.

kill_persistent(*persistent_key*)

See base class.

get_persistent_completions(*persistent_key*, *text*)

See base class.

get_persistent_history_item(*persistent_key*, *text*, *prefix*, *backwards*)

Returns an item from persistent history.

Parameters**persistent_key** (*tuple*) – persistent identifier**Returns**

history item or empty string if none

Return type

str

`spinetoolbox.spine_engine_manager.make_engine_manager(remote_execution_enabled=False, job_id=")`

Returns either a Local or a remote Spine Engine Manager based on settings.

Parameters

- **remote_execution_enabled** (*bool*) – True returns a local Spine Engine Manager instance,
- **instance** (*False* returns a remote Spine Engine Manager) –
- **job_id** (*str*) – Server execution job Id

spinetoolbox.spine_engine_worker

Contains SpineEngineWorker.

Module Contents

Classes

SpineEngineWorker

param engine_data
engine data

Functions

_handle_dag_execution_started(project_items)

_handle_node_execution_ignored(project_items)

_handle_node_execution_started(item, direction)

_handle_node_execution_finished(item, direction, ...)

_handle_event_message_arrived(item, filter_id, ...)

_handle_process_message_arrived(item, filter_id, ...)

_handle_prompt_arrived(prompt, engine_mgr[, logger])

_handle_flash_arrived(connection)

_mark_all_items_failed(items)

Fails all project items.

spinetoolbox.spine_engine_worker._handle_dag_execution_started(project_items)

spinetoolbox.spine_engine_worker._handle_node_execution_ignored(project_items)

spinetoolbox.spine_engine_worker._handle_node_execution_started(item, direction)

spinetoolbox.spine_engine_worker._handle_node_execution_finished(item, direction, item_state)

spinetoolbox.spine_engine_worker._handle_event_message_arrived(item, filter_id, msg_type, msg_text)

spinetoolbox.spine_engine_worker._handle_process_message_arrived(item, filter_id, msg_type, msg_text)

spinetoolbox.spine_engine_worker._handle_prompt_arrived(prompt, engine_mgr, logger=None)

spinetoolbox.spine_engine_worker._handle_flash_arrived(connection)

`spinetoolbox.spine_engine_worker._mark_all_items_failed(items)`

Fails all project items.

Parameters

items (*list of ProjectItem*) – project items

class `spinetoolbox.spine_engine_worker.SpineEngineWorker`(*engine_data, dag, dag_identifier, project_items, connections, logger, job_id*)

Bases: `PySide6.QtCore.QObject`

Parameters

- **engine_data** (*dict*) – engine data
- **dag** (*DirectedGraphHandler*) –
- **dag_identifier** (*str*) –
- **project_items** (*dict*) – mapping from project item name to `ProjectItem`
- **connections** (*dict*) – mapping from jump name to `LoggingConnection` or `LoggingJump`
- **logger** (*LoggerInterface*) – a logger
- **job_id** (*str*) – Job id for remote execution

property `job_id`

property `engine_data`

Engine data dictionary.

finished

_mark_items_ignored

_dag_execution_started

_node_execution_started

_node_execution_finished

_event_message_arrived

_process_message_arrived

_prompt_arrived

_flash_arrived

_all_items_failed

get_engine_data()

Returns the engine data. Together with `self.set_engine_data()` it can be used to modify the workflow after it's initially created. We use it at the moment for creating Julia sysimages.

Returns

`dict`

set_engine_data(*engine_data*)

Sets the engine data.

Parameters

engine_data (*dict*) – New data

_handle_event_message_arrived_silent(*item, filter_id, msg_type, msg_text*)

_handle_process_message_arrived_silent(*item, filter_id, msg_type, msg_text*)

stop_engine()

engine_final_state()

thread()

_connect_log_signals(*silent*)

start(*silent=False*)

Connects log signals.

Parameters

silent (*bool, optional*) – If True, log messages are not forwarded to the loggers but saved in internal dicts.

_included_and_ignored_items()

Returns two lists, where the first one contains project items that are about to be executed and the second one contains project items that are about to be ignored.

_included_items(*permitted_items, connections*)

Collects a list of project item names that are going to be executed in this DAG based on execution permits and connections in the DAG.

Parameters

- **permitted_items** (*dict*) – Mapping of item names to bool. True items have been selected by user for execution.
- **connections** (*list*) – Serialized connections

Returns

Project item names

Return type

list

do_work()

Does the work and emits finished when done.

_process_event(*event_type, data*)

_handle_prompt(*prompt*)

_handle_flash(*flash*)

_handle_standard_execution_msg(*msg*)

_handle_persistent_execution_msg(*msg*)

_handle_kernel_execution_msg(*msg*)

```

_handle_process_msg(data)
_do_handle_process_msg(item_name, filter_id, msg_type, msg_text)
_handle_event_msg(data)
_do_handle_event_msg(item_name, filter_id, msg_type, msg_text)
_handle_node_execution_started(data)
_do_handle_node_execution_started(item_name, direction)
    Starts item icon animation when executing forward.
_handle_node_execution_finished(data)
_do_handle_node_execution_finished(item_name, direction, state, item_state)
_handle_server_status_msg(data)
clean_up()

```

spinetoolbox.ui_main

Contains ToolboxUI class.

Module Contents

Classes

ToolboxUI

Class for application main GUI functions.

```

class spinetoolbox.ui_main.ToolboxUI
    Bases: PySide6.QtWidgets.QMainWindow
    Class for application main GUI functions.
    Initializes application and main window.

    msg
    msg_success
    msg_error
    msg_warning
    msg_proc
    msg_proc_error
    information_box
    error_box
    jupyter_console_requested

```

kernel_shutdown

persistent_console_requested

eventFilter(*obj, ev*)

_setup_properties_title()

connect_signals()

Connect signals.

_open_active_item_dir(*_checked=False*)

static set_error_mode()

Sets Windows error mode to show all error dialog boxes from subprocesses.

See <https://docs.microsoft.com/en-us/windows/win32/api/errhandlingapi/nf-errhandlingapi-seterrormode> for documentation.

_update_qsettings()

Updates obsolete settings.

_update_execute_enabled()

_update_execute_selected_enabled()

update_window_modified(*clean*)

Updates window modified status and save actions depending on the state of the undo stack.

parse_project_item_modules()

Collects data from project item factories.

set_work_directory(*new_work_dir=None*)

Creates a work directory if it does not exist or changes the current work directory to given.

Parameters

new_work_dir (*str, optional*) – If given, changes the work directory to given and creates the directory if it does not exist.

project()

Returns current project or None if no project open.

Returns

current project or None

Return type

SpineToolboxProject

qsettings()

Returns application preferences object.

item_specification_factories()

Returns project item specification factories.

Returns

specification factories

Return type

list of ProjectItemSpecificationFactory

update_window_title()

Updates main window title.

init_project(*project_dir*)

Initializes project at application start-up.

Opens the last project that was open when app was closed (if enabled in Settings) or starts the app without a project.

Parameters

project_dir (*str*) – project directory

new_project()

Opens a file dialog where user can select a directory where a project is created. Pops up a question box if selected directory is not empty or if it already contains a Spine Toolbox project. Initial project name is the directory name.

create_project(*proj_dir*)

Creates new project and sets it active.

Parameters

proj_dir (*str*) – Path to project directory

open_project(*load_dir=None*)

Opens project from a selected or given directory.

Parameters

load_dir (*str*, *optional*) – Path to project base directory. If default value is used, a file explorer dialog is opened where the user can select the project to open.

Returns

True when opening the project succeeded, False otherwise

Return type

bool

restore_project(*project_dir*, *ask_confirmation=True*)

Initializes UI, Creates project, models, connections, etc., when opening a project.

Parameters

- **project_dir** (*str*) – Project directory
- **ask_confirmation** (*bool*) – True closes the previous project with a confirmation box if user has enabled this

Returns

True when restoring project succeeded, False otherwise

Return type

bool

_toolbars()

Yields all toolbars in the window.

_disable_project_actions()

Disables all project-related actions, except New project, Open project and Open recent. Called in the constructor and when closing a project.

_enable_project_actions()

Enables all project-related actions. Called when a new project is created and when a project is opened.

refresh_toolbars()

Set toolbars' color using highest possible contrast.

show_recent_projects_menu()

Updates and sets up the recent projects menu to File-Open recent menu item.

fetch_kernels()

Starts a thread for fetching local kernels.

stop_fetching_kernels()

Terminates kernel fetcher thread.

restore_override_cursor()

Restores default mouse cursor.

save_project()

Saves project.

save_project_as()

Asks user for a new project directory and duplicates the current project there. The name of the duplicated project will be the new directory name. The duplicated project is activated.

close_project(*ask_confirmation=True*)

Closes the current project.

Parameters

ask_confirmation (*bool*) – if False, no confirmation whatsoever is asked from user

Returns

True when no project open or when it's closed successfully, False otherwise.

Return type

bool

set_project_description(*_=False*)

Opens a dialog where the user can enter a new description for the project.

init_project_item_model()

Initializes project item model. Create root and category items and add them to the model.

init_specification_model()

Initializes specification model.

make_item_properties_uis()

_make_properties_tab(*properties_ui*)

add_project_items(*items_dict, silent=False*)

Pushes an AddProjectItemsCommand to the undo stack.

Parameters

- **items_dict** (*dict*) – mapping from item name to item dictionary
- **silent** (*bool*) – if True, suppress log messages

supports_specifications(*item_type*)

Returns True if given project item type supports specifications.

Returns

True if item supports specifications, False otherwise

Return type

bool

restore_ui()

Restore UI state from previous session.

clear_ui()

Clean UI to make room for a new or opened project.

undo_critical_commands()

Undoes critical commands in the undo stack.

Returns

False if any critical commands aren't successfully undone

Return type

Bool

overwrite_check(*project_dir*)

Checks if given directory is a project directory and/or empty And asks the user what to do in that case.

Parameters**project_dir** (*str*) – Abs. path to a directory**Returns**

True if user wants to overwrite an existing project or if the directory is not empty and the user wants to make it into a Spine Toolbox project directory anyway. False if user cancels the action.

Return type

bool

item_selection_changed(*selected*, *deselected*)

Synchronizes selection with scene. The scene handles item/link de/activation.

refresh_active_elements(*active_project_item*, *active_link_item*, *selected_item_names*)**_activate_properties_tab()****_set_active_project_item(*active_project_item*)****Parameters****active_project_item** (*ProjectItemBase* or *NoneType*) –**_set_active_link_item(*active_link_item*)**

Sets active link and connects to corresponding properties widget.

Parameters**active_link_item** (*LoggingConnection* or *LoggingJump*, *optional*) –**activate_no_selection_tab()**

Shows 'No Selection' tab.

activate_item_tab()

Shows active project item properties tab according to item type.

activate_link_tab()

Shows link properties tab.

update_properties_ui()

_get_active_properties_widget()

add_specification(*specification*)

Pushes an AddSpecificationCommand to undo stack.

import_specification()

Opens a file dialog where the user can select an existing specification definition file (.json). If file is valid, pushes AddSpecificationCommand to undo stack.

replace_specification(*name*, *specification*)

Pushes an ReplaceSpecificationCommand to undo stack.

repair_specification(*name*)

Repairs specification if it is broken.

Parameters

name (*str*) – specification’s name

prompt_save_location(*title*, *proposed_path*, *file_filter*)

Shows a dialog for the user to select a path to save a file.

Parameters

- **title** (*str*) – dialog window title
- **proposed_path** (*str*) – A proposed location.
- **file_filter** (*str*) – file extension filter

Returns

absolute path or None if dialog was cancelled

Return type

str

_log_specification_saved(*name*, *path*)

Prints a message in the event log, saying that given spec was saved in a certain location, together with a clickable link to change the location.

Parameters

- **name** (*str*) – specification’s name
- **path** (*str*) – specification’s file path

remove_all_items()

Pushes a RemoveAllProjectItemsCommand to the undo stack.

register_anchor_callback(*url*, *callback*)

Registers a callback for a given anchor in event log, see `open_anchor()`. Used by ToolFactory. `repair_specification()`.

Parameters

- **url** (*str*) – The anchor url
- **callback** (*function*) – A function to call when the anchor is clicked on event log.

open_anchor(*qurl*)

Open file explorer in the directory given in qurl.

Parameters

qurl (*QUrl*) – The url to open

_change_specification_file_location(*name*)

Prompts user for new location for a project item specification.

Delegates saving to project if one is open by pushing a command to the undo stack, otherwise tries to find the specification from the plugin manager.

Parameters

name (*str*) – specification's name

show_specification_context_menu(*ind*, *global_pos*)

Context menu for item specifications.

Parameters

- **ind** (*QModelIndex*) – In the ProjectItemSpecificationModel
- **global_pos** (*QPoint*) – Mouse position

edit_specification(*index*, *item*)

Opens a specification editor widget.

Parameters

- **index** (*QModelIndex*) – Index of the item (from double-click or context menu signal)
- **item** (*ProjectItem*, *optional*) –

remove_specification(*index*)

Removes specification from project.

Parameters

index (*QModelIndex*) – Index of the specification item

open_specification_file(*index*)

Open the specification definition file in the default (.json) text-editor.

Parameters

index (*QModelIndex*) – Index of the item

new_db_editor()**_handle_zoom_minus_pressed()**

Slot for handling case when '-' button in menu is pressed.

_handle_zoom_plus_pressed()

Slot for handling case when '+' button in menu is pressed.

_handle_zoom_reset_pressed()

Slot for handling case when 'reset zoom' button in menu is pressed.

add_zoom_action()

Setups zoom widget action in view menu.

restore_dock_widgets()

Dock all floating and or hidden QDockWidgets back to the main window.

_add_execute_actions()

Adds execution handler actions to the main window.

set_debug_qactions()

Sets shortcuts for QActions that may be needed in debugging.

add_menu_actions()

Adds extra actions to Edit and View menu.

toggle_properties_tabbar_visibility()

Shows or hides the tab bar in properties dock widget. For debugging purposes.

update_datetime()

Returns a boolean, which determines whether date and time is prepended to every Event Log message.

add_message(*msg*)

Append regular message to Event Log.

Parameters

msg (*str*) – String written to QTextBrowser

add_success_message(*msg*)

Append message with green text color to Event Log.

Parameters

msg (*str*) – String written to QTextBrowser

add_error_message(*msg*)

Append message with red color to Event Log.

Parameters

msg (*str*) – String written to QTextBrowser

add_warning_message(*msg*)

Append message with yellow (golden) color to Event Log.

Parameters

msg (*str*) – String written to QTextBrowser

add_process_message(*msg*)

Writes message from stdout to process output QTextBrowser.

Parameters

msg (*str*) – String written to QTextBrowser

add_process_error_message(*msg*)

Writes message from stderr to process output QTextBrowser.

Parameters

msg (*str*) – String written to QTextBrowser

override_console_and_execution_list()**_override_console()**

Sets the jupyter console of the active project item in Jupyter Console and updates title.

_do_override_console(*console*)**_override_execution_list()**

Displays executions of the active project item in Executions and updates title.

_restore_original_console()

Sets the Console back to the original.

_set_override_console(*console*)

_refresh_console_execution_list()

Refreshes console executions as the active project item starts new executions.

_select_console_execution(*current*, *_previous*)

Sets the console of the selected execution in Console.

show_add_project_item_form(*item_type*, *x=0*, *y=0*, *spec=""*)

Show add project item widget.

supports_specification(*item_type*)

Returns True if given item type supports specifications.

Parameters

item_type (*str*) – item's type

Returns

True if item supports specifications, False otherwise

Return type

bool

show_specification_form(*item_type*, *specification=None*, *item=None*, *kwargs*)**

Shows specification widget.

Parameters

- **item_type** (*str*) – item's type
- **specification** (*ProjectItemSpecification*, *optional*) – specification
- **item** (*ProjectItem*, *optional*) – project item
- ****kwargs** – parameters passed to the specification widget

static get_all_multi_tab_spec_editors(*item_type*)**_get_existing_spec_editor(*item_type*, *specification*, *item*)****show_settings()**

Show Settings widget.

show_about()

Show About Spine Toolbox form.

show_user_guide()

Open Spine Toolbox documentation index page in browser.

show_getting_started_guide()

Open Spine Toolbox Getting Started HTML page in browser.

retrieve_project()

Retrieves project from server.

engine_server_settings()

Returns the user given Spine Engine Server settings in a tuple.

show_item_context_menu(*pos*)

Context menu for project items listed in the project QTreeView.

Parameters

pos (*QPoint*) – Mouse position

show_project_or_item_context_menu(*pos*, *index*)

Creates and shows the project item context menu.

Parameters

- **pos** (*QPoint*) – Mouse position
- **index** (*QModelIndex*, *optional*) – Index of concerned item or None

show_link_context_menu(*pos*, *link*)

Context menu for connection links.

Parameters

- **pos** (*QPoint*) – Mouse position
- **link** (*Link*(*QGraphicsPathItem*)) – The concerned link

refresh_edit_action_states()

Sets the enabled/disabled state for copy, paste, duplicate, and remove actions in File-Edit menu, project tree view context menu, and in Design View context menus just before the menus are shown to user.

enable_edit_actions()

Enables project item edit actions after a QMenu has been shown. This is needed to enable keyboard shortcuts (e.g. Ctrl-C & del) again.

_tasks_before_exit()

Returns a list of tasks to perform before exiting the application.

Possible tasks are:

- “*prompt exit*”: prompt user if quitting is really desired
- “*prompt save*”: prompt user if project should be saved before quitting
- “*save*”: save project before quitting

Returns

a list containing zero or more tasks

_perform_pre_exit_tasks()

Prompts user to confirm quitting and saves the project if necessary.

Returns

True if exit should proceed, False if the process was cancelled

_confirm_exit()

Confirms exiting from user.

Returns

True if exit should proceed, False if user cancelled

_confirm_project_close()

Confirms exit from user and saves the project if requested.

Returns

True if exiting should proceed, False if user cancelled

remove_path_from_recent_projects(*p*)

Removes entry that contains given path from the recent project files list in QSettings.

Parameters**p** (*str*) – Full path to a project directory**clear_recent_projects()**

Clears recent projects list in File->Open recent menu.

update_recent_projects()

Adds a new entry to QSettings variable that remembers twenty most recent project paths.

closeEvent(event)

Method for handling application exit.

Parameters**event** (*QCloseEvent*) – PySide6 event**_serialize_selected_items()**

Serializes selected project items into a dictionary.

The serialization protocol tries to imitate the format in which projects are saved.

Returns

a dict containing serialized version of selected project items

Return type

dict

_deserialized_item_position_shifts(item_dicts)

Calculates horizontal and vertical shifts for project items being deserialized.

If the mouse cursor is on the Design view we try to place the items under the cursor. Otherwise the items will get a small shift so they don't overlap a possible item below. In case the items don't fit the scene rect we clamp their coordinates within it.

Parameters**item_dicts** (*dict*) – a dictionary of serialized items being deserialized**Returns**

a tuple of (horizontal shift, vertical shift) in scene's coordinates

Return type

tuple

static _set_deserialized_item_position(item_dict, shift_x, shift_y, scene_rect)

Moves item's position by shift_x and shift_y while keeping it within the limits of scene_rect.

_deserialize_items(items_dict, duplicate_files=False)

Deserializes project items from a dictionary and adds them to the current project.

Parameters**items_dict** (*dict*) – serialized project items**project_item_to_clipboard()**

Copies the selected project items to system's clipboard.

project_item_from_clipboard(duplicate_files=False)

Adds project items in system's clipboard to the current project.

Parameters**duplicate_files** (*bool*) – Duplicate files boolean

duplicate_project_item(*duplicate_files=False*)

Duplicates the selected project items.

_share_item_edit_actions()

Adds generic actions to project tree view and Design View.

_show_message_box(*title, message*)

Shows an information message box.

_show_error_box(*title, message*)

_connect_project_signals()

Connects signals emitted by project.

_execute_project(*_=False*)

Executes all DAGs in project.

_execute_selection(*_=False*)

Executes selected items.

_stop_execution(*_=False*)

Stops execution in progress.

_set_execution_in_progress()

_unset_execution_in_progress()

set_icon_and_properties_ui(*item_name*)

Adds properties UI to given project item.

Parameters

item_name (*str*) – item’s name

project_item_properties_ui(*item_type*)

Returns the properties tab widget’s ui.

Parameters

item_type (*str*) – project item’s type

Returns

item’s properties tab widget

Return type

QWidget

project_item_icon(*item_type*)

_open_project_directory()

Opens project’s root directory in system’s file browser.

_open_project_item_directory()

Opens project item’s directory in system’s file browser.

_remove_selected_items()

Pushes commands to remove selected project items and links from project.

_rename_project_item()

Renames current project item.

item_category_context_menu()

Creates a context menu for category items.

Returns

category context menu

Return type

QMenu

project_item_context_menu(*additional_actions*)

Creates a context menu for project items.

Parameters

additional_actions (*list of QAction*) – actions to be prepended to the menu

Returns

project item context menu

Return type

QMenu

start_detached_jupyter_console(*kernel_name, icon, conda*)

Launches a new detached Console with the given kernel name or activates an existing Console if the kernel is already running.

Parameters

- **kernel_name** (*str*) – Requested kernel name
- **icon** (*QIcon*) – Icon representing the kernel language
- **conda** (*bool*) – Is this a Conda kernel?

_setup_jupyter_console(*item, filter_id, kernel_name, connection_file, connection_file_dict*)

Sets up jupyter console, eventually for a filter execution.

Parameters

- **item** (*ProjectItem*) – Item
- **filter_id** (*str*) – Filter identifier
- **kernel_name** (*str*) – Jupyter kernel name
- **connection_file** (*str*) – Path to connection file
- **connection_file_dict** (*dict*) – Contents of connection file when kernel manager runs on Spine Engine Server

_handle_kernel_shutdown(*item, filter_id*)

Closes the kernel client when kernel manager has been shutdown due to an enabled ‘Kill consoles at the end of execution’ option.

Parameters

- **item** (*ProjectItem*) – Item
- **filter_id** (*str*) – Filter identifier

_setup_persistent_console(*item, filter_id, key, language*)

Sets up persistent console, eventually for a filter execution.

Parameters

- **item** (*ProjectItem*) – Item

- **filter_id** (*str*) – Filter identifier
- **key** (*tuple*) – Key
- **language** (*str*) – Language (e.g. ‘python’ or ‘julia’)

persistent_killed(*item*, *filter_id*)

add_persistent_stdin(*item*, *filter_id*, *data*)

add_persistent_stdout(*item*, *filter_id*, *data*)

add_persistent_stderr(*item*, *filter_id*, *data*)

_get_console(*item*, *filter_id*)

_make_jupyter_console(*item*, *kernel_name*, *connection_file*)

Creates a new JupyterConsoleWidget for given connection file if none exists yet, and returns it.

Parameters

- **item** ([ProjectItem](#)) – Item that owns the console
- **kernel_name** (*str*) – Name of the kernel
- **connection_file** (*str*) – Path of kernel connection file

Returns

JupyterConsoleWidget

_make_persistent_console(*item*, *key*, *language*)

Creates a new PersistentConsoleWidget for given process key.

Parameters

- **item** ([ProjectItem](#)) – Item that owns the console
- **key** (*tuple*) – persistent process key in spine engine
- **language** (*str*) – for syntax highlighting and prompting, etc.

Returns

PersistentConsoleWidget

_cleanup_jupyter_console(*conn_file*)

Removes reference to a Jupyter Console and closes the kernel manager on Engine.

_shutdown_engine_kernels()

Shuts down all persistent and Jupyter kernels managed by Spine Engine.

_close_consoles()

Closes all Persistent and Jupyter Console widgets.

restore_and_activate()

Brings the app main window into focus.

static _make_log_entry_title(*title*)

start_execution(*timestamp*)

Starts execution.

Parameters

timestamp (*str*) – time stamp

add_log_message(*item_name*, *filter_id*, *message*)

Adds a message to an item's execution log.

Parameters

- **item_name** (*str*) – item name
- **filter_id** (*str*) – filter identifier
- **message** (*str*) – formatted message

spinetoolbox.version

Version info for Spine Toolbox package. Inspired by python `sys.version` and `sys.version_info`.

Module Contents

Classes

<i>VersionInfo</i>	A class for a named tuple containing the five components of the version number: major, minor,
--------------------	---

Attributes

<i>split</i>
<i>__version_info__</i>
<i>__version__</i>

class spinetoolbox.version.VersionInfo

Bases: `NamedTuple`

A class for a named tuple containing the five components of the version number: major, minor, micro, release-level, and serial. All values except `releaselevel` are integers; the release level is 'dev', 'alpha', 'beta', 'candidate', or 'final'.

major: `int`

minor: `int`

micro: `int`

releaselevel: `str`

serial: `int`

__str__() → `str`

Create a version string following PEP 440

`spinetoolbox.version.split`

`spinetoolbox.version.__version_info__`

`spinetoolbox.version.__version__`

20.1.3 Package Contents

`spinetoolbox.__version__`

`spinetoolbox.__version_info__`

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

BIBLIOGRAPHY

- [CB14] Chris Beams. 2014. ‘How to Write a Git Commit Message.’ <https://chris.beams.io/posts/git-commit/>
- [JF18] Jeff Forcier. 2018. ‘Contributing to Open Source Projects.’ <https://contribution-guide-org.readthedocs.io/>

PYTHON MODULE INDEX

S

spinetoolbox, 175
spinetoolbox.__main__, 503
spinetoolbox._version, 504
spinetoolbox.config, 504
spinetoolbox.execution_managers, 505
spinetoolbox.fetch_parent, 507
spinetoolbox.headless, 511
spinetoolbox.helpers, 516
spinetoolbox.kernel_fetcher, 534
spinetoolbox.link, 535
spinetoolbox.load_project_items, 541
spinetoolbox.log_mixin, 541
spinetoolbox.logger_interface, 542
spinetoolbox.main, 543
spinetoolbox.metaobject, 544
spinetoolbox.mvcmodels, 175
spinetoolbox.mvcmodels.array_model, 175
spinetoolbox.mvcmodels.compound_table_model, 177
spinetoolbox.mvcmodels.empty_row_model, 181
spinetoolbox.mvcmodels.file_list_models, 182
spinetoolbox.mvcmodels.filter_checkbox_list_model, 184
spinetoolbox.mvcmodels.filter_execution_model, 186
spinetoolbox.mvcmodels.indexed_value_table_model, 187
spinetoolbox.mvcmodels.map_model, 188
spinetoolbox.mvcmodels.minimal_table_model, 193
spinetoolbox.mvcmodels.minimal_tree_model, 196
spinetoolbox.mvcmodels.project_item_model, 199
spinetoolbox.mvcmodels.project_item_specification_model, 203
spinetoolbox.mvcmodels.project_tree_item, 205
spinetoolbox.mvcmodels.resource_filter_model, 208
spinetoolbox.mvcmodels.shared, 210
spinetoolbox.mvcmodels.time_pattern_model, 210
spinetoolbox.mvcmodels.time_series_model_fixed_resolution, 211
spinetoolbox.mvcmodels.time_series_model_variable_resolution, 213
spinetoolbox.plotting, 545
spinetoolbox.plugin_manager, 555
spinetoolbox.project, 557
spinetoolbox.project_commands, 571
spinetoolbox.project_item, 215
spinetoolbox.project_item.logging_connection, 215
spinetoolbox.project_item.project_item, 219
spinetoolbox.project_item.project_item_factory, 226
spinetoolbox.project_item.specification_editor_window, 228
spinetoolbox.project_item_icon, 577
spinetoolbox.project_settings, 583
spinetoolbox.project_upgrader, 583
spinetoolbox.qthread_pool_executor, 589
spinetoolbox.server, 231
spinetoolbox.server.engine_client, 231
spinetoolbox.spine_db_commands, 590
spinetoolbox.spine_db_editor, 235
spinetoolbox.spine_db_editor.graphics_items, 394
spinetoolbox.spine_db_editor.main, 401
spinetoolbox.spine_db_editor.mvcmodels, 235
spinetoolbox.spine_db_editor.mvcmodels.alternative_item, 235
spinetoolbox.spine_db_editor.mvcmodels.alternative_model, 236
spinetoolbox.spine_db_editor.mvcmodels.colors, 237
spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_model, 237
spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_model, 244
spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item, 248
spinetoolbox.spine_db_editor.mvcmodels.entity_tree_models,

[spinetoolbox.spine_engine_worker](#), 624
[spinetoolbox.ui_main](#), 627
[spinetoolbox.version](#), 641
[spinetoolbox.widgets](#), 402
[spinetoolbox.widgets.about_widget](#), 402
[spinetoolbox.widgets.add_project_item_widget](#), 404
[spinetoolbox.widgets.add_up_spine_opt_wizard](#), 405
[spinetoolbox.widgets.array_editor](#), 408
[spinetoolbox.widgets.array_value_editor](#), 409
[spinetoolbox.widgets.code_text_edit](#), 409
[spinetoolbox.widgets.commit_dialog](#), 410
[spinetoolbox.widgets.custom_combobox](#), 411
[spinetoolbox.widgets.custom_delegates](#), 412
[spinetoolbox.widgets.custom_editors](#), 413
[spinetoolbox.widgets.custom_menus](#), 417
[spinetoolbox.widgets.custom_qcombobox](#), 420
[spinetoolbox.widgets.custom_qgraphicsscene](#), 421
[spinetoolbox.widgets.custom_qgraphicsviews](#), 423
[spinetoolbox.widgets.custom_qlineedit](#), 426
[spinetoolbox.widgets.custom_qtableview](#), 427
[spinetoolbox.widgets.custom_qtextbrowser](#), 432
[spinetoolbox.widgets.custom_qtreeview](#), 434
[spinetoolbox.widgets.custom_qwidgets](#), 435
[spinetoolbox.widgets.datetime_editor](#), 443
[spinetoolbox.widgets.duration_editor](#), 444
[spinetoolbox.widgets.indexed_value_table_context_menu](#), 444
[spinetoolbox.widgets.install_julia_wizard](#), 449
[spinetoolbox.widgets.jump_properties_widget](#), 451
[spinetoolbox.widgets.jupyter_console_widget](#), 453
[spinetoolbox.widgets.kernel_editor](#), 455
[spinetoolbox.widgets.link_properties_widget](#), 460
[spinetoolbox.widgets.map_editor](#), 461
[spinetoolbox.widgets.map_value_editor](#), 462
[spinetoolbox.widgets.multi_tab_spec_editor](#), 462
[spinetoolbox.widgets.multi_tab_window](#), 463
[spinetoolbox.widgets.notification](#), 469
[spinetoolbox.widgets.open_project_widget](#), 471
[spinetoolbox.widgets.parameter_value_editor](#), 474
[spinetoolbox.widgets.parameter_value_editor_base](#), 474
[spinetoolbox.widgets.persistent_console_widget](#), 476
[spinetoolbox.widgets.plain_parameter_value_editor](#), 481
[spinetoolbox.widgets.plot_canvas](#), 481
[spinetoolbox.widgets.plot_widget](#), 482
[spinetoolbox.widgets.plugin_manager_widgets](#), 484
[spinetoolbox.widgets.project_item_drag](#), 485
[spinetoolbox.widgets.properties_widget](#), 489
[spinetoolbox.widgets.report_plotting_failure](#), 490
[spinetoolbox.widgets.select_database_items](#), 490
[spinetoolbox.widgets.set_description_dialog](#), 492
[spinetoolbox.widgets.settings_widget](#), 492
[spinetoolbox.widgets.statusbars](#), 498
[spinetoolbox.widgets.time_pattern_editor](#), 498
[spinetoolbox.widgets.time_series_fixed_resolution_editor](#), 499
[spinetoolbox.widgets.time_series_variable_resolution_editor](#), 500
[spinetoolbox.widgets.toolbars](#), 501

Symbols

- `_` (in module `spinetoolbox.widgets.custom_qtableview`), 427
- `_ADD_TO_SELECTION_STR` (spinetoolbox.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel attribute), 184
- `_ALL_RUNS` (spinetoolbox.widgets.custom_qtextbrowser.CustomQTextBrowser attribute), 433
- `_ALL_RUNS` (spinetoolbox.widgets.statusbars.MainStatusBar attribute), 498
- `_ALTERNATIVE` (spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin attribute), 384
- `_ALTERNATIVE_ICON` (in module `spinetoolbox.spine_db_editor.mvcmodels.alternative_item`), 235
- `_ANIMATION_LIFE_SPAN` (spinetoolbox.widgets.notification.ChangeNotifier attribute), 471
- `_ARC_LENGTH_HINT` (spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin attribute), 359
- `_ARC_WIDTH` (spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin attribute), 359
- `_AffectedItemsFromOneTable` (class in `spinetoolbox.spine_db_editor.widgets.commit_viewer`), 320
- `_BASE_SETTINGS` (in module `spinetoolbox.plotting`), 547
- `_CHECK` (spinetoolbox.project_item_icon.ExecutionIcon attribute), 581
- `_CHUNK_SIZE` (in module `spinetoolbox.spine_db_worker`), 613
- `_CHUNK_SIZE` (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModelBase attribute), 285
- `_CLOCK` (spinetoolbox.project_item_icon.ExecutionIcon attribute), 581
- `_COLOR` (spinetoolbox.link.ConnectionLinkDrawer attribute), 540
- `_COLOR` (spinetoolbox.link.JumpLink attribute), 539
- `_COLOR` (spinetoolbox.link.JumpLinkDrawer attribute), 541
- `_COLOR` (spinetoolbox.link.Link attribute), 539
- `_COLOR` (spinetoolbox.link.LinkBase attribute), 536
- `_CROSS` (spinetoolbox.project_item_icon.ExecutionIcon attribute), 581
- `_ChoppedIcon` (class in `spinetoolbox.widgets.project_item_drag`), 487
- `_ChoppedIconEngine` (class in `spinetoolbox.widgets.project_item_drag`), 487
- `_CommitItem` (class in `spinetoolbox.spine_db_editor.widgets.commit_viewer`), 320
- `_CustomLineEdit` (class in `spinetoolbox.widgets.persistent_console_widget`), 476
- `_CustomLineEditDelegate` (class in `spinetoolbox.widgets.custom_editors`), 414
- `_CustomStatusBar` (class in `spinetoolbox.spine_db_editor.widgets.multi_spine_db_editor`), 370
- `__DATAPACKAGE` (spinetoolbox.link.Link attribute), 539
- `__DATA_ITEMS` (spinetoolbox.widgets.select_database_items.SelectDatabaseItems attribute), 491
- `_DBCommitViewer` (class in `spinetoolbox.spine_db_editor.widgets.commit_viewer`), 320
- `_DBListWidget` (class in `spinetoolbox.spine_db_editor.widgets.url_toolbar`), 393
- `_DUPLICATE_SCENARIO` (spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView attribute), 343
- `_EMPTY_STR` (spinetoolbox.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel attribute), 184
- `_FADE_IN_OUT_DURATION` (spinetoolbox.widgets.notification.Notification attribute), 469
- `_FEATURE_ICON` (in module `spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item`), 302

<code>_FILTERS</code> (<i>spinetoolbox.link.Link</i> attribute), 539	<code>_ITEM_NAME_KEY</code> (<i>spinetool-</i>
<code>_FILTER_TYPES</code> (<i>spinetool-</i>	<i>box.spine_db_editor.mvcmodels.item_metadata_table_model.Item</i>
<i>box.mvcmodels.resource_filter_model.ResourceFilterModel</i> attribute), 258	<i>attribute</i>), 208
<code>_FILTER_TYPE_TO_TEXT</code> (<i>spinetool-</i>	<i>box.spine_db_editor.mvcmodels.metadata_table_model.Metadata</i>
<i>box.mvcmodels.resource_filter_model.ResourceFilterModel</i> attribute), 260	<i>attribute</i>), 208
<code>_FLUSH_INTERVAL</code> (<i>spinetool-</i>	<i>box.spine_db_editor.mvcmodels.metadata_table_model_base.Met</i>
<i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i> attribute), 262	<i>attribute</i>), 477
<code>_FilterArrayWidget</code> (class in <i>spinetool-</i>	<i>box.spine_db_editor.mvcmodels.item_metadata_table_model.Item</i>
<i>box.spine_db_editor.widgets.url_toolbar</i>), 393	<i>attribute</i>), 258
<code>_FilterWidget</code> (class in <i>spinetool-</i>	<i>box.spine_db_editor.mvcmodels.metadata_table_model.Metadata</i>
<i>box.spine_db_editor.widgets.url_toolbar</i>), 393	<i>attribute</i>), 260
<code>_HEADER</code> (<i>spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase</i>	<i>box.spine_db_editor.mvcmodels.metadata_table_model_base.Met</i>
<i>attribute</i>), 262	<i>attribute</i>), 262
<code>_H_MARGIN</code> (<i>spinetoolbox.spine_db_editor.widgets.tabular_view_toolbar_widget.TabularViewToolbarWidget</i>	<i>box.spine_db_editor.widgets.tabular_view_toolbar_widget.TabularViewToolbarWidget</i>
<i>attribute</i>), 384	<i>attribute</i>), 384
<code>_INDEX</code> (<i>spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</i>	<i>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</i>
<i>attribute</i>), 385	<i>attribute</i>), 416
<code>_INDEX_EXPANSION</code> (<i>spinetool-</i>	<i>box.widgets.plugin_manager_widgets</i>), 485
<i>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</i> attribute), 384	<i>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</i> attribute), 384
<code>_INSERT_MULTIPLE_COLUMNS_AFTER</code> (in module <i>spinetool-</i>	<i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i>
<i>box.widgets.indexed_value_table_context_menu</i>), 445	<i>attribute</i>), 477
<code>_INSERT_MULTIPLE_COLUMNS_BEFORE</code> (in module <i>spinetool-</i>	<i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i>
<i>box.widgets.indexed_value_table_context_menu</i>), 446	<i>attribute</i>), 477
<code>_INSERT_MULTIPLE_ROWS_AFTER</code> (in module <i>spinetool-</i>	<i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i>
<i>box.widgets.indexed_value_table_context_menu</i>), 445	<i>attribute</i>), 477
<code>_INSERT_MULTIPLE_ROWS_BEFORE</code> (in module <i>spinetool-</i>	<i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i>
<i>box.widgets.indexed_value_table_context_menu</i>), 446	<i>attribute</i>), 477
<code>_INSERT_SINGLE_COLUMN_AFTER</code> (in module <i>spinetool-</i>	<i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i>
<i>box.widgets.indexed_value_table_context_menu</i>), 445	<i>attribute</i>), 477
<code>_INSERT_SINGLE_COLUMN_BEFORE</code> (in module <i>spinetool-</i>	<i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i>
<i>box.widgets.indexed_value_table_context_menu</i>), 445	<i>attribute</i>), 477
<code>_INSERT_SINGLE_ROW_AFTER</code> (in module <i>spinetool-</i>	<i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i>
<i>box.widgets.indexed_value_table_context_menu</i>), 445	<i>attribute</i>), 477
<code>_INSERT_SINGLE_ROW_BEFORE</code> (in module <i>spinetool-</i>	<i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i>
<i>box.widgets.indexed_value_table_context_menu</i>), 446	<i>attribute</i>), 477
<code>_ISSUE</code> (<i>spinetoolbox.link.JumpLink</i> attribute), 539	<i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i> attribute), 477
	<code>_MEMORY</code> (<i>spinetoolbox.link.Link</i> attribute), 539
	<code>_METHOD_ICON</code> (in module <i>spinetool-</i>
	<i>box.spine_db_editor.mvcmodels.tool_feature_item</i>), 302
	<code>_ManagePluginsModel</code> (class in <i>spinetool-</i>
	<i>box.widgets.plugin_manager_widgets</i>), 485
	<code>_MenuToolBar</code> (class in <i>spinetool-</i>
	<i>box.widgets.custom_qwidgets</i>), 439
	<code>_NOT_TIME_STAMP</code> (in module <i>spinetool-</i>
	<i>box.widgets.custom_qtableview</i>), 432
	<code>_OPEN_EDITOR</code> (in module <i>spinetool-</i>
	<i>box.widgets.indexed_value_table_context_menu</i>), 446
	<code>_PARAMETER</code> (<i>spinetool-</i>
	<i>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMi</i>
	<i>attribute</i>), 384
	<code>_PARAMETER_VALUE</code> (<i>spinetool-</i>
	<i>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMi</i>
	<i>attribute</i>), 384
	<code>_PLOT</code> (in module <i>spinetool-</i>

`box.widgets.indexed_value_table_context_menu`), `_SCENARIO_ITEMS` (`spinetool-`
446 `box.widgets.select_database_items.SelectDatabaseItems`
`_PLOT_IN_WINDOW` (in module `spinetool-`
`box.widgets.indexed_value_table_context_menu`), `_SELECTORS` (in module `spinetool-`
446 `box.widgets.parameter_value_editor_base`),
`_PURGE` (`spinetoolbox.link.Link` attribute), 539 475
`_PageId` (class in `spinetool-`
`box.widgets.add_up_spine_opt_wizard`), 405
`_PageId` (class in `spinetool-`
`box.widgets.install_julia_wizard`), 450
`_PlotDataView` (class in `spinetool-`
`box.widgets.plot_widget`), 484
`_PlotDataWidget` (class in `spinetool-`
`box.widgets.plot_widget`), 484
`_PlotStackedBars` (class in `spinetoolbox.plotting`), 550
`_PluginWorker` (class in `spinetoolbox.plugin_manager`),
556
`_QDateTime_to_datetime()` (in module `spinetool-`
`box.widgets.datetime_editor`), 443
`_RELATIONSHIP` (`spinetool-`
`box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin`), 384
attribute), 384
`_REMOVE_ALTERNATIVE` (`spinetool-`
`box.spine_db_editor.widgets.custom_qtableview.PivotTableWidget`
attribute), 343
`_REMOVE_COLUMNS` (in module `spinetool-`
`box.widgets.indexed_value_table_context_menu`), 446
`_REMOVE_OBJECT` (`spinetool-`
`box.spine_db_editor.widgets.custom_qtableview.PivotTableWidget`
attribute), 343
`_REMOVE_PARAMETER` (`spinetool-`
`box.spine_db_editor.widgets.custom_qtableview.PivotTableWidget`
attribute), 343
`_REMOVE_RELATIONSHIP` (`spinetool-`
`box.spine_db_editor.widgets.custom_qtableview.PivotTableWidget`
attribute), 343
`_REMOVE_ROWS` (in module `spinetool-`
`box.widgets.indexed_value_table_context_menu`), 446
`_REMOVE_SCENARIO` (`spinetool-`
`box.spine_db_editor.widgets.custom_qtableview.PivotTableWidget`
attribute), 343
`_SCATTER_LINE_PLOT_SETTINGS` (in module `spinetool-`
`box.plotting`), 547
`_SCATTER_PLOT_SETTINGS` (in module `spinetool-`
`box.plotting`), 547
`_SCENARIO_ALTERNATIVE` (`spinetool-`
`box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin`), 384
attribute), 384
`_SCENARIO_ICON` (in module `spinetool-`
`box.spine_db_editor.mvcmodels.scenario_item`),
294

`box.widgets.toolbars.MainToolBar` attribute),
502
`_SKIP` (`spinetoolbox.project_item_icon.ExecutionIcon` at-
tribute), 581
`_SPACING` (`spinetoolbox.spine_db_editor.widgets.tabular_view_header_widget`), 384
`_ScenarioNameResolution` (class in `spinetool-`
`box.spine_db_editor.widgets.scenario_generator`),
375
`_SceneSvgRenderer` (class in `spinetool-`
`box.spine_db_icon_manager`), 595
`_SelectDatabases` (class in `spinetool-`
`box.spine_db_editor.widgets.mass_select_items_dialogs`),
365
`_SpecificationEditorToolBar` (class in `spinetool-`
`box.project_item.specification_editor_window`),
230
`_SymbolIcon` (class in `spinetoolbox.link`), 537
`_TIME_SERIES_PLOT_SETTINGS` (in module `spinetool-`
`box.plotting`), 547
`_TOOL_ICON` (in module `spinetool-`
`box.spine_db_editor.mvcmodels.tool_feature_item`),
302
`_TRIM_COLUMNS` (in module `spinetool-`
`box.widgets.indexed_value_table_context_menu`),
446
`_TYPE_LABEL` (`spinetool-`
`box.spine_db_editor.widgets.scenario_generator.ScenarioGenerator`
attribute), 375
`_TextIcon` (class in `spinetoolbox.link`), 537
`_UrlFilterDialog` (class in `spinetool-`
`box.spine_db_editor.widgets.url_toolbar`),
393
`_WARNING` (`spinetoolbox.link.Link` attribute), 539
`_WarningTextIcon` (class in `spinetoolbox.link`), 537
`__call__()` (`spinetoolbox.helpers.QuietLogger`
method), 527
`__call__()` (`spinetoolbox.plotting._PlotStackedBars`

method), 551
 __get__() (spinetoolbox.widgets.custom_qwidgets.QWizardProcessPageAddEntityTab method), 440
 __getattr__() (spinetoolbox.helpers.QuietLogger method), 527
 __hash__() (spinetoolbox.project_item.logging_connection.LoggingConnection method), 217
 __lt__() (spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.item_metadata_table_model.ItemMetadataTableModel method), 298
 __set__() (spinetoolbox.widgets.custom_qwidgets.QWizardProcessPageAddEntityTab method), 440
 __set_name__() (spinetoolbox.widgets.custom_qwidgets.QWizardProcessPageAddEntityTab method), 440
 __str__() (spinetoolbox.fetch_parent.ItemTypeFetchParent method), 510
 __str__() (spinetoolbox.version.VersionInfo method), 641
 __version__ (in module spinetoolbox), 642
 __version__ (in module spinetoolbox.version), 642
 __version_info__ (in module spinetoolbox), 642
 __version_info__ (in module spinetoolbox.version), 641
 _accepts_entity_group_item() (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.ObjectClassMixin method), 250
 _accepts_entity_metadata_item() (spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel method), 258
 _accepts_item() (spinetoolbox.spine_db_editor.widgets.custom_menus.ParameterViewFillMenu method), 334
 _accepts_object_item() (spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin method), 359
 _accepts_parameter_value_metadata_item() (spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel method), 258
 _accepts_relationship_item() (spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin method), 359
 _activate_properties_tab() (spinetoolbox.ui_main.ToolboxUI method), 631
 _add_args() (spinetoolbox.widgets.jump_properties_widget.JumpPropertiesWidget method), 452
 _add_arrow_path() (spinetoolbox.link.LinkBase method), 536
 _add_column_to_plot() (spinetoolbox.spine_db_editor.widgets.pivot_table_header_value_pivot_header_view.PivotHeaderView method), 374
 _add_command_name (spinetoolbox.spine_db_commands.AddItemCommand attribute), 592
 _add_data() (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase method), 264
 _add_data_to_db_mgr() (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel method), 260
 _add_data_to_db_mgr() (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase method), 263
 _add_default_actions() (spinetoolbox.widgets.indexed_value_table_context_menu.ContextMenuBase method), 446
 _add_ellipse_path() (spinetoolbox.link.LinkBase method), 536
 _add_entities_on_the_fly (spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModels attribute), 247
 _add_entities_on_the_fly (spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInEntityIds attribute), 274
 _add_execute_actions() (spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel method), 633
 _add_filling() (spinetoolbox.widgets.project_item_drag.ProjectItemSpecArray method), 488
 _add_item() (spinetoolbox.spine_db_worker.SpineDBWorker method), 151
 _add_leaf_item() (spinetoolbox.mvcmodels.project_item_model.ProjectItemModel method), 196
 _add_line() (spinetoolbox.widgets.custom_qwidgets.TitleWidgetAction method), 440
 _add_middle_actions() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349
 _add_middle_actions() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView method), 350
 _add_middle_actions() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.RelationshipTreeView method), 351
 _add_msg() (spinetoolbox.widgets.custom_qwidgets.QWizardProcessPageAddEntityTab method), 441
 _add_msg_error() (spinetoolbox.spine_db_commands.AddItemCommand attribute), 592

655

<code>method)</code> , 379	<code>box.widgets.array_editor.ArrayEditor</code> <code>method)</code> , 408
<code>_build_auto_filter()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_menus.ParameterValueEditor</code> <code>method)</code> , 334	<code>box.fetch_parent.FetchParent</code> <code>attribute)</code> , 507
<code>_can_build_pivot_table()</code> (<code>spinetool-box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</code> <code>method)</code> , 388	<code>box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel</code> <code>method)</code> , 184
<code>_can_fetch_members_item()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.entity_tree_item.EntityTreeItem</code> <code>method)</code> , 252	<code>box.project_item.logging_connection.LoggingConnection</code> <code>method)</code> , 219
<code>_can_fetch_more_entity_groups()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.entity_tree_item.EntityTreeItem</code> <code>method)</code> , 250	<code>box.server.engine_client.EngineClient</code> <code>method)</code> , 232
<code>_can_remove_relationships()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_qtableview.PivotTableView</code> <code>method)</code> , 344	<code>box.spine_db_editor.widgets.scenario_generator.ScenarioGenerator</code> <code>method)</code> , 376
<code>_cancel_filter()</code> (<code>spinetool-box.widgets.custom_qwidgets.FilterWidget</code> <code>method)</code> , 437	<code>_check_filter()</code> (<code>spinetool-box.widgets.custom_menus.FilterMenuBase</code> <code>method)</code> , 420
<code>_carry_splitter_state()</code> (<code>spinetool-box.spine_db_editor.widgets.commit_viewer.CommitViewer</code> <code>method)</code> , 321	<code>_check_frozen_value_selected()</code> (<code>spinetool-box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</code> <code>method)</code> , 389
<code>_center_scene()</code> (in module <code>spinetool-box.spine_db_icon_manager</code>), 595	<code>_check_if_plotting_enabled()</code> (<code>spinetool-box.widgets.array_editor.ArrayEditor</code> <code>method)</code> , 408
<code>_change_condition()</code> (<code>spinetool-box.widgets.jump_properties_widget.JumpPropertiesWidget</code> <code>method)</code> , 452	<code>_check_item()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel</code> <code>method)</code> , 246
<code>_change_context()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_qtableview.PivotTableView</code> <code>method)</code> , 346	<code>_check_item()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel</code> <code>method)</code> , 247
<code>_change_datetime()</code> (<code>spinetool-box.widgets.datetime_editor.DatetimeEditor</code> <code>method)</code> , 443	<code>_check_notifications()</code> (<code>spinetool-box.project_item.project_item.ProjectItem</code> <code>method)</code> , 221
<code>_change_duration()</code> (<code>spinetool-box.widgets.duration_editor.DurationEditor</code> <code>method)</code> , 444	<code>_check_pivot()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.pivot_model.PivotModel</code> <code>method)</code> , 280
<code>_change_filter()</code> (<code>spinetool-box.widgets.custom_menus.FilterMenuBase</code> <code>method)</code> , 420	<code>_check_project_version()</code> (<code>spinetool-box.headless.ActionsWithProject</code> <code>method)</code> , 514
<code>_change_filter_checked_state()</code> (<code>spinetool-box.mvcmodels.resource_filter_model.ResourceFilterModel</code> <code>method)</code> , 209	<code>_check_validity()</code> (<code>spinetool-box.spine_db_editor.widgets.add_items_dialogs.AddObjectGroupDialog</code> <code>method)</code> , 319
<code>_change_frozen_value()</code> (<code>spinetool-box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</code> <code>method)</code> , 390	<code>_check_validity()</code> (<code>spinetool-box.spine_db_editor.widgets.add_items_dialogs.ObjectGroupDialog</code> <code>method)</code> , 319
<code>_change_parameter_type()</code> (<code>spinetool-box.widgets.parameter_value_editor_base.ParameterValueEditorBase</code> <code>method)</code> , 475	<code>_children_sort_key</code> (<code>spinetool-box.spine_db_editor.mvcmodels.entity_tree_item.ObjectClassItem</code> <code>property)</code> , 250
<code>_change_selected_frozen_row()</code> (<code>spinetool-box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</code> <code>method)</code> , 389	<code>_children_sort_key</code> (<code>spinetool-box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem</code> <code>property)</code> , 267
<code>_change_specification_file_location()</code> (<code>spinetoolbox.ui_main.ToolboxUI</code> <code>method)</code> , 632	<code>_children_sort_key()</code> (<code>spinetool-</code>
<code>_change_value_type()</code> (<code>spinetool-</code>	

<code>box.spine_db_editor.mvcmodels.parameter_value_list_item.ListItem</code> method), 278	<code>box.spine_db_editor.widgets.custom_delegates.ParameterDelegates</code> method), 325
<code>_children_sort_key()</code> (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.SortChildrenMixin) method), 309	<code>_close_editor()</code> (spinetoolbox.spine_db_editor.widgets.custom_delegates.ParameterValueListCompoundParameterModel) method), 239
<code>_class_filter_accepts_model()</code> (spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel) method), 239	<code>_close_editor()</code> (spinetoolbox.spine_db_editor.widgets.custom_delegates.ScenarioDelegate) method), 330
<code>_clean_up_export_items_dialog()</code> (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor) method), 380	<code>_close_editor()</code> (spinetoolbox.spine_db_editor.widgets.custom_delegates.ToolFeatureDelegate) method), 337
<code>_clean_up_heat_map_items()</code> (spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityGraphicView) method), 337	<code>_close_editor()</code> (spinetoolbox.spine_db_editor.widgets.select_position_parameters_dialog.PositionParametersDialog) method), 377
<code>_clean_up_purge_items_dialog()</code> (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor) method), 381	<code>_close_enough()</code> (spinetoolbox.link.LinkBase) method), 537
<code>_clean_up_purge_settings_dialog()</code> (spinetoolbox.widgets.link_properties_widget.LinkPropertiesWidget) method), 461	<code>_close_tab()</code> (spinetoolbox.widgets.multi_tab_window.MultiTabWindow) method), 467
<code>_clean_up_worker()</code> (spinetoolbox.plugin_manager.PluginManager) method), 556	<code>_close_waiting_for_fetcher()</code> (spinetoolbox.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDBEditor) method), 370
<code>_cleanup_jupyter_console()</code> (spinetoolbox.ui_main.ToolboxUI) method), 640	<code>_closest_connector()</code> (spinetoolbox.project_item_icon.ProjectItemIcon) method), 579
<code>_clear_all_other_selections()</code> (spinetoolbox.spine_db_editor.widgets.parameter_view_mixin.ParameterViewMixin) method), 372	<code>_collapse()</code> (spinetoolbox.spine_db_editor.graphics_items.ObjectItem) method), 398
<code>_clear_filter()</code> (spinetoolbox.widgets.custom_menus.FilterMenuBase) method), 420	<code>_collect_more_columns()</code> (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel) method), 285
<code>_clear_layout()</code> (in module spinetoolbox.widgets.add_up_spine_opt_wizard), 408	<code>_collect_more_data()</code> (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel) method), 285
<code>_clear_plot()</code> (in module spinetoolbox.plotting), 551	<code>_collect_more_data()</code> (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel) method), 285
<code>_clear_selection_lists()</code> (spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView) method), 343	<code>_collect_more_data()</code> (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel) method), 285
<code>_clear_selection_lists()</code> (spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView) method), 344	<code>_collect_more_data()</code> (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel) method), 288
<code>_clear_selection_lists()</code> (spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView) method), 344	<code>_collect_more_data()</code> (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel) method), 374
<code>_clear_selection_lists()</code> (spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView) method), 345	<code>_collect_more_data()</code> (spinetoolbox.spine_db_editor.widgets.pivot_table_header_view.ParameterValueContext) method), 373
<code>_close_consoles()</code> (spinetoolbox.ui_main.ToolboxUI) method), 640	<code>_command_checked</code> (spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget) attribute), 477
<code>_close_db_map()</code> (spinetoolbox.spine_db_worker.SpineDBWorker) method), 616	<code>_command_finished</code> (spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget) attribute), 477
<code>_close_editor()</code> (spinetoolbox.spine_db_editor.widgets.custom_delegates.AlternativeDataView) method), 477	

<code>_commit_session()</code> (<i>spinetoolbox.spine_db_worker.SpineDBWorker</i> method), 617	<code>_connect_tab_signals()</code> (<i>spinetoolbox.widgets.multi_tab_window.MultiTabWindow</i> method), 466
<code>_complete_graph()</code> (<i>spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin</i> method), 361	<code>_constructor_args_from_dict()</code> (<i>spinetoolbox.project_item.logging_connection.HeadlessConnection</i> static method), 216
<code>_completions_available</code> (<i>spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget</i> attribute), 477	<code>_context_menu_make()</code> (<i>spinetoolbox.widgets.jupyter_console_widget.JupyterConsoleWidget</i> method), 454
<code>_compute_max_zoom()</code> (<i>spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView</i> method), 338	<code>_convert_leaves()</code> (<i>spinetoolbox.widgets.kinmap_editor.MapEditor</i> method), 461
<code>_compute_max_zoom()</code> (<i>spinetoolbox.widgets.custom_qgraphicsviews.CustomQGraphicsView</i> method), 424	<code>_convert_legacy_resource_filter_ids_to_filter_settings()</code> (<i>spinetoolbox.project_item.logging_connection.HeadlessConnection</i> method), 216
<code>_compute_max_zoom()</code> (<i>spinetoolbox.widgets.custom_qgraphicsviews.DesignQGraphicsView</i> method), 425	<code>_convert_to_data_type()</code> (<i>spinetoolbox.mvcmodels.array_model.ArrayModel</i> method), 176
<code>_confirm_exit()</code> (<i>spinetoolbox.ui_main.ToolboxUI</i> method), 636	<code>_convert_to_db()</code> (<i>spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.ConvertToDBMixin</i> method), 271
<code>_confirm_project_close()</code> (<i>spinetoolbox.ui_main.ToolboxUI</i> method), 636	<code>_convert_to_db()</code> (<i>spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInAlternativesMixin</i> method), 271
<code>_connect_log_signals()</code> (<i>spinetoolbox.spine_engine_worker.SpineEngineWorker</i> method), 626	<code>_convert_to_db()</code> (<i>spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInEntityClassMixin</i> method), 271
<code>_connect_pivot_table_header_signals()</code> (<i>spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</i> method), 385	<code>_convert_to_db()</code> (<i>spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInEntityIdsMixin</i> method), 274
<code>_connect_project_item_model_signals()</code> (<i>spinetoolbox.spine_db_editor.widgets.url_toolbar.UrlToolBar</i> method), 392	<code>_convert_to_db()</code> (<i>spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInParameterValuesMixin</i> method), 275
<code>_connect_project_signals()</code> (<i>spinetoolbox.ui_main.ToolboxUI</i> method), 638	<code>_convert_to_db()</code> (<i>spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInParameterValuesMixin</i> method), 272
<code>_connect_signals()</code> (<i>spinetoolbox.project_item.project_item.ProjectItem</i> method), 221	<code>_convert_to_db()</code> (<i>spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInValueListMixin</i> method), 272
<code>_connect_signals()</code> (<i>spinetoolbox.spine_db_manager.SpineDBManager</i> method), 597	<code>_convert_to_db()</code> (<i>spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.ImposeEntityClassMixin</i> method), 275
<code>_connect_signals()</code> (<i>spinetoolbox.widgets.custom_qwidgets.QWizardProcessPage</i> method), 441	<code>_convert_to_db()</code> (<i>spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.InferEntityClassMixin</i> method), 275
<code>_connect_single_model()</code> (<i>spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel</i> method), 180	<code>_convert_to_leaf()</code> (in module <i>spinetoolbox.plotting</i>), 554
<code>_connect_tab()</code> (<i>spinetoolbox.widgets.multi_tab_window.MultiTabWindow</i> method), 466	<code>_could_be_time_stamp()</code> (in module <i>spinetoolbox.widgets.custom_qtableview</i>), 432
<code>_connect_tab_signals()</code> (<i>spinetoolbox.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDBEditor</i> method), 368	<code>_create_spine_db_renderer()</code> (<i>spinetoolbox.spine_db_icon_manager.SpineDBIconManager</i> method), 595
<code>_connect_tab_signals()</code> (<i>spinetoolbox.widgets.multi_tab_spec_editor.MultiTabSpecEditor</i> method), 463	<code>_create_context_menu()</code> (<i>spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView</i> method), 463

method), 349

`_create_database_editor()` (spinetool-
box.spine_db_editor.widgets.custom_delegates.ManageItemsDialog), 289

method), 332

`_create_empty_model()` (spinetool-
box.mvcmodels.compound_table_model.CompoundTableModel), 180

method), 180

`_create_empty_model()` (spinetool-
box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel), 239

method), 239

`_create_group_renderer()` (spinetool-
box.spine_db_icon_manager.SpineDBIconManager), 595

method), 595

`_create_icon_renderer()` (spinetool-
box.spine_db_icon_manager.SpineDBIconManager), 595

method), 595

`_create_new_children()` (spinetool-
box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem), 264

method), 268

`_create_or_request_parameter_value_editor()`
(spinetoolbox.spine_db_editor.widgets.custom_delegates.ParameterValueOrDefaultValueDelegate), 326

method), 326

`_create_plugin_widget()` (spinetool-
box.widgets.plugin_manager_widgets.ManagePluginsDialog), 485

method), 485

`_create_project_structure()` (spinetool-
box.project.SpineToolboxProject), 559

method), 559

`_create_rel_cls_renderer()` (spinetool-
box.spine_db_icon_manager.SpineDBIconManager), 595

method), 595

`_create_single_model()` (spinetool-
box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel), 241

method), 241

`_create_single_model()` (spinetool-
box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel), 243

method), 243

`_create_worker()` (spinetool-
box.plugin_manager.PluginManager), 556

method), 556

`_cross_hairs_has_valid_target()` (spinetool-
box.spine_db_editor.widgets.custom_qgraphicsviews.EntityGraphicsView), 337

method), 337

`_dag_execution_started` (spinetool-
box.spine_engine_worker.SpineEngineWorker
attribute), 625

`_dag_iterator()` (spinetool-
box.project.SpineToolboxProject), 565

method), 565

`_dags()` (spinetoolbox.headless.ActionsWithProject), 514

method), 514

`_data()` (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.RelationshipPivotTableModel), 291

method), 291

`_data()` (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.RelationshipPivotTableModel), 291

method), 291

method), 290

`_data()` (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.RelationshipPivotTableModel), 292

method), 292

`_data_length()` (in module spinetool-
box.mvcmodels.compound_parameter_model.CompoundParameterModel), 292

method), 293

`_database_table_name()` (spinetool-
box.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel), 259

method), 259

`_database_table_name()` (spinetool-
box.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel), 260

method), 260

`_database_table_name()` (spinetool-
box.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase), 264

method), 264

`_datetime_to_QDateTime()` (in module spinetool-
box.widgets.datetime_editor), 443

method), 443

`_db_create_item()` (spinetool-
box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel), 299

method), 299

`_db_map_alt_ids_from_selection()` (spinetool-
box.spine_db_editor.widgets.custom_qtreeview.AlternativeTreeView), 352

method), 352

`_db_map_alternative_ids_from_selection()`
(spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ScenarioTreeView), 353

method), 353

`_db_map_class_ids()` (spinetool-
box.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin), 391

method), 391

`_db_map_ids()` (spinetool-
box.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin), 391

method), 391

`_db_map_ids()` (spinetool-
box.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin), 391

method), 391

`_db_map_ids()` (spinetool-
box.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin), 391

method), 391

`_db_map_lock()` (in module spinetool-
box.spine_db_worker), 613

method), 613

`_db_map_object_ids()` (spinetool-
box.spine_db_editor.mvcmodels.pivot_table_models.ParameterValuePivotTableModel), 290

method), 290

`_dec_value()` (spinetool-
box.widgets.custom_qwidgets.HorizontalSpinBox), 441

method), 441

`_deep_refresh_children()` (spinetool-
box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem), 267

method), 267

`_default_pivot()` (spinetool-
box.spine_db_editor.mvcmodels.pivot_table_models.ParameterValuePivotTableModel), 290

method), 290

`_default_pivot()` (spinetool-
box.spine_db_editor.mvcmodels.pivot_table_models.RelationshipPivotTableModel), 291

method), 291

method), 292

`_default_pivot()` (spinetool-
box.spine_db_editor.mvcmodels.pivot_table_models.ScenarioAlternativePivotTableModels.pivot_table_models.ParameterVa
method), 293

`_default_specification_file_path()` (spine-
toolbox.project.SpineToolboxProject method),
562

`_deserialize_items()` (spinetool-
box.ui_main.ToolboxUI method), 637

`_deserialized_item_position_shifts()` (spine-
toolbox.ui_main.ToolboxUI method), 637

`_dict_intersection()` (spinetool-
box.spine_db_editor.widgets.parameter_view_mixin.ParameterViewMixin
static method), 372

`_disable_project_actions()` (spinetool-
box.ui_main.ToolboxUI method), 629

`_disconnect_project_item_model_signals()`
(spinetoolbox.spine_db_editor.widgets.url_toolbar.UrlToolBar
method), 392

`_disconnect_signals()` (spinetool-
box.project_item.project_item.ProjectItem
method), 221

`_disconnect_tab_signals()` (spinetool-
box.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDBEditor
method), 369

`_disconnect_tab_signals()` (spinetool-
box.widgets.multi_tab_spec_editor.MultiTabSpecEditor
method), 463

`_disconnect_tab_signals()` (spinetool-
box.widgets.multi_tab_window.MultiTabWindow
method), 466

`_display_completions()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget
method), 480

`_display_history_item()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget
method), 480

`_display_icon()` (spinetool-
box.spine_db_editor.mvcmodels.entity_tree_item.EntityClassItem
method), 249

`_display_icon()` (spinetool-
box.spine_db_editor.mvcmodels.entity_tree_item.MembersItem
method), 251

`_do_add_items()` (spinetool-
box.mvcmodels.filter_checkbox_list_model.LazyFilterCheckboxListModel
method), 185

`_do_add_items()` (spinetool-
box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel
method), 185

`_do_advance_query()` (spinetool-
box.spine_db_worker.SpineDBWorker method),
614

`_do_autocomplete()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget
method), 480

method), 480

`_do_batch_set_inner_data()` (spinetool-
box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel
method), 290

`_do_batch_set_inner_data()` (spinetool-
box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel
method), 289

`_do_batch_set_inner_data()` (spinetool-
box.spine_db_editor.mvcmodels.pivot_table_models.Relationship
method), 292

`_do_batch_set_inner_data()` (spinetool-
box.spine_db_editor.mvcmodels.pivot_table_models.ScenarioAlternativePivotTableModels.pivot_table_models.ParameterVa
method), 293

`_do_check_command()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget
method), 479

`_do_clone()` (spinetool-
box.spine_db_commands.AddItemCommand
method), 592

`_do_clone()` (spinetool-
box.spine_db_commands.AgedUndoCommand
method), 591

`_do_clone()` (spinetool-
box.spine_db_commands.RemoveItemsCommand
method), 594

`_do_clone()` (spinetool-
box.spine_db_commands.SpineDBMacro
method), 591

`_do_clone()` (spinetool-
box.spine_db_commands.UpdateItemsCommand
method), 593

`_do_find_next_relationship()` (spinetool-
box.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView
method), 350

`_do_get_db_map()` (spinetool-
box.spine_db_manager.SpineDBManager
method), 599

`_do_handle_event_msg()` (spinetool-
box.spine_engine_worker.SpineEngineWorker
method), 627

`_do_handle_node_execution_finished()` (spine-
toolbox.spine_engine_worker.SpineEngineWorker
method), 627

`_do_handle_node_execution_started()` (spinetool-
box.spine_engine_worker.SpineEngineWorker
method), 627

`_do_handle_process_msg()` (spinetool-
box.spine_engine_worker.SpineEngineWorker
method), 627

`_do_insert_stdin_text()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget
method), 478

`_do_interrupt_persistent()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget
method), 478

method), 480

`_do_issue_command()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget method), 197

method), 479

`_do_kill_persistent()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget method), 277

method), 480

`_do_make_kernel()` (spinetool-
box.widgets.kernel_editor.KernelEditorBase method), 294

method), 455

`_do_make_kernel()` (spinetool-
box.widgets.kernel_editor.MinijuliaKernelEditor method), 304

method), 459

`_do_make_kernel()` (spinetool-
box.widgets.kernel_editor.MinipythonKernelEditor method), 304

method), 459

`_do_make_pixmap()` (spinetool-
box.helpers.ColoredIconEngine method), 309

524

`_do_move_history()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget method), 556

method), 480

`_do_override_console()` (spinetool-
box.ui_main.ToolboxUI method), 634

`_do_paint()` (spinetool-
box.widgets.custom_delegates.CheckBoxDelegate static method), 412

`_do_paint()` (spinetool-
box.widgets.custom_delegates.RankDelegate static method), 413

`_do_refresh()` (spinetool-
box.mvcmodels.compound_table_model.CompoundTableModel method), 178

`_do_resize()` (spinetool-
box.spine_db_editor.widgets.custom_qtableview.PidTableWidget method), 340

`_do_resize()` (spinetool-
box.spine_db_editor.widgets.custom_qtableview.PidTableWidget method), 346

`_do_resize()` (spinetool-
box.spine_db_editor.widgets.custom_qtreeview.ResizableTreeView method), 348

`_do_resize()` (spinetool-
box.widgets.custom_qwidgets.ResizingViewMixin method), 442

`_do_restart_persistent()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget method), 480

`_do_select_commit()` (spinetool-
box.spine_db_editor.widgets.commit_viewer.DBCommitViewer method), 320

`_do_set_killed()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget method), 479

`_do_set_up()` (spinetool-
box.mvcmodels.minimal_tree_model.TreeItem method), 197

`_do_set_up()` (spinetool-
box.spine_db_editor.mvcmodels.parameter_value_list_item.ListItem method), 277

`_do_set_up()` (spinetool-
box.spine_db_editor.mvcmodels.scenario_item.ScenarioItem method), 294

`_do_set_up()` (spinetool-
box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 304

`_do_set_up()` (spinetool-
box.spine_db_editor.mvcmodels.tool_feature_item.ToolLeafItem method), 304

`_do_set_up()` (spinetool-
box.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildMixin method), 309

`_do_show_install_plugin_dialog()` (spinetool-
box.plugin_manager.PluginManager method), 556

`_do_show_manage_plugins_dialog()` (spinetool-
box.plugin_manager.PluginManager method), 556

`_do_update_add_args_button_enabled()` (spine-
toolbox.widgets.jump_properties_widget.JumpPropertiesWidget method), 452

`_do_update_geometry()` (spinetool-
box.link.JumpOrLink method), 538

`_do_update_geometry()` (spinetoolbox.link.LinkBase method), 536

`_do_update_parents_will_have_children()` (spinetoolbox.spine_db_worker.SpineDBWorker method), 615

`_do_update_path()` (spinetool-
box.project_item_icon.ProjectItemIcon method), 578

`_do_update_remove_args_button_enabled()` (spinetoolbox.widgets.jump_properties_widget.JumpPropertiesWidget method), 452

`_do_work()` (spinetool-
box.plugin_manager._PluginWorker method), 557

`_do_work()` (spinetool-
box.qthread_pool_executor.QtBasedThreadPoolExecutor method), 590

`_download_file()` (in module spinetool-
box.plugin_manager), 555

`_download_plugin()` (in module spinetool-
box.plugin_manager), 555

`_draw_grid_bg()` (spinetool-
box.widgets.custom_qgraphicsscene.DesignGraphicsScene method), 422

`_draw_solid_bg()` (spinetool-

static method), 259

`_extra_cells_from_added_item()` (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModelBase static method), 261

`_extra_cells_from_added_item()` (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase static method), 264

`_fetch_all()` (spinetoolbox.spine_db_worker.SpineDBWorker method), 616

`_fetch_members_item()` (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.EntityItem method), 252

`_fetch_more_entity_groups()` (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.ObjectClassItem method), 250

`_fetch_more_if_possible()` (spinetoolbox.project_item.logging_connection.LoggingConnection method), 217

`_fetch_more_visible()` (spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView method), 346

`_fetch_more_visible()` (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349

`_fetch_parents()` (spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel method), 258

`_fetch_parents()` (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel method), 261

`_fetch_parents()` (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase method), 263

`_fetch_parents()` (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ParameterViewPivotTableModel method), 290

`_fetch_parents()` (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModelBase method), 285

`_fetch_parents()` (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.RelationshipPivotTableModel method), 292

`_fetch_parents()` (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ScenarioLinkPivotTableModel method), 292

`_fetch_parents()` (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EntityItemUtility method), 309

`_fill_in_entity_class_id()` (spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInEntityClassIdMixin method), 273

`_fill_in_entity_ids()` (spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInEntityIdsMixin method), 274

`_fill_in_parameter_ids()` (spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInParameterIdsMixin method), 274

`_fill_in_value_list_id()` (spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins.FillInValueListIdMixin method), 272

`_filter_accepts_row()` (spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel method), 299

`_filter_alternative_ids` (spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel attribute), 301

`_filter_consoles` (spinetoolbox.spine_db_editor.mvcmodels.filter_execution_model.FilterExecutionModel attribute), 186

`_filter_db_map_class_entity_ids` (spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel attribute), 301

`_filter_entity_ids` (spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel attribute), 301

`_filter_list()` (spinetoolbox.widgets.custom_qwidgets.FilterWidget method), 437

`_filter_model()` (spinetoolbox.spine_db_editor.widgets.install_plugin_dialog.InstallPluginDialog method), 485

`_finalize_editing()` (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase method), 412

`_find_base_alternative()` (in module spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase), 376

`_find_db_map()` (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase method), 265

`_find_filter_type_item()` (spinetoolbox.spine_db_editor.mvcmodels.resource_filter_model.ResourceFilterModel method), 209

`_find_first()` (spinetoolbox.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel method), 256

`_find_module_material()` (in module spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ScenarioLinkPivotTableModel), 292

`_find_new_point()` (spinetoolbox.link.LinkBase method), 537

`_find_uniformed_rows_by_id()` (spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem method), 269

`_fill_in_entity_class_id()` (spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene method), 421

`_fill_in_entity_ids()` (spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene method), 421

<code>box.widgets.indexed_value_table_context_menu.MapTableContextMenu</code>	<code>box.spine_db_manager.SpineDBManager</code>
<code>method), 448</code>	<code>method), 610</code>
<code>_first_row()</code>	<code>(spinetool- _get_db_map()</code>
<code>box.widgets.indexed_value_table_context_menu.ContextMenuBase</code>	<code>box.project_item.logging_connection.LoggingConnection</code>
<code>method), 446</code>	<code>method), 217</code>
<code>_fix_1d_array_to_array()</code> (in module <code>spinetool-</code>	<code>_get_db_map()</code>
<code>box.project_upgrader), 589</code>	<code>box.spine_db_editor.widgets.custom_delegates.ParameterDelegat</code>
<code>_flash_arrived</code>	<code>method), 325</code>
<code>(spinetool-</code>	<code>_get_db_map()</code>
<code>box.spine_engine_worker.SpineEngineWorker</code>	<code>(spinetool-</code>
<code>attribute), 625</code>	<code>box.spine_db_worker.SpineDBWorker method),</code>
<code>_flush_needed</code>	<code>614</code>
<code>(spinetool-</code>	<code>get_db_map_entities()</code>
<code>box.widgets.persistent_console_widget.PersistentConsoleWidget</code>	<code>(spinetool-</code>
<code>attribute), 477</code>	<code>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMi</code>
<code>_flush_text_buffer()</code>	<code>method), 385</code>
<code>(spinetool-</code>	<code>get_db_map_parameter_value_or_def_ids()</code>
<code>box.widgets.persistent_console_widget.PersistentConsoleWidget</code>	<code>(spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.Tabul</code>
<code>method), 478</code>	<code>method), 386</code>
<code>_focus_line_edit()</code>	<code>method), 386</code>
<code>(spinetool-</code>	<code>_get_db_map_parameter_values_or_defs()</code> (spine-
<code>box.widgets.custom_qwidgets.HorizontalSpinBox</code>	<code>toolbox.spine_db_editor.widgets.tabular_view_mixin.TabularView</code>
<code>method), 441</code>	<code>method), 386</code>
<code>_format_list_value()</code>	<code>_get_db_map_relationship_ids_to_expand_or_collapse()</code>
<code>(spinetool-</code>	<code>(spinetoolbox.spine_db_editor.graphics_items.ObjectItem</code>
<code>box.spine_db_manager.SpineDBManager</code>	<code>method), 398</code>
<code>method), 603</code>	<code>_get_db_map_relationships_for_graph()</code> (spine-
<code>_format_value()</code>	<code>toolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewM</code>
<code>(spinetool-</code>	<code>method), 361</code>
<code>box.spine_db_manager.SpineDBManager</code>	<code>_get_display_value()</code>
<code>method), 604</code>	<code>(spinetool-</code>
<code>_frame_height()</code>	<code>box.spine_db_editor.widgets.custom_menus.ParameterViewFilter</code>
<code>(spinetool-</code>	<code>method), 334</code>
<code>box.widgets.multi_tab_window.MultiTabWindow</code>	<code>_get_dst_offset()</code>
<code>method), 467</code>	<code>(spinetoolbox.link.LinkBase</code>
<code>_frozen</code> (in module <code>spinetoolbox.config</code>), 504	<code>method), 537</code>
<code>_frozen_table_reload_disabled()</code>	<code>_get_dst_offset()</code>
<code>(spinetool-</code>	<code>(spinetool-</code>
<code>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</code>	<code>box.link.LinkDrawerBase method), 540</code>
<code>method), 390</code>	<code>_get_existing_spec_editor()</code>
<code>_gather_index_names()</code> (in module <code>spinetool-</code>	<code>box.ui_main.ToolboxUI method), 635</code>
<code>box.mvcmodels.map_model), 192</code>	<code>_get_existing_spec_editor_spine_db_editor()</code>
<code>_generate_scenarios()</code>	<code>(spinetool-</code>
<code>(spinetool-</code>	<code>box.spine_db_manager.SpineDBManager</code>
<code>box.spine_db_editor.widgets.scenario_generator.ScenarioGenerator</code>	<code>method), 610</code>
<code>method), 376</code>	<code>_get_field_item()</code>
<code>_get_active_properties_widget()</code>	<code>(spinetool-</code>
<code>(spinetool-</code>	<code>box.spine_db_editor.mvcmodels.single_parameter_models.Single</code>
<code>box.ui_main.ToolboxUI method), 631</code>	<code>method), 300</code>
<code>_get_base_dir()</code>	<code>_get_fields()</code>
<code>(spinetool-</code>	<code>(spinetool-</code>
<code>box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase</code>	<code>box.spine_db_parcel.SpineDBParcel method),</code>
<code>static method), 382</code>	<code>634</code>
<code>_get_commit_msg()</code>	<code>_get_filling()</code>
<code>(spinetool-</code>	<code>(spinetool-</code>
<code>box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase</code>	<code>box.widgets.project_item_drag.ProjectItemSpecArray</code>
<code>method), 381</code>	<code>method), 488</code>
<code>_get_console()</code> (spinetoolbox.ui_main.ToolboxUI	<code>_get_first_chopped_index()</code>
<code>method), 640</code>	<code>(spinetool-</code>
<code>_get_current_class_item()</code>	<code>box.widgets.project_item_drag.ProjectItemSpecArray</code>
<code>(spinetool-</code>	<code>method), 488</code>
<code>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</code>	<code>_get_header_data_from_db()</code>
<code>static method), 385</code>	<code>(spinetool-</code>
<code>_get_current_text()</code>	<code>box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftHead</code>
<code>(spinetool-</code>	<code>method), 282</code>
<code>box.widgets.persistent_console_widget.PersistentConsoleWidget</code>	<code>_get_ids_from_feat_name()</code>
<code>method), 479</code>	<code>(spinetool-</code>
<code>_get_data_for_export()</code>	

box.spine_db_editor.mvcmodels.tool_feature_item.FeatureItem (method), 381
 method), 304
 _get_index_data() (spinetool- box.mvcmodels.compound_table_model.CompoundWithEmptyTableModel (method), 180
 box.spine_db_editor.widgets.custom_delegates.ToolFeatureDelegate (static method), 329
 _get_insert_index() (spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin (method), 361
 box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin (static method), 389
 _get_insert_position() (spinetool- box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView (method), 180
 box.mvcmodels.compound_table_model.CompoundWithEmptyTableModel (method), 180
 _get_insert_position() (spinetool- method), 537
 box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel (method), 241
 _get_joint_angle() (spinetoolbox.link.LinkBase (method), 536
 _get_julia_settings() (spinetool- box.widgets.settings_widget.SettingsWidget (method), 496
 _get_kernel_name_by_exe() (in module spinetool- box.widgets.settings_widget), 497
 _get_method_index() (spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem (method), 306
 method), 306
 _get_names() (spinetool- box.spine_db_editor.widgets.custom_delegates.ScenarioDelegate (method), 330
 method), 330
 _get_names() (spinetool- box.spine_db_editor.widgets.custom_delegates.ToolFeatureDelegate (method), 328
 method), 328
 _get_object_key() (spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin (method), 361
 box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin (method), 361
 _get_offset() (spinetoolbox.link.LinkBase (static method), 536
 method), 536
 _get_parameter_positions() (spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin (method), 361
 box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin (method), 361
 _get_parents() (spinetool- box.spine_db_worker.SpineDBWorker (method), 614
 box.spine_db_worker.SpineDBWorker (method), 614
 _get_parsed_value() (in module spinetool- box.plotting), 554
 box.plotting), 554
 _get_plot_data() (spinetool- box.widgets.plot_widget.PlotWidget (method), 483
 box.widgets.plot_widget.PlotWidget (method), 483
 _get_prefix() (spinetool- box.widgets.persistent_console_widget.PersistentConsoleWidget (method), 479
 box.widgets.persistent_console_widget.PersistentConsoleWidget (method), 479
 _get_relationship_key() (spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin (method), 361
 box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin (method), 361
 _get_rollback_confirmation() (spinetool- box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor (method), 514
 box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor (method), 514

method), 460

`_handle_check_box_clicked()` (spinetoolbox.widgets.add_up_spine_opt_wizard.FailurePage method), 407

`_handle_check_box_state_changed()` (spinetoolbox.spine_db_editor.widgets.mass_select_items_dialogs.MassSelectItemsDialog method), 366

`_handle_check_box_state_changed()` (spinetoolbox.widgets.custom_qwidgets.SelectDatabaseItemsDialog method), 442

`_handle_check_install_finished()` (spinetoolbox.widgets.add_up_spine_opt_wizard.CheckPreviousInstallPage method), 406

`_handle_collisions()` (spinetoolbox.project_item_icon.ProjectItemIcon method), 580

`_handle_command_checked()` (spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget method), 479

`_handle_command_finished()` (spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget method), 480

`_handle_contents_changed()` (spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget method), 478

`_handle_copy_clicked()` (spinetoolbox.widgets.custom_qwidgets.QWizardProcessPage method), 441

`_handle_cursor_position_changed()` (spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget method), 478

`_handle_cursor_position_changed()` (spinetoolbox.widgets.persistent_console_widget._CustomLink method), 477

`_handle_dag_execution_started()` (in module spinetoolbox.spine_engine_worker), 624

`_handle_data_changed()` (spinetoolbox.mvcmodels.empty_row_model.EmptyRowModel method), 181

`_handle_delegate_text_edited()` (spinetoolbox.widgets.custom_editors.SearchBarEditor method), 415

`_handle_drag_about_to_start()` (spinetoolbox.widgets.project_item_drag.ProjectItemButton method), 486

`_handle_empty_rows_inserted()` (spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel method), 180

`_handle_empty_rows_removed()` (spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel method), 180

`_handle_engine_worker_finished()` (spinetoolbox.project.SpineToolboxProject method), 566

`_handle_entities_added()` (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ParameterView method), 289

`_handle_entities_removed()` (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ParameterView method), 289

`_handle_entity_graph_visibility_changed()` (spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphView method), 360

`_handle_entity_group_items_added()` (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.ObjectClassItem method), 250

`_handle_entity_group_items_updated()` (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.ObjectClassItem method), 250

`_handle_entity_tree_current_changed()` (spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularView method), 385

`_handle_entity_tree_selection_changed()` (spinetoolbox.spine_db_editor.widgets.parameter_view_mixin.ParameterView method), 372

`_handle_event_message_arrived()` (in module spinetoolbox.spine_engine_worker), 624

`_handle_event_message_arrived_silent()` (spinetoolbox.spine_engine_worker.SpineEngineWorker method), 626

`_handle_event_msg()` (spinetoolbox.headless.ActionsWithProject method), 515

`_handle_event_msg()` (spinetoolbox.spine_engine_worker.SpineEngineWorker method), 627

`_handle_execution_animation_value_changed()` (spinetoolbox.link.JumpOrLink method), 538

`_handle_first_redo_complete()` (spinetoolbox.spine_db_commands.AddItemsCommand method), 593

`_handle_first_redo_complete()` (spinetoolbox.spine_db_commands.RemoveItemsCommand method), 594

`_handle_first_redo_complete()` (spinetoolbox.spine_db_commands.SpineDBCommand method), 592

`_handle_first_redo_complete()` (spinetoolbox.spine_db_commands.UpdateItemsCommand method), 593

`_handle_first_redo_complete()` (spinetoolbox.spine_engine_worker.SpineEngineWorker method), 626

`_handle_first_redo_complete()` (in module spinetoolbox.spine_engine_worker), 624

`_handle_graph_selection_changed()` (spinetoolbox.spine_db_editor.widgets.parameter_view_mixin.ParameterView method), 371

<code>_handle_hovered()</code>	(spinetool- box.widgets.custom_qwidgets.CustomWidgetAction method), 437	box.spine_db_editor.widgets.add_items_dialogs.AddRelationshipsDialog method), 318
<code>_handle_hovered()</code>	(spinetool- box.widgets.custom_qwidgets.ToolBarWidgetAction method), 438	<code>_handle_model_data_changed()</code> (spinetool- box.spine_db_editor.widgets.manage_items_dialogs.ManageItemsDialog method), 364
<code>_handle_index_clicked()</code>	(spinetool- box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel method), 185	<code>_handle_model_reset()</code> (spinetool- box.spine_db_editor.widgets.manage_items_dialogs.ManageItemsDialog method), 364
<code>_handle_item_move_finished()</code>	(spinetool- box.widgets.custom_qgraphicsviews.CustomQGraphicsView method), 424	<code>_handle_msg_available()</code> (spinetool- box.widgets.persistent_console_widget.PersistentConsoleWidget method), 480
<code>_handle_items_added()</code>	(spinetool- box.spine_db_editor.widgets.custom_menus.ParameterViewFilterMenu method), 334	<code>_handle_node_execution_finished()</code> (in module spinetoolbox.spine_engine_worker), 624
<code>_handle_items_added()</code>	(spinetool- box.spine_db_editor.widgets.custom_menus.TabularViewFilterMenu method), 335	<code>_handle_node_execution_finished()</code> (spinetool- box.headless.ActionsWithProject method), 515
<code>_handle_items_added()</code>	(spinetool- box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 381	<code>_handle_node_execution_ignored()</code> (in module spinetoolbox.spine_engine_worker), 624
<code>_handle_items_removed()</code>	(spinetool- box.spine_db_editor.widgets.custom_menus.ParameterViewFilterMenu method), 334	<code>_handle_node_execution_started()</code> (in module spinetoolbox.spine_engine_worker), 624
<code>_handle_items_removed()</code>	(spinetool- box.spine_db_editor.widgets.custom_menus.TabularViewFilterMenu method), 335	<code>_handle_node_execution_started()</code> (spinetool- box.headless.ActionsWithProject method), 515
<code>_handle_items_removed()</code>	(spinetool- box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 381	<code>_handle_node_execution_started()</code> (spinetool- box.spine_engine_worker.SpineEngineWorker method), 627
<code>_handle_items_updated()</code>	(spinetool- box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 381	<code>_handle_object_tree_selection_changed()</code> (spinetoolbox.spine_db_editor.widgets.parameter_view_mixin.ParameterViewMixin method), 372
<code>_handle_julia_install_finished()</code>	(spinetool- box.widgets.install_julia_wizard.InstallJuliaPage method), 451	<code>_handle_object_tree_selection_changed()</code> (spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin method), 385
<code>_handle_kernel_execution_msg()</code>	(spinetool- box.headless.ActionsWithProject method), 515	<code>_handle_objects_added()</code> (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.RelationshipPivotTableModel method), 292
<code>_handle_kernel_execution_msg()</code>	(spinetool- box.spine_engine_worker.SpineEngineWorker method), 626	<code>_handle_objects_added()</code> (spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin method), 359
<code>_handle_kernel_shutdown()</code>	(spinetool- box.ui_main.ToolboxUI method), 639	<code>_handle_objects_removed()</code> (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.RelationshipPivotTableModel method), 292
<code>_handle_kernel_started_msg()</code>	(spinetool- box.widgets.jupyter_console_widget.JupyterConsoleWidget method), 453	<code>_handle_objects_removed()</code> (spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin method), 360
<code>_handle_line_edit_return_pressed()</code>	(spinetool- box.spine_db_editor.widgets.url_toolbar.UrlToolBar method), 393	<code>_handle_objects_updated()</code> (spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin method), 360
<code>_handle_model_data_changed()</code>	(spinetool- box.spine_db_editor.widgets.add_items_dialogs.AddRelationshipsDialog method), 317	<code>_handle_ok_clicked()</code> (spinetool- box.widgets.dialogs.dialog_manager_widgets.InstallPluginDialog method), 485
<code>_handle_model_data_changed()</code>	(spinetool-	<code>_handle_parameter_definitions_added()</code> (spine-

<code>_handle_selection_changed()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtreeview.ScenarioTreeView method), 353	<code>_handle_use_memory_db_state_changed()</code>	(spine- box.widgets.link_properties_widget.LinkPropertiesWidget method), 460
<code>_handle_selection_changed()</code>	(spinetool- box.widgets.persistent_console_widget.PersistentConsoleWidget method), 478	<code>_handle_value_changed()</code>	(spinetool- box.spine_db_editor.widgets.custom_qwidgets.ShootingLabel method), 355
<code>_handle_server_status_msg()</code>	(spinetool- box.spine_engine_worker.SpineEngineWorker method), 627	<code>_handle_will_have_children_change()</code>	(spinetool- box.spine_db_worker.SpineDBWorker static method), 615
<code>_handle_single_model_about_to_be_reset()</code>	(spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel method), 180	<code>_handle_write_index_value_changed()</code>	(spinetool- box.widgets.link_properties_widget.LinkPropertiesWidget method), 460
<code>_handle_single_model_reset()</code>	(spinetool- box.mvcmodels.compound_table_model.CompoundTableModel method), 180	<code>_handle_zoom_minus_pressed()</code>	(spinetool- box.ui_main.ToolboxUI method), 633
<code>_handle_spin_box_value_changed()</code>	(spinetool- box.spine_db_editor.widgets.add_items_dialogs.AddReadyToShipDialog method), 317	<code>_handle_zoom_plus_pressed()</code>	(spinetool- box.ui_main.ToolboxUI method), 633
<code>_handle_spine_opt_add_up_finished()</code>	(spinetool- box.widgets.add_up_spine_opt_wizard.AddUpSpineOptPageBox method), 407	<code>_handle_zoom_time_line_advanced()</code>	(spinetool- box.widgets.custom_qgraphicsviews.CustomQGraphicsView method), 424
<code>_handle_standard_execution_msg()</code>	(spinetool- box.headless.ActionsWithProject method), 515	<code>_has_x_column()</code>	(in module spinetoolbox.plotting), 553
<code>_handle_standard_execution_msg()</code>	(spinetool- box.spine_engine_worker.SpineEngineWorker method), 626	<code>_header_data()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 289
<code>_handle_tab_window_title_changed()</code>	(spinetool- box.widgets.multi_tab_window.MultiTabWindow method), 466	<code>_header_id()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 288
<code>_handle_table_view_cell_clicked()</code>	(spinetool- box.spine_db_editor.widgets.add_items_dialogs.AddReadyToShipDialog method), 315	<code>_header_ids()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 288
<code>_handle_table_view_current_changed()</code>	(spine- toolbox.spine_db_editor.widgets.add_items_dialogs.AddReadyToShipDialog method), 315	<code>_header_name()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 289
<code>_handle_text_changed()</code>	(spinetool- box.widgets.persistent_console_widget._CustomLineEdit method), 477	<code>_hide_class()</code>	(spinetool- box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 336
<code>_handle_timeout()</code>	(spinetool- box.widgets.custom_qgraphicsscene.DesignGraphicsScene method), 421	<code>_highlight_current_input()</code>	(spinetool- box.widgets.persistent_console_widget.PersistentConsoleWidget method), 479
<code>_handle_transformation_time_line_finished()</code>	(spinetoolbox.widgets.custom_qgraphicsviews.CustomQGraphicsView method), 424	<code>_history_item_available</code>	(spinetool- box.widgets.persistent_console_widget.PersistentConsoleWidget method), 477
<code>_handle_tree_selection_changed()</code>	(spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin method), 360	<code>_ids_from_added_item()</code>	(spinetool- box.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel method), 259
<code>_handle_update_request()</code>	(spinetool- box.widgets.persistent_console_widget.PersistentConsoleWidget method), 478	<code>_ids_from_added_item()</code>	(spinetool- box.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel method), 261
<code>_handle_use_datapackage_state_changed()</code>	(spinetoolbox.widgets.link_properties_widget.LinkPropertiesWidget method), 460	<code>_ids_from_added_item()</code>	(spinetool- box.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase method), 264
		<code>_import_data_cmds()</code>	(spinetool-

<code>box.spine_db_manager.SpineDBManager</code>	<code>box.widgets.indexed_value_table_context_menu.MapTableContext</code>
<code>method), 605</code>	<code>method), 448</code>
<code>_impose_entity_class_id()</code> (<code>spinetool-</code>	<code>_insert_multiple_columns_before()</code> (<code>spinetool-</code>
<code>box.spine_db_editor.mvcmodels.parameter_mixins.ImposeEntityClassMixin</code>	<code>box.widgets.indexed_value_table_context_menu.MapTableContext</code>
<code>method), 276</code>	<code>method), 448</code>
<code>_inc_value()</code> (<code>spinetool-</code>	<code>_insert_multiple_rows_after()</code> (<code>spinetool-</code>
<code>box.widgets.custom_qwidgets.HorizontalSpinBox</code>	<code>box.widgets.indexed_value_table_context_menu.ContextMenuBas</code>
<code>method), 441</code>	<code>method), 446</code>
<code>_included_and_ignored_items()</code> (<code>spinetool-</code>	<code>_insert_multiple_rows_before()</code> (<code>spinetool-</code>
<code>box.spine_engine_worker.SpineEngineWorker</code>	<code>box.widgets.indexed_value_table_context_menu.ContextMenuBas</code>
<code>method), 626</code>	<code>method), 446</code>
<code>_included_items()</code> (<code>spinetool-</code>	<code>_insert_prompt()</code> (<code>spinetool-</code>
<code>box.spine_engine_worker.SpineEngineWorker</code>	<code>box.widgets.persistent_console_widget.PersistentConsoleWidget</code>
<code>method), 626</code>	<code>method), 478</code>
<code>_incoming_connections_and_jumps()</code> (<code>spinetool-</code>	<code>_insert_row_map()</code> (<code>spinetool-</code>
<code>box.project.SpineToolboxProject</code>	<code>box.mvcmodels.compound_table_model.CompoundWithEmptyTab</code>
<code>method), 569</code>	<code>method), 180</code>
<code>_incoming_jumps()</code> (<code>spinetool-</code>	<code>_insert_single_column_after()</code> (<code>spinetool-</code>
<code>box.project.SpineToolboxProject</code>	<code>box.widgets.indexed_value_table_context_menu.MapTableContext</code>
<code>method), 569</code>	<code>method), 448</code>
<code>_index_key_getter()</code> (<code>spinetool-</code>	<code>_insert_single_column_before()</code> (<code>spinetool-</code>
<code>box.spine_db_editor.mvcmodels.pivot_model.PivotModel</code>	<code>box.widgets.indexed_value_table_context_menu.MapTableContext</code>
<code>method), 280</code>	<code>method), 448</code>
<code>_indexes()</code> (<code>spinetool-</code>	<code>_insert_single_model()</code> (<code>spinetool-</code>
<code>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</code>	<code>box.mvcmodels.compound_table_model.CompoundWithEmptyTab</code>
<code>method), 387</code>	<code>method), 180</code>
<code>_infer_and_fill_in_entity_class_id()</code> (<code>spine-</code>	<code>_insert_single_row_after()</code> (<code>spinetool-</code>
<code>toolbox.spine_db_editor.mvcmodels.parameter_mixins.InferEntityClassMixin</code>	<code>box.widgets.indexed_value_table_context_menu.ContextMenuBas</code>
<code>method), 275</code>	<code>method), 446</code>
<code>_init_bg()</code> (<code>spinetool-</code>	<code>_insert_single_row_before()</code> (<code>spinetool-</code>
<code>box.spine_db_editor.graphics_items.EntityItem</code>	<code>box.widgets.indexed_value_table_context_menu.ContextMenuBas</code>
<code>method), 395</code>	<code>method), 446</code>
<code>_init_bg()</code> (<code>spinetool-</code>	<code>_insert_specs()</code> (<code>spinetool-</code>
<code>box.spine_db_editor.graphics_items.RelationshipItem</code>	<code>box.widgets.project_item_drag.ProjectItemSpecArray</code>
<code>method), 397</code>	<code>method), 489</code>
<code>_initialize_page_solution1()</code> (<code>spinetool-</code>	<code>_insert_statusbar_button()</code> (<code>spinetool-</code>
<code>box.widgets.add_up_spine_opt_wizard.TroubleshootSolutionPage</code>	<code>box.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDB</code>
<code>method), 407</code>	<code>method), 369</code>
<code>_initialize_page_solution2()</code> (<code>spinetool-</code>	<code>_insert_stdin_text()</code> (<code>spinetool-</code>
<code>box.widgets.add_up_spine_opt_wizard.TroubleshootSolutionPage</code>	<code>box.widgets.persistent_console_widget.PersistentConsoleWidget</code>
<code>method), 407</code>	<code>method), 478</code>
<code>_input_start_pos</code> (<code>spinetool-</code>	<code>_insert_stdout_text()</code> (<code>spinetool-</code>
<code>box.widgets.persistent_console_widget.PersistentConsoleWidget</code>	<code>box.widgets.persistent_console_widget.PersistentConsoleWidget</code>
<code>property), 477</code>	<code>method), 478</code>
<code>_insert_base_alternative()</code> (<code>spinetool-</code>	<code>_insert_text()</code> (<code>spinetool-</code>
<code>box.spine_db_editor.widgets.scenario_generator.ScenarioGenerator</code>	<code>box.widgets.persistent_console_widget.PersistentConsoleWidget</code>
<code>method), 376</code>	<code>method), 478</code>
<code>_insert_children_sorted()</code> (<code>spinetool-</code>	<code>_insert_text_before_prompt()</code> (<code>spinetool-</code>
<code>box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem</code>	<code>box.widgets.persistent_console_widget.PersistentConsoleWidget</code>
<code>method), 268</code>	<code>method), 478</code>
<code>_insert_connect_tab()</code> (<code>spinetool-</code>	<code>_install_plugin()</code> (<code>spinetool-</code>
<code>box.widgets.multi_tab_window.MultiTabWindow</code>	<code>box.plugin_manager.PluginManager</code>
<code>method), 465</code>	<code>method), 556</code>
<code>_insert_multiple_columns_after()</code> (<code>spinetool-</code>	<code>_interrupt_persistent()</code> (<code>spinetool-</code>

box.widgets.persistent_console_widget.PersistentConsoleWidget), 457
 method), 480
 _invalidate_filter() (spinetool- box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTable
 box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel
 method), 240
 _kill_persistent() (spinetool-
 box.widgets.persistent_console_widget.PersistentConsoleWidget
 method), 480
 _killed (spinetoolbox.widgets.persistent_console_widget.PersistentConsol
 attribute), 477
 _last_column() (spinetool-
 box.widgets.indexed_value_table_context_menu.MapTableContext
 method), 448
 _last_row() (spinetool-
 box.widgets.indexed_value_table_context_menu.ContextMenuBas
 method), 446
 _layout_available() (spinetool-
 box.spine_db_editor.widgets.graph_layout_generator.GraphLayou
 method), 359
 _layout_progressed() (spinetool-
 box.spine_db_editor.widgets.graph_layout_generator.GraphLayou
 method), 359
 _limit_string_x_tick_labels() (in module spine-
 toolbox.plotting), 552
 _load_empty_table_delegate() (spinetool-
 box.widgets.jump_properties_widget.JumpPropertiesWidget
 method), 452
 _load_empty_parameter_value_data() (spinetool-
 box.spine_db_editor.mvcmodels.pivot_table_models.IndexExpans
 method), 291
 _load_empty_relationship_data() (spinetool-
 box.spine_db_editor.mvcmodels.pivot_table_models.Relationship
 method), 292
 _load_full_parameter_value_data() (spinetool-
 box.spine_db_editor.mvcmodels.pivot_table_models.IndexExpans
 method), 291
 _load_full_parameter_value_data() (spinetool-
 box.spine_db_editor.mvcmodels.pivot_table_models.ParameterVa
 method), 290
 _load_installed_plugin() (spinetool-
 box.plugin_manager.PluginManager method),
 556
 _load_registry() (spinetool-
 box.plugin_manager.PluginManager method),
 556
 _log_error() (spinetoolbox.headless.HeadlessLogger
 method), 512
 _log_items_change() (spinetool-
 box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase
 method), 381
 _log_message() (spinetool-
 box.headless.HeadlessLogger method), 512

<code>_log_specification_saved()</code>	(spinetool- box.ui_main.ToolboxUI method), 632	<code>box.spine_db_editor.mvcmodels.alternative_model.AlternativeModelBase</code> static method), 236
<code>_log_warning()</code>	(spinetool- box.headless.HeadlessLogger method), 512	<code>_make_db_item()</code> (spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_model.ParameterValueListModel) static method), 279
<code>_make_add_item_callback()</code>	(spinetool- box.spine_db_worker.SpineDBWorker method), 615	<code>_make_db_item()</code> (spinetool- box.spine_db_editor.mvcmodels.scenario_model.ScenarioModelBase) static method), 296
<code>_make_adder_row()</code>	(spinetool- box.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModelBase) class method), 262	<code>_make_db_item()</code> (spinetool- box.spine_db_editor.mvcmodels.tool_feature_model.ToolFeatureModelBase) static method), 307
<code>_make_all_frozen_headers()</code>	(spinetool- box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin) method), 388	<code>_make_db_map_data()</code> (spinetool- box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModels) method), 245
<code>_make_argument_parser()</code> (in module spinetool- box.main), 544	<code>_make_default_format()</code> (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModels) method), 239	<code>_make_delegate()</code> (spinetool- box.spine_db_editor.widgets.custom_qtableview.ParameterTableView) method), 339
<code>_make_argument_parser()</code> (in module spinetool- box.spine_db_editor.main), 401	<code>_make_delegate()</code> (spinetool- box.spine_db_editor.mvcmodels.alternative_item.DBItemBase) method), 235	<code>_make_docks_menu()</code> (spinetool- box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase) method), 379
<code>_make_auto_filter_menu()</code>	(spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModels) method), 239	<code>_make_execution_animation()</code> (spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_item.ParameterValueListItemBase) method), 538
<code>_make_child()</code>	(spinetool- box.spine_db_editor.mvcmodels.alternative_item.DBItemBase) method), 235	<code>_make_fetch_parent()</code> (spinetool- box.project_item.logging_connection.LoggingConnection) method), 217
<code>_make_child()</code>	(spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_item.ParameterValueListItemBase) method), 277	<code>_make_frozen_headers()</code> (spinetool- box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin) method), 388
<code>_make_child()</code>	(spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_item.ParameterValueListItemBase) method), 277	<code>_make_get_id()</code> (spinetool- box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin) method), 385
<code>_make_child()</code>	(spinetool- box.spine_db_editor.mvcmodels.scenario_item.ScenarioDBItemBase) method), 294	<code>_make_guide_path()</code> (spinetoolbox.link.LinkBase) method), 537
<code>_make_child()</code>	(spinetool- box.spine_db_editor.mvcmodels.scenario_item.ScenarioItemBase) method), 295	<code>_make_header()</code> (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModels) method), 243
<code>_make_child()</code>	(spinetool- box.spine_db_editor.mvcmodels.scenario_item.ScenarioItemBase) method), 295	<code>_make_header()</code> (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModels) method), 243
<code>_make_child()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.FeatureRootItemBase) method), 303	<code>_make_header()</code> (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModels) method), 243
<code>_make_child()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.FeatureRootItemBase) method), 303	<code>_make_header()</code> (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModels) method), 243
<code>_make_child()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.FeatureRootItemBase) method), 303	<code>_make_header()</code> (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModels) method), 243
<code>_make_child()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.FeatureRootItemBase) method), 303	<code>_make_header()</code> (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModels) method), 243
<code>_make_child()</code>	(spinetool- box.spine_db_editor.mvcmodels.tree_item_utility.FeatureItemBase) method), 309	<code>_make_header()</code> (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModels) method), 243
<code>_make_condition_from_ui()</code>	(spinetool- box.widgets.jump_properties_widget.JumpPropertiesWidget) method), 452	<code>_make_header()</code> (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModels) method), 244
<code>_make_db_item()</code>	(spinetool- box.spine_db_editor.mvcmodels.alternative_model.AlternativeModelBase) static method), 236	

<code>_make_hidden_adder_columns()</code>	(spinetool- box.spine_db_editor.mvcmodels.item_metadata_table_model.MetadataTableModel static method), 258	<code>_make_item_to_update()</code>	(spinetool- box.spine_db_editor.mvcmodels.item_metadata_table_model.MetadataTableModel method), 278	<code>_make_item_to_update()</code>	(spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_item.ValueListItem method), 278
<code>_make_hidden_adder_columns()</code>	(spinetool- box.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel static method), 260	<code>_make_item_to_update()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.FeatureLeafItem method), 303	<code>_make_item_to_update()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.FeatureLeafItem method), 303
<code>_make_hidden_adder_columns()</code>	(spinetool- box.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase static method), 262	<code>_make_item_to_update()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 306	<code>_make_item_to_update()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 306
<code>_make_icon()</code>	(spinetool- box.mvcmodels.file_list_models.CommandLineArgItem static method), 183	<code>_make_item_to_update()</code>	(spinetool- box.spine_db_editor.mvcmodels.tree_item_utility.LeafItem method), 311	<code>_make_item_to_update()</code>	(spinetool- box.spine_db_editor.mvcmodels.tree_item_utility.LeafItem method), 311
<code>_make_inserted_frozen_headers()</code>	(spinetool- box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin method), 388	<code>_make_julia_kernel_context_menu()</code>	(spinetool- box.widgets.settings_widget.SettingsWidget method), 494	<code>_make_julia_kernel_context_menu()</code>	(spinetool- box.widgets.settings_widget.SettingsWidget method), 494
<code>_make_item()</code>	(spinetool- box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModelUI method), 245	<code>_make_jupyter_console()</code>	(spinetool- box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModelUI method), 640	<code>_make_jupyter_console()</code>	(spinetool- box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModelUI method), 640
<code>_make_item()</code>	(spinetool- box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyRelationshipParameterValueModel method), 247	<code>_make_layout_generator()</code>	(spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin method), 247	<code>_make_layout_generator()</code>	(spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin method), 247
<code>_make_item_data()</code>	(spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_item.ValueListItem method), 277	<code>_make_log_entry_title()</code>	(spinetool- box.ui_main.ToolboxUI static method), 640	<code>_make_log_entry_title()</code>	(spinetool- box.ui_main.ToolboxUI static method), 640
<code>_make_item_data()</code>	(spinetool- box.spine_db_editor.mvcmodels.scenario_item.ScenarioItem method), 295	<code>_make_log_entry_title()</code>	(spinetool- box.widgets.custom_qtextbrowser.CustomQTextBrowser static method), 433	<code>_make_log_entry_title()</code>	(spinetool- box.widgets.custom_qtextbrowser.CustomQTextBrowser static method), 433
<code>_make_item_data()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 303	<code>_make_main_toolbar()</code>	(spinetool- box.project_item.specification_editor_window.SpecNameDescriptionEditor method), 230	<code>_make_main_toolbar()</code>	(spinetool- box.project_item.specification_editor_window.SpecNameDescriptionEditor method), 230
<code>_make_item_data()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 306	<code>_make_new_items()</code>	(spinetool- box.spine_db_editor.graphics_items.EntityItem method), 396	<code>_make_new_items()</code>	(spinetool- box.spine_db_editor.graphics_items.EntityItem method), 396
<code>_make_item_data()</code>	(spinetool- box.spine_db_editor.mvcmodels.tree_item_utility.LeafItem method), 310	<code>_make_new_items()</code>	(spinetool- box.spine_db_editor.graphics_items.ObjectItem method), 398	<code>_make_new_items()</code>	(spinetool- box.spine_db_editor.graphics_items.ObjectItem method), 398
<code>_make_item_to_add()</code>	(spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_item.ValueListItem method), 278	<code>_make_new_items()</code>	(spinetool- box.widgets.project_item_drag.ProjectItemButton method), 487	<code>_make_new_items()</code>	(spinetool- box.widgets.project_item_drag.ProjectItemButton method), 487
<code>_make_item_to_add()</code>	(spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_item.ValueListItem method), 278	<code>_make_new_items()</code>	(spinetool- box.widgets.project_item_drag.ProjectItemButtonBase method), 486	<code>_make_new_items()</code>	(spinetool- box.widgets.project_item_drag.ProjectItemButtonBase method), 486
<code>_make_item_to_add()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 303	<code>_make_new_items()</code>	(spinetool- box.widgets.project_item_drag.ProjectItemSpecButton method), 487	<code>_make_new_items()</code>	(spinetool- box.widgets.project_item_drag.ProjectItemSpecButton method), 487
<code>_make_item_to_add()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 305	<code>_make_new_items()</code>	(spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin method), 361	<code>_make_new_items()</code>	(spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin method), 361
<code>_make_item_to_add()</code>	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 306	<code>_make_new_items()</code>	(spinetool- box.project_item.specification_editor_window.SpecificationEditor method), 229	<code>_make_new_items()</code>	(spinetool- box.project_item.specification_editor_window.SpecificationEditor method), 229
<code>_make_item_to_add()</code>	(spinetool- box.spine_db_editor.mvcmodels.tree_item_utility.LeafItem method), 311	<code>_make_new_items()</code>	(spinetool- box.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDBEditor method), 369	<code>_make_new_items()</code>	(spinetool- box.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDBEditor method), 369
<code>_make_item_to_add()</code>	(spinetool- box.spine_db_editor.mvcmodels.tree_item_utility.LeafItem method), 311	<code>_make_new_items()</code>	(spinetool- box.widgets.multi_tab_spec_editor.MultiTabSpecEditor method), 311	<code>_make_new_items()</code>	(spinetool- box.widgets.multi_tab_spec_editor.MultiTabSpecEditor method), 311

<i>method</i>), 463		502	
<code>_make_new_tab()</code>	(spinetool- box.widgets.multi_tab_window.MultiTabWindow method), 464	<code>_make_tool_tip()</code>	(spinetool- box.spine_db_editor.graphics_items.CrossHairsItem method), 399
<code>_make_other()</code>	(spinetool- box.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDBEditor method), 368	<code>_make_tool_tip()</code>	(spinetool- box.spine_db_editor.graphics_items.CrossHairsRelationshipItem method), 400
<code>_make_other()</code>	(spinetool- box.widgets.multi_tab_spec_editor.MultiTabSpecEditor method), 462	<code>_make_tool_tip()</code>	(spinetool- box.spine_db_editor.graphics_items.EntityItem method), 395
<code>_make_other()</code>	(spinetool- box.widgets.multi_tab_window.MultiTabWindow method), 464	<code>_make_tool_tip()</code>	(spinetool- box.spine_db_editor.graphics_items.ObjectItem method), 398
<code>_make_parameter_value_to_add()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.ParameterValueTableModel method), 291	<code>_make_tool_tip()</code>	(spinetool- box.spine_db_editor.graphics_items.RelationshipItem method), 397
<code>_make_path()</code>	(spinetool- box.project_item_icon.RankIcon method), 582	<code>_make_ui()</code>	(spinetool- box.project_item.specification_editor_window.SpecificationEditor method), 229
<code>_make_pen()</code>	(spinetool- box.spine_db_editor.graphics_items.ArcItem method), 399	<code>_make_unique_id()</code>	(spinetool- box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel method), 247
<code>_make_pen()</code>	(spinetool- box.spine_db_editor.graphics_items.CrossHairsArcItem method), 401	<code>_make_unique_id()</code>	(spinetool- box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel method), 245
<code>_make_persistent_console()</code>	(spinetool- box.ui_main.ToolboxUI method), 640	<code>_make_unique_id()</code>	(spinetool- box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel method), 247
<code>_make_pinned_value()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtableview.ParameterValueTableView method), 341	<code>_make_unique_id()</code>	(spinetool- box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel method), 247
<code>_make_pixmap()</code>	(spinetool- box.helpers.ColoredIconEngine method), 524	<code>_make_unique_relationship_id()</code>	(spinetool- box.spine_db_editor.mvcmodels.parameter_mixins.MakeRelationshipMixin static method), 276
<code>_make_plot_function()</code>	(in module spinetool- box.plotting), 551	<code>_make_update_item_callback()</code>	(spinetool- box.spine_db_worker.SpineDBWorker method), 615
<code>_make_prompt()</code>	(spinetool- box.widgets.persistent_console_widget.PersistentConsoleWidget method), 478	<code>_make_x_plottable()</code>	(in module spinetool- box.plotting), 550
<code>_make_prompt_block()</code>	(spinetool- box.widgets.persistent_console_widget.PersistentConsoleWidget method), 478	<code>_mark_items_failed()</code>	(in module spinetool- box.spine_engine_worker), 624
<code>_make_properties_tab()</code>	(spinetool- box.ui_main.ToolboxUI method), 630	<code>_mark_items_ignored</code>	(spinetool- box.spine_engine_worker.SpineEngineWorker attribute), 625
<code>_make_python_kernel_context_menu()</code>	(spinetool- box.widgets.settings_widget.SettingsWidget method), 494	<code>_matplotlib_version</code>	(in module spinetool- box.helpers), 519
<code>_make_relationship_on_the_fly()</code>	(spinetool- box.spine_db_editor.mvcmodels.parameter_mixins.MakeRelationshipMixin method), 276	<code>_merge_children()</code>	(spinetool- box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem method), 268
<code>_make_remove_item_callback()</code>	(spinetool- box.spine_db_worker.SpineDBWorker method), 615	<code>_merge_intervals()</code>	(in module spinetool- box.widgets.indexed_value_table_context_menu), 449
<code>_make_tool_button()</code>	(spinetool- box.widgets.toolbars.MainToolBar method),	<code>_merge_local_data_to_project_info()</code>	(spine- toolbox.project.SpineToolboxProject static

method), 560

`_message_available()` (spinetool-
box.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGeneratorRunWidget
method), 359

`_model_data()` (spinetoolbox.helpers.IconListManager
method), 523

`_models_with_db_map()` (spinetool-
box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel
method), 240

`_more_available` (spinetool-
box.spine_db_worker.SpineDBWorker
attribute), 613

`_move_and_resize_line_edit()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget
method), 478

`_move_history()` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget
method), 480

`_move_plus_button()` (spinetool-
box.widgets.multi_tab_window.TabBarPlus
method), 468

`_move_to()` (spinetool-
box.project_commands.MoveIconCommand
method), 572

`_msg_available` (spinetool-
box.widgets.persistent_console_widget.PersistentConsoleWidget
attribute), 477

`_name_from_data()` (spinetool-
box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel
method), 257

`_needs_to_update_headers()` (spinetool-
box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin
method), 385

`_node_execution_finished` (spinetool-
box.spine_engine_worker.SpineEngineWorker
attribute), 625

`_node_execution_started` (spinetool-
box.spine_engine_worker.SpineEngineWorker
attribute), 625

`_notify_resource_changes()` (spinetool-
box.project.SpineToolboxProject
method), 567

`_numpy_string_to_python_strings()` (in module
spinetoolbox.mvcmodels.map_model), 193

`_object_parameter_value_to_add()` (spinetool-
box.spine_db_editor.mvcmodels.pivot_table_models.ParameterValuePivotTableModel
method), 291

`_ok_button_can_be_disabled` (spinetool-
box.widgets.custom_qwidgets.PurgeSettingsDialog
attribute), 442

`_ok_button_can_be_disabled` (spinetool-
box.widgets.custom_qwidgets.SelectDatabaseItemsDialog
attribute), 442

`_open_active_item_dir()` (spinetool-
box.ui_main.ToolboxUI method), 628

`_open_ds_url()` (spinetool-
box.spine_db_editor.widgets.url_toolbar.UrlToolBar
method), 392

`_open_header_editor()` (spinetool-
box.widgets.array_editor.ArrayEditor
method), 409

`_open_header_editor()` (spinetool-
box.widgets.map_editor.MapEditor
method), 461

`_open_header_editor()` (spinetool-
box.widgets.time_pattern_editor.TimePatternEditor
method), 499

`_open_header_editor()` (spinetool-
box.widgets.time_series_fixed_resolution_editor.TimeSeriesFixedResolutionEditor
method), 500

`_open_header_editor()` (spinetool-
box.widgets.time_series_variable_resolution_editor.TimeSeriesVariableResolutionEditor
method), 501

`_open_julia_kernel_resource_dir()` (spinetool-
box.widgets.settings_widget.SettingsWidget
method), 495

`_open_project()` (spinetool-
box.headless.ActionsWithProject
method), 514

`_open_project_directory()` (spinetool-
box.ui_main.ToolboxUI method), 638

`_open_project_item_directory()` (spinetool-
box.ui_main.ToolboxUI method), 638

`_open_purge_settings_dialog()` (spinetool-
box.widgets.link_properties_widget.LinkPropertiesWidget
method), 461

`_open_python_kernel_resource_dir()` (spinetool-
box.widgets.settings_widget.SettingsWidget
method), 495

`_open_scenario_generator()` (spinetool-
box.spine_db_editor.widgets.custom_qtableview.PivotTableView.
method), 345

`_open_scenario_generator()` (spinetool-
box.spine_db_editor.widgets.custom_qtreeview.AlternativeTreeView.
method), 352

`_open_sqlite_url()` (spinetool-
box.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDBEditor
method), 369

`_other_editor_windows` (spinetool-
box.widgets.pivot_table_models.MultiTabWindow
attribute), 464

`_outgoing_connections_and_jumps()` (spinetool-
box.project.SpineToolboxProject
method), 569

`_outgoing_jumps()` (spinetool-
box.project.SpineToolboxProject
method), 569

`_override_console()` (spinetool-

- `box.ui_main.ToolboxUI` method), 634
- `_override_execution_list()` (`spinetoolbox.ui_main.ToolboxUI` method), 634
- `_pack_index()` (`spinetoolbox.mvcmodels.file_list_models.FileListModel` method), 183
- `_paint_as_deselected()` (`spinetoolbox.spine_db_editor.graphics_items.EntityItem` method), 396
- `_paint_as_selected()` (`spinetoolbox.spine_db_editor.graphics_items.EntityItem` method), 395
- `_parameter_position_x()` (`spinetoolbox.spine_db_editor.widgets.select_position_parameters_dialog.SelectPositionParametersDialog` method), 377
- `_parameter_position_y()` (`spinetoolbox.spine_db_editor.widgets.select_position_parameters_dialog.SelectPositionParametersDialog` method), 377
- `_parameter_value_to_update()` (`spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.IndexExpansionPivotTableModel` static method), 291
- `_parameter_value_to_update()` (`spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ParameterValuePivotTableModel` static method), 291
- `_parse_value()` (`spinetoolbox.spine_db_manager.SpineDBManager` static method), 604
- `_paste_single_column()` (`spinetoolbox.widgets.custom_qtableview.IndexedValueTableView` method), 430
- `_paste_to_values_column()` (`spinetoolbox.widgets.custom_qtableview.TimeSeriesFixedResolutionTableView` method), 429
- `_paste_two_columns()` (`spinetoolbox.widgets.custom_qtableview.IndexedValueTableView` method), 430
- `_path_to_executable` (in module `spinetoolbox.config`), 504
- `_perform_pre_exit_tasks()` (`spinetoolbox.ui_main.ToolboxUI` method), 636
- `_pivot_display_row()` (in module `spinetoolbox.plotting`), 554
- `_pivot_index_names()` (in module `spinetoolbox.plotting`), 554
- `_pk_fields` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.ObjectParameterTableView` property), 342
- `_pk_fields` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.ParameterValueTableView` property), 341
- `_pk_fields` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.RelationshipParameterTableView` property), 342
- `_place_icons()` (`spinetoolbox.link.JumpOrLink` method), 538
- `_plot()` (`spinetoolbox.widgets.indexed_value_table_context_menu.MapTableContextMenu` method), 448
- `_plot_bar()` (in module `spinetoolbox.plotting`), 550
- `_plot_column()` (`spinetoolbox.spine_db_editor.widgets.pivot_table_header_view.ParameterTableView` method), 374
- `_plot_double_y_axis()` (in module `spinetoolbox.plotting`), 550
- `_plot_in_window()` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView` method), 344
- `_plot_in_window()` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView` method), 448
- `_plot_or_step()` (in module `spinetoolbox.plotting`), 550
- `_plot_selection()` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.ObjectParameterTableView` method), 342
- `_plot_selection()` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.ObjectParameterTableView` method), 341
- `_plot_selection()` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.ParameterTableView` method), 340
- `_plot_selection()` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.ParameterValueTableView` method), 341
- `_plot_selection()` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.RelationshipParameterTableView` method), 342
- `_plot_selection()` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.RelationshipParameterTableView` method), 342
- `_plot_single_y_axis()` (in module `spinetoolbox.plotting`), 549
- `_plot_stacked_line()` (in module `spinetoolbox.plotting`), 549
- `_pop_local_data_from_items_dict()` (`spinetoolbox.project.SpineToolboxProject` method), 559
- `_pop_unused_columns()` (`spinetoolbox.project_item.logging_connection.LoggingConnection` method), 217
- `_populate_value_table_header_map_menu()` (`spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView` method), 337
- `_populate_relationships_menu()` (`spinetoolbox.spine_db_editor.graphics_items.ObjectItem` method), 398

<code>_populate_cmd_line_args_model()</code>	(<i>spinetoolbox.widgets.jump_properties_widget.JumpPropertiesWidget</i> method), 452	<code>_proxy_model_filter_accepts_row()</code>	(<i>spinetoolbox.widgets.custom_editors.IconColorEditor</i> method), 416
<code>_populate_commit_cache()</code>	(<i>spinetoolbox.spine_db_worker.SpineDBWorker</i> method), 616	<code>_proxy_model_filter_accepts_row()</code>	(<i>spinetoolbox.widgets.custom_editors.SearchBarEditor</i> method), 415
<code>_populate_context_menu()</code>	(<i>spinetoolbox.spine_db_editor.widgets.custom_qtableview.MetadataTableViewBase</i> method), 347	<code>_prune_class()</code>	(<i>spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView</i> method), 337
<code>_populate_executions_menu()</code>	(<i>spinetoolbox.widgets.custom_qtextbrowser.CustomQTextBrowser</i> method), 433	<code>_purge_change_notifiers()</code>	(<i>spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase</i> method), 382
<code>_populate_executions_menu()</code>	(<i>spinetoolbox.widgets.statusbars.MainStatusBar</i> method), 498	<code>_push_notification()</code>	(<i>spinetoolbox.widgets.notification.ChangeNotifier</i> method), 471
<code>_populate_expand_collapse_menu()</code>	(<i>spinetoolbox.spine_db_editor.graphics_items.ObjectItem</i> method), 398	<code>_push_update_cmd_line_args_command()</code>	(<i>spinetoolbox.widgets.jump_properties_widget.JumpPropertiesWidget</i> method), 452
<code>_populate_extension_menu()</code>	(<i>spinetoolbox.widgets.project_item_drag.ProjectItemSpecArray</i> method), 488	<code>_python_interpreter_name()</code>	(<i>spinetoolbox.widgets.kernel_editor.KernelEditorBase</i> method), 456
<code>_populate_filter_validation_menu()</code>	(<i>spinetoolbox.widgets.link_properties_widget.LinkPropertiesWidget</i> method), 460	<code>_python_kernel_display_name()</code>	(<i>spinetoolbox.widgets.kernel_editor.KernelEditorBase</i> method), 456
<code>_populate_main_menu()</code>	(<i>spinetoolbox.project_item.specification_editor_window.SpecificationEditorWindowBase</i> method), 230	<code>_python_kernel_display_name()</code>	(<i>spinetoolbox.spine_db_editor.widgets.spine_db_editor.MiniJuliaKernelEditor</i> method), 459
<code>_preview_available()</code>	(<i>spinetoolbox.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGenerator</i> method), 359	<code>_python_kernel_display_name()</code>	(<i>spinetoolbox.spine_db_editor.widgets.spine_db_editor.MiniPythonKernelEditor</i> method), 459
<code>_print()</code>	(<i>spinetoolbox.headless.HeadlessLogger</i> method), 512	<code>_python_kernel_name()</code>	(<i>spinetoolbox.widgets.kernel_editor.KernelEditorBase</i> method), 456
<code>_process_engine_event()</code>	(<i>spinetoolbox.headless.ActionsWithProject</i> method), 514	<code>_python_kernel_name()</code>	(<i>spinetoolbox.widgets.kernel_editor.MiniJuliaKernelEditor</i> method), 459
<code>_process_event()</code>	(<i>spinetoolbox.spine_engine_worker.SpineEngineWorker</i> method), 626	<code>_python_kernel_name()</code>	(<i>spinetoolbox.widgets.kernel_editor.MiniPythonKernelEditor</i> method), 459
<code>_process_message_arrived</code>	(<i>spinetoolbox.spine_engine_worker.SpineEngineWorker</i> attribute), 625	<code>_qsettings</code>	(<i>spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView</i> property), 336
<code>_program_root</code>	(in module <i>spinetoolbox.config</i>), 504	<code>_qsettings</code>	(<i>spinetoolbox.widgets.custom_qgraphicsviews.CustomQGraphicsView</i> property), 423
<code>_prompt_arrived</code>	(<i>spinetoolbox.spine_engine_worker.SpineEngineWorker</i> attribute), 625	<code>_qsettings</code>	(<i>spinetoolbox.widgets.custom_qgraphicsviews.DesignQGraphicsView</i> property), 425
<code>_prompt_column_count()</code>	(<i>spinetoolbox.widgets.indexed_value_table_context_menu.MapTableContextMenu</i> method), 448	<code>_query()</code>	(<i>spinetoolbox.spine_db_worker.SpineDBWorker</i> method), 614
<code>_prompt_row_count()</code>	(<i>spinetoolbox.widgets.indexed_value_table_context_menu.ContextMenuBase</i> method), 447	<code>_range()</code>	(in module <i>spinetoolbox.widgets.custom_qtableview</i>), 432
<code>_prompt_to_commit_changes()</code>	(<i>spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase</i> method), 381	<code>_read_engine_settings()</code>	(in module <i>spinetoolbox.project</i>), 570

<code>box.widgets.settings_widget.SettingsWidget</code> (method), 496	<code>_refresh_relationship_classes()</code> (<i>spinetool-</i> <code>box.spine_db_editor.graphics_items.ObjectItem</code> (method), 398
<code>_read_pasted_text()</code> (<i>spinetool-</i> <code>box.widgets.custom_qtableview.ArrayTableView</code> (static method), 431	<code>_refresh_selected_indexes()</code> (<i>spinetool-</i> <code>box.spine_db_editor.widgets.custom_qtableview.PivotTableView.</code> (method), 343
<code>_read_pasted_text()</code> (<i>spinetool-</i> <code>box.widgets.custom_qtableview.CopyPasteTableView</code> (static method), 428	<code>_refresh_selected_indexes()</code> (<i>spinetool-</i> <code>box.spine_db_editor.widgets.custom_qtreeview.EntityTreeView</code> (method), 349
<code>_read_pasted_text()</code> (<i>spinetool-</i> <code>box.widgets.custom_qtableview.IndexedParameterTableView</code> (static method), 429	<code>_refresh_single_model()</code> (<i>spinetool-</i> <code>box.mvcmodels.compound_table_model.CompoundWithEmptyTab</code> (method), 180
<code>_read_pasted_text()</code> (<i>spinetool-</i> <code>box.widgets.custom_qtableview.IndexedValueTableView</code> (static method), 430	<code>_refresh_tab_order()</code> (<i>spinetool-</i> <code>box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor</code> (method), 382
<code>_read_pasted_text()</code> (<i>spinetool-</i> <code>box.widgets.custom_qtableview.MapTableView</code> (static method), 431	<code>_refresh_undo_redo_actions()</code> (<i>spinetool-</i> <code>box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase</code> (method), 379
<code>_read_pasted_text()</code> (<i>spinetool-</i> <code>box.widgets.custom_qtableview.TimeSeriesFixedRangeTableView</code> (static method), 429	<code>_register_fetch_parent()</code> (<i>spinetool-</i> <code>box.spine_db_worker.SpineDBWorker</code> (method), 614
<code>_rebind_recursively()</code> (<i>spinetool-</i> <code>box.spine_db_worker.SpineDBWorker</code> (method), 616	<code>_regular_poligon_points()</code> (in module <i>spinetool-</i> <code>box.link</code>), 541
<code>_receive_data_changed()</code> (<i>spinetool-</i> <code>box.project_item.logging_connection.LoggingConnection</code> (method), 217	<code>_relationship_parameter_value_to_add()</code> (<i>spine-</i> <code>toolbox.spine_db_editor.mvcmodels.pivot_table_models.Parameter</code> (method), 291
<code>_recompute_empty_row_map()</code> (<i>spinetool-</i> <code>box.mvcmodels.compound_table_model.CompoundWithEmptyTableModel</code> (method), 180	<code>_reload_entity_metadata()</code> (<i>spinetool-</i> <code>box.spine_db_editor.widgets.item_metadata_editor.ItemMetadata</code> (method), 363
<code>_reconstruct_map()</code> (in module <i>spinetool-</i> <code>box.mvcmodels.map_model</code>), 192	<code>_reload_pivot_table_if_needed()</code> (<i>spinetool-</i> <code>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMi</code> (method), 385
<code>_redo_next()</code> (<i>spinetool-</i> <code>box.spine_db_commands.SpineDBMacro</code> (method), 591	<code>_reload_value_metadata()</code> (<i>spinetool-</i> <code>box.spine_db_editor.widgets.item_metadata_editor.ItemMetadata</code> (method), 363
<code>_refresh_child_map()</code> (<i>spinetool-</i> <code>box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem</code> (method), 269	<code>_reload_value_if_needed()</code> (<i>spinetool-</i> <code>box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckbox</code> (method), 185
<code>_refresh_console_execution_list()</code> (<i>spinetool-</i> <code>box.ui_main.ToolboxUI</code> (method), 634	<code>_remove_and_replace_filtered()</code> (<i>spinetool-</i> <code>box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckbox</code> (method), 185
<code>_refresh_copy_paste_actions()</code> (<i>spinetool-</i> <code>box.spine_db_editor.widgets.custom_qtableview.MetadataTableWidget</code> (method), 347	<code>_remove_arg()</code> (<i>spinetool-</i> <code>box.widgets.jump_properties_widget.JumpPropertiesWidget</code> (method), 452
<code>_refresh_copy_paste_actions()</code> (<i>spinetool-</i> <code>box.spine_db_editor.widgets.custom_qtableview.ParameterTableView</code> (method), 341	<code>_remove_columns()</code> (<i>spinetool-</i> <code>box.widgets.indexed_value_table_context_menu.MapTableContext</code> (method), 448
<code>_refresh_copy_paste_actions()</code> (<i>spinetool-</i> <code>box.spine_db_editor.widgets.custom_qtableview.PivotTableView</code> (method), 346	<code>_remove_data()</code> (<i>spinetool-</i> <code>box.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase</code> (method), 265
<code>_refresh_copy_paste_actions()</code> (<i>spinetool-</i> <code>box.spine_db_editor.widgets.custom_qtreeview.ItemTreeView</code> (method), 351	<code>_remove_disconnect_tab()</code> (<i>spinetool-</i> <code>box.widgets.multi_tab_window.MultiTabWindow</code> (method), 465
<code>_refresh_icons()</code> (<i>spinetool-</i> <code>box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin</code> (method), 359	<code>_remove_item()</code> (<i>spinetool-</i>

`box.spine_db_worker.SpineDBWorker` method), 615

`_remove_items_commands()` (`spinetool-box.spine_db_manager.SpineDBManager` method), 609

`_remove_leaf_item()` (`spinetool-box.mvcmodels.project_item_model.ProjectItemModel` method), 199

`_remove_plugin()` (`spinetool-box.plugin_manager.PluginManager` method), 556

`_remove_rows()` (`spinetool-box.widgets.indexed_value_table_context_menu.ContextMenuBase` method), 447

`_remove_selected()` (`spinetool-box.spine_db_editor.widgets.custom_qtableview.MarshaledTableViewReader` method), 347

`_remove_selected_items()` (`spinetool-box.ui_main.ToolboxUI` method), 638

`_remove_spec()` (`spinetool-box.widgets.project_item_drag.ProjectItemSpecArray` method), 489

`_remove_specs()` (`spinetool-box.widgets.project_item_drag.ProjectItemSpecArray` method), 489

`_remove_values_from_frozen_table()` (`spinetool-box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin` method), 389

`_rename_item()` (`spinetool-box.mvcmodels.project_item_model.ProjectItemModel` method), 200

`_rename_project_item()` (`spinetool-box.ui_main.ToolboxUI` method), 638

`_renew_fetch_parents()` (`spinetool-box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin` method), 359

`_repaint()` (`spinetool-box.project_item_icon.ExecutionIcon` method), 582

`_replace_undo_redo_actions()` (`spinetool-box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase` method), 379

`_reposition_name_item()` (`spinetool-box.project_item_icon.ProjectItemIcon` method), 578

`_reserved_metadata()` (`spinetool-box.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase` method), 265

`_reset_data_count()` (`spinetool-box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModelBase` method), 285

`_reset_fetch_parents()` (`spinetool-box.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel` method), 259

`_reset_fetching_if_required()` (`spinetool-box.spine_db_worker.SpineDBWorker` method), 614

`_reset_filters()` (`spinetool-box.spine_db_editor.widgets.parameter_view_mixin.ParameterViewMixin` method), 371

`_reset_metadata()` (`spinetool-box.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel` method), 258

`_reset_root()` (`spinetool-box.mvcmodels.file_list_models.CommandLineArgsModel` static method), 184

`_reset_specs()` (`spinetool-box.widgets.project_item_drag.ProjectItemSpecArray` method), 489

`_resize_table_view_reader_columns()` (`spinetool-box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin` method), 388

`_resolution_changed()` (`spinetool-box.widgets.time_series_fixed_resolution_editor.TimeSeriesFixedResolutionEditor` method), 499

`_resolution_to_text()` (in module `spinetool-box.widgets.time_series_fixed_resolution_editor`), 499

`_resort()` (`spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.SortableTreeItemUtility` method), 309

`_resources_online()` (`spinetool-box.project_item.logging_connection.LoggingConnection` method), 219

`_resources_to_predecessors_changed()` (`spinetoolbox.project_item.project_item.ProjectItem` method), 222

`_resources_to_predecessors_replaced()` (`spinetoolbox.project_item.project_item.ProjectItem` method), 222

`_resources_to_successors_changed()` (`spinetoolbox.project_item.project_item.ProjectItem` method), 223

`_resources_to_successors_replaced()` (`spinetoolbox.project_item.project_item.ProjectItem` method), 223

`_reestablish_bumped_items()` (`spinetool-box.project_item_icon.ProjectItemIcon` method), 580

`_restart_persistent()` (`spinetool-box.widgets.persistent_console_widget.PersistentConsoleWidget` method), 477

`_restart_timer_refresh_tab_order()` (`spinetool-box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor` method), 477

`_restarted` (`spinetool-box.widgets.persistent_console_widget.PersistentConsoleWidget` attribute), 477

`_restore_dock_widgets()` (`spinetool-`

method), 409
 _set_data() (spinetool- box.widgets.map_value_editor.MapValueEditor method), 462
 _set_data() (spinetool- box.widgets.parameter_value_editor.ParameterValueEditor method), 474
 _set_data() (spinetool- box.widgets.parameter_value_editor_base.ParameterValueEditorBase method), 476
 _set_default_node() (in module spinetool- box.plotting), 553
 _set_default_parameter_data() (spinetool- box.spine_db_editor.widgets.parameter_view_mixin.ParameterViewMixin method), 371
 _set_description() (spinetool- box.project_item.specification_editor_window.SpecNameDescriptionEditor method), 230
 _set_deserialized_item_position() (spinetool- box.ui_main.ToolboxUI static method), 637
 _set_disable_max_relationship_dimension() (spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityItem method), 336
 _set_execution_in_progress() (spinetool- box.ui_main.ToolboxUI method), 638
 _set_execution_visible() (spinetool- box.widgets.custom_qtextbrowser.CustomQTextBrowser method), 433
 _set_extra_columns() (spinetool- box.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel method), 259
 _set_extra_columns() (spinetool- box.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel method), 261
 _set_extra_columns() (spinetool- box.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase method), 264
 _set_max_relationship_dimension() (spinetool- box.spine_db_editor.widgets.custom_qgraphicsviews.EntityItem method), 336
 _set_merge_dbs() (spinetool- box.spine_db_editor.widgets.custom_qgraphicsviews.EntityItem method), 336
 _set_model_data() (spinetool- box.spine_db_editor.widgets.custom_qtableview.MetadataTableView method), 347
 _set_model_data() (spinetool- box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin method), 385
 _set_name() (spinetool- box.project_item.specification_editor_window.SpecNameDescriptionEditor method), 230
 _set_number_or_string_enabled() (spinetool- box.widgets.plain_parameter_value_editor.PlainParameterValueEditor method), 481
 method), 481
 _set_ok_enabled() (spinetool- box.widgets.set_description_dialog.SetDescriptionDialog method), 492
 _set_override_console() (spinetool- box.ui_main.ToolboxUI method), 634
 _set_parameter_data() (in module spinetool- box.spine_db_editor.widgets.custom_qtableview), 440
 _set_position_parameters() (spinetool- box.spine_db_editor.widgets.custom_qgraphicsviews.EntityItem method), 337
 _set_preferred_scene_rect() (spinetool- box.spine_db_editor.widgets.custom_qgraphicsviews.CustomQGraphicsView method), 424
 _set_renderer() (spinetool- box.project_item.specification_editor_window.SpecNameDescriptionEditor method), 395
 _set_save_script_button_enabled() (spinetool- box.widgets.jump_properties_widget.JumpPropertiesWidget method), 452
 _set_single_entity_filters() (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel method), 240
 _set_string_enabled() (spinetool- box.widgets.plain_parameter_value_editor.PlainParameterValueEditor method), 481
 _set_text_elided() (spinetool- box.widgets.custom_qwidgets.ElidedTextMixin method), 420
 _set_up() (spinetoolbox.widgets.custom_menus.FilterMenuBase method), 420
 _set_window_title() (spinetool- box.project_item.specification_editor_window.SpecificationEditor method), 230
 _set_base_metadata() (spinetool- box.spine_db_editor.widgets.pivot_table_header_view.ParameterView method), 374
 _set_def_qgraphicsviews() (spinetool- box.spine_db_parcel.SpineDBParcel method), 612
 _setup() (spinetoolbox.widgets.custom_qgraphicsviews.EntityItem method), 578
 _setup_action_button() (spinetool- box.widgets.custom_qwidgets._MenuToolBar method), 439
 _setup_jupyter_console() (spinetool- box.ui_main.ToolboxUI method), 639
 _setup_persistent_console() (spinetool- box.ui_main.ToolboxUI method), 639
 _setup_properties_dialog() (spinetool- box.ui_main.ToolboxUI method), 628
 _share_item_edit_actions() (spinetool- box.ui_main.ToolboxUI method), 638

<code>_show_add_to_selection</code>	(spinetool- box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel property), 184	<code>box.widgets.jump_properties_widget.JumpPropertiesWidget</code>
<code>_show_add_up_spine_opt_wizard()</code>	(spinetool- box.widgets.settings_widget.SettingsWidget method), 495	<code>_show_value_editor()</code> (spinetool- box.widgets.indexed_value_table_context_menu.ArrayTableContextMenu method), 447
<code>_show_calendar()</code>	(spinetool- box.widgets.time_series_fixed_resolution_editor.TimeSeriesFixedResolutionEditor method), 500	<code>_show_value_editor()</code> (spinetool- box.widgets.indexed_value_table_context_menu.MapTableContextMenu method), 448
<code>_show_close_button()</code>	(spinetool- box.widgets.kernel_editor.KernelEditorBase method), 455	<code>_show_waiting_for_fetcher()</code> (spinetool- box.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDBEditor method), 369
<code>_show_empty</code>	(spinetool- box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel property), 184	<code>_shutdown_engine_kernels()</code> (spinetool- box.ui_main.ToolboxUI method), 640
<code>_show_error_box()</code>	(spinetool- box.headless.HeadlessLogger method), 512	<code>_single_click_model_type()</code> (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel property), 238
<code>_show_error_box()</code>	(spinetoolbox.ui_main.ToolboxUI method), 638	<code>_solve_new_kernel_name()</code> (spinetool- box.widgets.kernel_editor.KernelEditorBase method), 455
<code>_show_filter_menu()</code>	(spinetool- box.spine_db_editor.widgets.url_toolbar.UrlToolBar method), 393	<code>_sort_key()</code> (spinetool- box.spine_db_editor.mvcmodels.single_parameter_models.HalfSortedParameterModel method), 298
<code>_show_information_box()</code>	(spinetool- box.headless.HeadlessLogger method), 512	<code>_sort_key()</code> (spinetool- box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel method), 300
<code>_show_install_julia_wizard()</code>	(spinetool- box.widgets.settings_widget.SettingsWidget method), 495	<code>_sort_key()</code> (spinetool- box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel method), 301
<code>_show_log()</code>	(spinetool- box.widgets.add_up_spine_opt_wizard.TroubleshootingWizard method), 407	<code>_spawn_thread_page()</code> (spinetool- box.qthread_pool_executor.QtBasedThreadPoolExecutor method), 590
<code>_show_message_box()</code>	(spinetool- box.ui_main.ToolboxUI method), 638	<code>_specification_dicts()</code> (in module spinetool- box.headless), 516
<code>_show_spec_form()</code>	(spinetool- box.widgets.project_item_drag.ProjectItemSpecificationArrayEditor method), 488	<code>_specification_id()</code> (spinetool- box.project.SpineToolboxProject method), 561
<code>_show_status_bar_msg()</code>	(spinetool- box.project_item.specification_editor_window.SpecificationEditorWindow method), 229	<code>_split_items_by_type()</code> (spinetool- box.spine_db_worker.SpineDBWorker method), 616
<code>_show_table_context_menu()</code>	(spinetool- box.widgets.array_editor.ArrayEditor method), 409	<code>_split_to_subdags()</code> (spinetool- box.project.SpineToolboxProject method), 566
<code>_show_table_context_menu()</code>	(spinetool- box.widgets.map_editor.MapEditor method), 461	<code>_start</code> (spinetoolbox.headless.ActionsWithProject attribute), 514
<code>_show_table_context_menu()</code>	(spinetool- box.widgets.time_pattern_editor.TimePatternEditor method), 498	<code>_start_flush_timer()</code> (spinetool- box.widgets.persistent_console_widget.PersistentConsoleWidget method), 478
<code>_show_table_context_menu()</code>	(spinetool- box.widgets.time_series_fixed_resolution_editor.TimeSeriesFixedResolutionEditor method), 499	<code>_start_link()</code> (spinetool- box.widgets.persistent_console_widget.PersistentConsoleWidget method), 478
<code>_show_table_context_menu()</code>	(spinetool- box.widgets.time_series_variable_resolution_editor.TimeSeriesVariableResolutionEditor method), 500	<code>_start_relationship()</code> (spinetool- box.widgets.persistent_console_widget.PersistentConsoleWidget method), 478
<code>_show_tool_spec_form()</code>	(spinetool- box.widgets.time_series_variable_resolution_editor.TimeSeriesVariableResolutionEditor method), 500	<code>_start_time_changed()</code> (spinetool- box.widgets.persistent_console_widget.PersistentConsoleWidget method), 478

box.widgets.time_series_fixed_resolution_editor.TimeSeriesFixedResolutionEditor
 method), 500 _tooltip_from_data() (spinetool-
 _stop_execution() (spinetoolbox.ui_main.ToolboxUI box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTable
 method), 638 method), 257
 _stop_extending_graph() (spinetool- _top_children() (spinetool-
 box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin box.spine_db_editor.mvcmodels.alternative_model.AlternativeMo
 method), 359 static method), 236
 _stop_layout_generators() (spinetool- _top_children() (spinetool-
 box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin box.spine_db_editor.mvcmodels.parameter_value_list_model.Par
 method), 361 static method), 279
 _store_export_settings() (spinetool- _top_children() (spinetool-
 box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor box.spine_db_editor.mvcmodels.scenario_model.ScenarioModel
 method), 380 static method), 296
 _store_purge_settings() (spinetool- _top_children() (spinetool-
 box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor box.spine_db_editor.mvcmodels.tool_feature_model.ToolFeatureM
 method), 380 static method), 307
 _suffix() (in module spinetool- _top_children() (spinetool-
 box.spine_db_editor.widgets.scenario_generator), box.spine_db_editor.mvcmodels.tree_model_base.TreeModelBase
 377 static method), 312
 _synch_selection_with_header_tables() (spine- _trigger_filter_menu() (spinetool-
 toolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView box.spine_db_editor.widgets.custom_qtableview.AutoFilterCopyPasteTableView
 method), 346 method), 429
 _tab_slots (spinetool- _trim_columns() (spinetool-
 box.widgets.multi_tab_window.MultiTabWindow box.widgets.indexed_value_table_context_menu.MapTableContex
 attribute), 464 method), 448
 _table_display_row() (in module spinetool- _undo_next() (spinetool-
 box.plotting), 552 box.spine_db_commands.SpineDBMacro
 _take_tab() (spinetool- method), 591
 box.widgets.multi_tab_window.MultiTabWindow _unique_column_ranges() (in module spinetool-
 method), 466 box.widgets.indexed_value_table_context_menu),
 _tasks_before_exit() (spinetool- 449
 box.ui_main.ToolboxUI method), 636 _unique_row_ranges() (in module spinetool-
 _text_alignment_data() (spinetool- box.widgets.indexed_value_table_context_menu),
 box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModelBase
 method), 289 _unique_values() (spinetool-
 _text_edited() (spinetool- box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTable
 box.widgets.custom_qwidgets.FilterWidget method), 256
 method), 437 _unique_window_name() (spinetool-
 _text_to_resolution() (in module spinetool- box.widgets.plot_widget.PlotWidget static
 box.widgets.time_series_fixed_resolution_editor), method), 484
 499 _unset_execution_in_progress() (spinetool-
 _to_selection_lists() (spinetool- box.ui_main.ToolboxUI method), 638
 box.spine_db_editor.widgets.custom_qtableview.PivotTableView _update_actions_availability() (spinetool-
 method), 343 box.spine_db_editor.widgets.custom_qtableview.PivotTableView.
 _to_selection_lists() (spinetool- method), 343
 box.spine_db_editor.widgets.custom_qtableview.PivotTableView _update_actions_availability() (spinetool-
 method), 344 box.spine_db_editor.widgets.custom_qtableview.PivotTableView.
 _to_selection_lists() (spinetool- method), 344
 box.spine_db_editor.widgets.custom_qtableview.PivotTableView _update_actions_availability() (spinetool-
 method), 345 box.spine_db_editor.widgets.custom_qtableview.PivotTableView.
 _toggle_checked_state() (spinetool- method), 345
 box.spine_db_editor.widgets.custom_qtableview.PivotTableView _update_actions_availability() (spinetool-
 method), 346 box.spine_db_editor.widgets.custom_qtableview.PivotTableView.
 _toolbars() (spinetoolbox.ui_main.ToolboxUI method), 345

_update_actions_availability() (spinetool- method), 393
 box.spine_db_editor.widgets.custom_qtableview.PivotTableViewFilterScreenAdditionOptions() (spinetool-
 method), 345 box.widgets.link_properties_widget.LinkPropertiesWidget
 _update_actions_visibility() (spinetool- method), 460
 box.spine_db_editor.widgets.custom_qgraphicsview.update QGraphicsData() (spinetool-
 method), 336 box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin
 _update_add_args_button_enabled() (spinetool- method), 361
 box.widgets.jump_properties_widget.JumpPropertiesWidget
 _update_alternative_selection() (spinetool- method), 346
 box.spine_db_editor.widgets.parameter_view_mixin.update_header_tables() (spinetool-
 method), 372 box.spine_db_editor.widgets.custom_qtableview.PivotTableView
 _update_button_geom() (spinetool- method), 346
 box.widgets.project_item_drag.ProjectItemSpecArray.update_header_tables_geometry() (spinetool-
 method), 488 box.spine_db_editor.widgets.custom_qtableview.PivotTableView
 _update_button_visible_icon_color() (spinetool- method), 392
 box.widgets.project_item_drag.ProjectItemSpecArray.update_history_actions_availability() (spine-
 method), 488 toolbox.spine_db_editor.widgets.url_toolbar.UrlToolBar
 _update_class_attributes() (spinetool- 612
 box.spine_db_editor.widgets.tabular_view_mixin.update_ids() (spinetool-
 method), 385 box.spine_db_parcel.SpineDBParcel method),
 _update_command_name (spinetool- method), 567
 box.spine_db_commands.UpdateItemsCommand _update_item() (spinetool-
 attribute), 593 box.spine_db_worker.SpineDBWorker method),
 _update_cross_hairs_pos() (spinetool- 615
 box.spine_db_editor.widgets.custom_qgraphicsview.update_jump_resources() (spinetool-
 method), 337 box.project.SpineToolboxProject method),
 _update_data() (spinetool- 568
 box.spine_db_editor.mvcmodels.metadata_table_mutable_base.MetadataWidgetMetadataBase() (spinetool-
 method), 265 box.widgets.settings_widget.SettingsWidget
 _update_data_in_db_mgr() (spinetool- method), 495
 box.spine_db_editor.mvcmodels.item_metadata_table_mutable_base.MetadataWidgetMetadataBase() (spinetool-
 method), 259 box.project.SpineToolboxProject method),
 _update_data_in_db_mgr() (spinetool- 565
 box.spine_db_editor.mvcmodels.metadata_table_mutable_base.MetadataWidgetMetadataBase() (spinetool-
 method), 260 box.widgets.code_text_edit.CodeTextEdit
 _update_data_in_db_mgr() (spinetool- method), 410
 box.spine_db_editor.mvcmodels.metadata_table_mutable_base.MetadataWidgetMetadataBase() (spinetool-
 method), 263 box.widgets.code_text_edit.CodeTextEdit
 _update_drop_actions() (spinetool- method), 410
 box.widgets.toolbars.MainToolBar _update_line_number_area_width() (spinetool-
 503 box.widgets.code_text_edit.CodeTextEdit
 _update_ds_url_menu_enabled() (spinetool- method), 410
 box.spine_db_editor.widgets.url_toolbar.UrlToolBar _update_link_drawer_destination() (spine-
 method), 392 toolbox.project_item_icon.ProjectItemIcon
 _update_execute_enabled() (spinetool- method), 579
 box.ui_main.ToolboxUI method), 628 _update_ok_button_enabled() (spinetool-
 _update_execute_selected_enabled() (spinetool- box.widgets.plugin_manager_widgets.InstallPluginDialog
 box.ui_main.ToolboxUI method), 628 method), 485
 _update_export_enabled() (spinetool- _update_open_project_url_menu() (spinetool-
 box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase() (spinetool-
 method), 381 box.spine_db_editor.widgets.url_toolbar.UrlToolBar
 _update_filter_enabled() (spinetool- _update_outgoing_connection_and_jump_resources() (spine-
 box.spine_db_editor.widgets.url_toolbar.UrlFilterDialog (spinetoolbox.project.SpineToolboxProject

method), 567

`_update_parameter_values()` (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.IndexExpansionPivotTableModel method), 291

`_update_parameter_values()` (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ParameterValuePivotTableModel method), 291

`_update_parents_will_have_children()` (spinetoolbox.spine_db_worker.SpineDBWorker method), 615

`_update_path()` (spinetoolbox.project_item_icon.ProjectItemIcon method), 578

`_update_pinned_values()` (spinetoolbox.spine_db_editor.widgets.custom_qtableview.ParameterValueTableView method), 341

`_update_plot()` (spinetoolbox.widgets.array_editor.ArrayEditor method), 409

`_update_plot()` (spinetoolbox.widgets.time_series_fixed_resolution_editor.TimeSeriesFixedResolutionEditor method), 500

`_update_plot()` (spinetoolbox.widgets.time_series_variable_resolution_editor.TimeSeriesVariableResolutionEditor method), 500

`_update_plugin()` (spinetoolbox.plugin_manager.PluginManager method), 556

`_update_predecessor()` (spinetoolbox.project.SpineToolboxProject method), 570

`_update_properties_widget()` (spinetoolbox.widgets.settings_widget.SettingsWidget method), 496

`_update_python_widgets_enabled()` (spinetoolbox.widgets.settings_widget.SettingsWidget method), 494

`_update_qsettings()` (spinetoolbox.ui_main.ToolboxUI method), 628

`_update_ranks()` (spinetoolbox.project.SpineToolboxProject method), 570

`_update_remote_execution_page_widget_status()` (spinetoolbox.widgets.settings_widget.SettingsWidget method), 495

`_update_remove_args_button_enabled()` (spinetoolbox.widgets.jump_properties_widget.JumpPropertiesWidget method), 452

`_update_section_height()` (spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView method), 346

`_update_section_width()` (spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView method), 346

`_update_spec_button_name()` (spinetoolbox.widgets.toolbars.PluginToolBar method), 560

`_update_specification_local_data_store()` (spinetoolbox.project.SpineToolboxProject method), 560

`_update_src_dst_inds()` (spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin method), 361

`_update_successor()` (spinetoolbox.project.SpineToolboxProject method), 570

`_update_text()` (spinetoolbox.widgets.custom_qwidgets.ElidedTextMixin method), 476

`_update_undo_data()` (spinetoolbox.spine_db_commands.RemoveItemsCommand method), 594

`_update_user_input()` (spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget method), 476

`_update_window_modified()` (spinetoolbox.project_item.specification_editor_window.SpecificationEditorWindow method), 518

`_update_zoom_limits()` (spinetoolbox.widgets.custom_qgraphicsviews.CustomQGraphicsView method), 424

`_use_default_editor()` (spinetoolbox.widgets.parameter_value_editor_base.ParameterValueEditorBase method), 475

`_use_editor()` (spinetoolbox.widgets.parameter_value_editor_base.ParameterValueEditorBase method), 475

`_use_smooth_zoom()` (spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 338

`_use_smooth_zoom()` (spinetoolbox.widgets.custom_qgraphicsviews.CustomQGraphicsView method), 423

`_validate_dags()` (spinetoolbox.project.SpineToolboxProject method), 567

`_warn_checked_non_data_items` (spinetoolbox.spine_db_editor.widgets.mass_select_items_dialogs.MassSelectItemsDialog attribute), 367

`_warn_checked_non_data_items` (spinetoolbox.widgets.custom_qwidgets.SelectDatabaseItemsDialog attribute), 442

`_will_have_children_change` (spinetoolbox.spine_db_worker.SpineDBWorker attribute), 613

`_zoom()` (spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 338

`_zoom()` (spinetoolbox.widgets.custom_qgraphicsviews.CustomQGraphicsView method), 423

A

- method*), 424
- AboutWidget (class in *spinetool-*
box.widgets.about_widget), 403
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 316
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 319
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 316
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 315
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 317
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 318
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 320
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 318
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 356
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 356
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 357
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 357
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 357
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 367
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 366
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 366
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 371
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 376
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 377
- accept() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog*
method), 393
- accept() (*spinetoolbox.widgets.install_julia_wizard.InstallJuliaWizard*
method), 451
- accept() (*spinetoolbox.widgets.parameter_value_editor_base.ParameterValueEditorBase*
method), 475
- accept() (*spinetoolbox.widgets.set_description_dialog.SetDescriptionDialog*
method), 492
- accept_index() (*spinetool-*
box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableFilterProxy
method), 293
- accepted_rows() (*spinetool-*
box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel
method), 245
- accepted_rows() (*spinetool-*
box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel
method), 245
- accepted_single_models() (*spinetool-*
box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel
method), 239
- accepting_new_tabs (*spinetool-*
box.widgets.multi_tab_window.MultiTabWindow
property), 464
- accepts_entity_item() (*spinetool-*
box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin
method), 390
- accepts_item() (*spinetool-*
box.fetch_parent.FetchParent *method*), 508
- accepts_item() (*spinetool-*
box.fetch_parent.FlexibleFetchParent *method*), 510
- accepts_item() (*spinetool-*
box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel
method), 239
- accepts_item() (*spinetool-*
box.spine_db_editor.widgets.entity_tree_item.EntityClassItem
method), 249
- accepts_item() (*spinetool-*
box.spine_db_editor.widgets.entity_tree_item.MembersItem
method), 251
- accepts_item() (*spinetool-*
box.spine_db_editor.mvcmodels.entity_tree_item.ObjectItem
method), 253
- accepts_item() (*spinetool-*
box.spine_db_editor.mvcmodels.entity_tree_item.ObjectRelationshipItem
method), 251
- accepts_item() (*spinetool-*
box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem
method), 268
- accepts_item() (*spinetool-*
box.spine_db_editor.mvcmodels.parameter_value_list_item.ListItem
method), 277
- accepts_item() (*spinetool-*
box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureMe
method), 306
- accepts_item() (*spinetool-*
box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureRo
method), 304
- accepts_item() (*spinetool-*
box.spine_db_editor.mvcmodels.tree_item_utility.FetchMoreMixin
method), 309
- accepts_ith_member_object_item() (*spinetool-*
box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMi
method), 390
- accepts_member_object_item() (*spinetool-*
box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMi
method), 390

box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin
method), 390 *add_data()* (*spinetool-*
accepts_parameter_item() (*spinetool-* *box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftAlter*
box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin), 284
method), 390 *add_data()* (*spinetool-*
actionEvent() (*spinetool-* *box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftData*
box.widgets.custom_qwidgets._MenuToolBar *method*), 284
method), 440 *add_data()* (*spinetool-*
actions() (*spinetoolbox.project_item.project_item.ProjectItem* *box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftHead*
method), 224 *method*), 282
ActionsWithProject (*class in spinetoolbox.headless*), *add_data()* (*spinetool-*
513 *box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftObje*
activate() (*spinetool-* *method*), 283
box.project_item.project_item.ProjectItem *add_data()* (*spinetool-*
method), 221 *box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftParan*
activate_item_tab() (*spinetool-* *method*), 283
box.ui_main.ToolboxUI *method*), 631 *add_data()* (*spinetool-*
activate_link_tab() (*spinetool-* *box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftParan*
box.ui_main.ToolboxUI *method*), 631 *method*), 283
activate_no_selection_tab() (*spinetool-* *add_data()* (*spinetool-*
box.ui_main.ToolboxUI *method*), 631 *box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftScen*
add_action() (*spinetool-* *method*), 284
box.widgets.custom_menus.CustomContextMenu *add_db_file()* (*spinetool-*
method), 418 *box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase*
add_action() (*spinetool-* *method*), 379
box.widgets.custom_menus.CustomPopupMenu *add_db_map_id()* (*spinetool-*
method), 418 *box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTree*
add_alternatives() (*spinetool-* *method*), 267
box.spine_db_manager.SpineDBManager *add_db_map_ids()* (*spinetool-*
method), 605 *box.spine_db_editor.graphics_items.EntityItem*
add_arc_item() (*spinetool-* *method*), 397
box.spine_db_editor.graphics_items.EntityItem *add_db_map_ids_to_items()* (*spinetool-*
method), 396 *box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin*
add_arc_item() (*spinetool-* *method*), 360
box.spine_db_editor.graphics_items.RelationshipItem *add_db_map_listener()* (*spinetool-*
method), 397 *box.spine_db_manager.SpineDBManager*
add_array_plot() (*in module spinetoolbox.plotting*), *method*), 600
555 *add_entity_groups()* (*spinetool-*
add_check_boxes() (*in module spinetool-* *box.spine_db_manager.SpineDBManager*
box.widgets.select_database_items), 490 *method*), 606
add_child() (*spinetool-* *add_entity_metadata()* (*spinetool-*
box.mvcmodels.project_tree_item.BaseProjectTreeItem *box.spine_db_manager.SpineDBManager*
method), 206 *method*), 607
add_child() (*spinetool-* *add_error_message()* (*spinetool-*
box.mvcmodels.project_tree_item.CategoryProjectTreeItem *box.ui_main.ToolboxUI* *method*), 634
method), 207 *add_error_message()* (*spinetool-*
add_child() (*spinetool-* *box.widgets.kernel_editor.KernelEditorBase*
box.mvcmodels.project_tree_item.LeafProjectTreeItem *method*), 458
method), 207 *add_event_message()* (*spinetool-*
add_child() (*spinetool-* *box.log_mixin.LogMixin* *method*), 542
box.mvcmodels.project_tree_item.RootProjectTreeItem *add_execute_buttons()* (*spinetool-*
method), 207 *box.widgets.toolbars.MainToolBar* *method*),
503
add_connection() (*spinetool-* *add_features()* (*spinetool-*
box.project.SpineToolboxProject *method*),

<code>box.spine_db_manager.SpineDBManager</code> <code>method</code>), 606	<code>add_items()</code> <code>box.spine_db_worker.SpineDBWorker</code> <code>method</code>), 616
<code>add_frame()</code> <code>box.widgets.custom_qwidgets._MenuToolBar</code> <code>method</code>), 439	<code>add_items_to_cache()</code> <code>box.spine_db_manager.SpineDBManager</code> <code>method</code>), 597
<code>add_heat_map()</code> <code>box.spine_db_editor.widgets.custom_qgraphicsviews.DesignQGraphicsView</code> <code>method</code>), 337	<code>add_items_to_db()</code> <code>box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel</code> <code>method</code>), 246
<code>add_icon()</code> <code>box.widgets.custom_qgraphicsviews.DesignQGraphicsView</code> <code>method</code>), 425	<code>add_items_to_db()</code> <code>box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel</code> <code>method</code>), 245
<code>add_item()</code> <code>(spinetoolbox.fetch_parent.FetchParent</code> <code>method</code>), 508	<code>add_items_to_db()</code> <code>box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel</code> <code>method</code>), 247
<code>add_item()</code> <code>(spinetoolbox.project.SpineToolboxProject</code> <code>method</code>), 562	<code>add_items_to_db()</code> <code>box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel</code> <code>method</code>), 247
<code>add_item_metadata()</code> <code>box.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel</code> <code>method</code>), 259	<code>add_items_to_filter_list()</code> <code>box.widgets.custom_menus.FilterMenuBase</code> <code>method</code>), 420
<code>add_item_to_db()</code> <code>box.spine_db_editor.mvcmodels.alternative_item.AlternativeItem</code> <code>method</code>), 236	<code>add_julia_kernel()</code> <code>box.widgets.settings_widget.SettingsWidget</code> <code>method</code>), 497
<code>add_item_to_db()</code> <code>box.spine_db_editor.mvcmodels.parameter_value_list_item.ParameterValueListItem</code> <code>method</code>), 278	<code>add_jump()</code> <code>(spinetoolbox.project.SpineToolboxProject</code> <code>method</code>), 564
<code>add_item_to_db()</code> <code>box.spine_db_editor.mvcmodels.parameter_value_list_item.ParameterValueListItem</code> <code>method</code>), 278	<code>add_jump()</code> <code>box.widgets.custom_qgraphicsviews.DesignQGraphicsView</code> <code>method</code>), 426
<code>add_item_to_db()</code> <code>box.spine_db_editor.mvcmodels.scenario_item.ScenarioAlternativeItem</code> <code>method</code>), 295	<code>add_kernel()</code> <code>box.widgets.custom_menus.KernelsPopupMenu</code> <code>method</code>), 419
<code>add_item_to_db()</code> <code>box.spine_db_editor.mvcmodels.scenario_item.ScenarioAlternativeItem</code> <code>method</code>), 294	<code>add_legend()</code> <code>box.widgets.plot_widget.PlotWidget</code> <code>method</code>), 489
<code>add_item_to_db()</code> <code>box.spine_db_editor.mvcmodels.tool_feature_item.FeatureLeafItem</code> <code>method</code>), 303	<code>add_link()</code> <code>(spinetoolbox.link.ConnectionLinkDrawer</code> <code>method</code>), 540
<code>add_item_to_db()</code> <code>box.spine_db_editor.mvcmodels.tool_feature_item.FeatureLeafItem</code> <code>method</code>), 305	<code>add_link()</code> <code>(spinetoolbox.link.JumpLinkDrawer</code> <code>method</code>), 541
<code>add_item_to_db()</code> <code>box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem</code> <code>method</code>), 306	<code>add_link()</code> <code>(spinetoolbox.link.LinkDrawerBase</code> <code>method</code>), 540
<code>add_item_to_db()</code> <code>box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem</code> <code>method</code>), 304	<code>add_link()</code> <code>box.widgets.custom_qgraphicsviews.DesignQGraphicsView</code> <code>method</code>), 425
<code>add_item_to_db()</code> <code>box.spine_db_editor.mvcmodels.tree_item_utility.LeafItem</code> <code>method</code>), 310	<code>add_list_values()</code> <code>box.spine_db_manager.SpineDBManager</code> <code>method</code>), 606
<code>add_items()</code> <code>box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel</code> <code>method</code>), 185	<code>add_log_message()</code> <code>(spinetoolbox.log_mixin.LogMixin</code> <code>method</code>), 542
<code>add_items()</code> <code>box.spine_db_manager.SpineDBManager</code> <code>method</code>), 605	<code>add_log_message()</code> <code>box.widgets.custom_qtextbrowser.CustomQTextBrowser</code> <code>method</code>), 433

add_main_menu() (spinetool- method), 605
 box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase.add_objects_at_position() (spinetool-
 method), 379
 box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView.add_main_menu() (spinetool-
 method), 336
 box.spine_db_editor.widgets.url_toolbar.UrlToolBar.add_objects_at_position() (spinetool-
 method), 392
 box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin.add_members() (spinetool-
 method), 362
 box.spine_db_editor.widgets.add_items_dialogs.AddItemsDialog.add_items_definitions() (spinetool-
 method), 319
 box.spine_db_manager.SpineDBManager.add_menu_actions() (spinetool-
 method), 606
 box.ui_main.ToolboxUI method), 633
 add_message() (spinetool-
 box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase method), 606
 method), 379
 add_message() (spinetoolbox.ui_main.ToolboxUI
 method), 634
 box.spine_db_manager.SpineDBManager
 method), 607
 add_message() (spinetool-
 box.widgets.kernel_editor.KernelEditorBase
 method), 458
 add_metadata() (spinetool-
 box.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel method), 640
 method), 261
 add_metadata() (spinetool-
 box.spine_db_manager.SpineDBManager
 method), 607
 add_new_tab() (spinetool-
 box.widgets.multi_tab_window.MultiTabWindow
 method), 465
 add_notification() (spinetool-
 box.project_item.project_item.ProjectItem
 method), 222
 add_notification() (spinetool-
 box.project_item_icon.ExclamationIcon
 method), 582
 add_object_classes() (spinetool-
 box.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView method), 350
 method), 350
 add_object_classes() (spinetool-
 box.spine_db_manager.SpineDBManager
 method), 605
 add_object_group() (spinetool-
 box.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView method), 350
 method), 350
 add_object_groups() (spinetool-
 box.spine_db_manager.SpineDBManager
 method), 606
 add_object_metadata() (spinetool-
 box.spine_db_manager.SpineDBManager
 method), 605
 add_objects() (spinetool-
 box.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView method), 350
 method), 350
 add_objects() (spinetool-
 box.spine_db_manager.SpineDBManager
 method), 351

<code>add_relationship_classes()</code>	(<i>spinetool-</i> <i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 605	<i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 607
<code>add_relationships()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.widgets.add_items_dialogs.ManageRelationshipDialog</i> <i>method</i>), 318	<i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 606
<code>add_relationships()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView</i> <i>method</i>), 350	<i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 606
<code>add_relationships()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.widgets.custom_qtreeview.RelationshipTreeView</i> <i>method</i>), 351	<i>box.widgets.add_up_spine_opt_wizard._PageId</i> <i>attribute</i>), 406
<code>add_relationships()</code>	(<i>spinetool-</i> <i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 606	<i>box.widgets.add_up_spine_opt_wizard._PageId</i> <i>attribute</i>), 406
<code>add_row_to_exception()</code>	(in module <i>spinetool-</i> <i>box.plotting</i>), 554	<i>box.spine_db_editor.widgets.url_toolbar.UrlToolBar</i> <i>method</i>), 392
<code>add_rows()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.single_parameter_models.HalfSpineTableModel</i> <i>method</i>), 298	<i>box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTable</i> <i>method</i>), 255
<code>add_scenarios()</code>	(<i>spinetool-</i> <i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 605	<i>box.ui_main.ToolboxUI</i> <i>method</i>), 634
<code>add_specification()</code>	(<i>spinetool-</i> <i>box.mvcmodels.project_item_specification_models.ProjectItemSpecificationModel</i> <i>method</i>), 203	<i>box.widgets.kernel_editor.KernelEditorBase</i> <i>method</i>), 458
<code>add_specification()</code>	(<i>spinetool-</i> <i>box.project.SpineToolboxProject</i> <i>method</i>), 560	<i>box.ui_main.ToolboxUI</i> <i>method</i>), 633
<code>add_specification()</code>	(<i>spinetool-</i> <i>box.ui_main.ToolboxUI</i> <i>method</i>), 632	<i>box.widgets.custom_qwidgets._MenuToolBar</i> <i>method</i>), 439
<code>add_stderr()</code>	(<i>spinetool-</i> <i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i> <i>method</i>), 479	<i>box.widgets.custom_qwidgets._MenuToolBar</i> <i>method</i>), 439
<code>add_stdin()</code>	(<i>spinetool-</i> <i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i> <i>method</i>), 479	<i>box.project_commands</i>), 573
<code>add_stdout()</code>	(<i>spinetool-</i> <i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i> <i>method</i>), 479	<i>box.spine_db_commands</i>), 592
<code>add_success_message()</code>	(<i>spinetool-</i> <i>box.ui_main.ToolboxUI</i> <i>method</i>), 634	<i>box.spine_db_editor.widgets.add_items_dialogs</i>), 315
<code>add_success_message()</code>	(<i>spinetool-</i> <i>box.widgets.kernel_editor.KernelEditorBase</i> <i>method</i>), 458	<i>box.project_commands</i>), 574
<code>add_time_series_plot()</code>	(in module <i>spinetool-</i> <i>box.plotting</i>), 555	<i>box.spine_db_editor.widgets.add_items_dialogs</i>), 315
<code>add_to_model()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.pivot_model.PivotModel</i> <i>method</i>), 279	<i>box.spine_db_editor.widgets.add_items_dialogs</i>), 319
<code>add_to_model()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.pivot_table_model.PivotTableModel</i> <i>method</i>), 286	<i>box.spine_db_editor.widgets.add_items_dialogs</i>), 316
<code>add_tool_feature_methods()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.widgets.add_items_dialogs</i>), 315	<i>box.spine_db_editor.widgets.add_items_dialogs</i>), 316

- 317
- AddProjectItemsCommand (class in *spinetoolbox.project_commands*), 572
- AddProjectItemWidget (class in *spinetoolbox.widgets.add_project_item_widget*), 404
- AddReadyRelationshipsDialog (class in *spinetoolbox.spine_db_editor.widgets.add_items_dialogs*), 314
- AddRelationshipClassesDialog (class in *spinetoolbox.spine_db_editor.widgets.add_items_dialogs*), 316
- AddRelationshipsDialog (class in *spinetoolbox.spine_db_editor.widgets.add_items_dialogs*), 317
- AddSpecificationCommand (class in *spinetoolbox.project_commands*), 576
- AddUpSpineOptAgainPage (class in *spinetoolbox.widgets.add_up_spine_opt_wizard*), 408
- AddUpSpineOptPage (class in *spinetoolbox.widgets.add_up_spine_opt_wizard*), 407
- AddUpSpineOptWizard (class in *spinetoolbox.widgets.add_up_spine_opt_wizard*), 406
- advance_query() (*spinetoolbox.spine_db_worker.SpineDBWorker* method), 614
- age (*spinetoolbox.spine_db_commands.AgedUndoCommand* property), 591
- AgedUndoCommand (class in *spinetoolbox.spine_db_commands*), 591
- AgedUndoStack (class in *spinetoolbox.spine_db_commands*), 590
- all_combinations() (in module *spinetoolbox.spine_db_editor.scenario_generation*), 402
- all_databases() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs* method), 315
- all_databases() (*spinetoolbox.spine_db_editor.widgets.edit_or_remove_items_dialogs* method), 355
- all_db_maps() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs* method), 316
- all_db_maps() (*spinetoolbox.spine_db_editor.widgets.edit_or_remove_items_dialogs* method), 356
- all_header_names() (*spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ParameterWidgetPivotTableModel* method), 290
- all_item_names (*spinetoolbox.project.SpineToolboxProject* property), 558
- all_tabs() (*spinetoolbox.widgets.multi_tab_window.MultiTabWindow* method), 465
- ALTERNATIVE_DATA (in module *spinetoolbox.spine_db_editor.mvcmodels.mime_types*), 266
- alternative_id (*spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioAlternative* property), 295
- alternative_id_list (*spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioItem* property), 294
- alternative_selection_changed (*spinetoolbox.spine_db_editor.widgets.custom_qtreeview.AlternativeTreeView* attribute), 352
- AlternativeDelegate (class in *spinetoolbox.spine_db_editor.widgets.custom_delegates*), 329
- AlternativeItem (class in *spinetoolbox.spine_db_editor.mvcmodels.alternative_item*), 235
- AlternativeModel (class in *spinetoolbox.spine_db_editor.mvcmodels.alternative_model*), 236
- AlternativeNameDelegate (class in *spinetoolbox.spine_db_editor.widgets.custom_delegates*), 328
- AlternativeTreeView (class in *spinetoolbox.spine_db_editor.widgets.custom_qtreeview*), 352
- AnsiEscapeCodeHandler (class in *spinetoolbox.widgets.persistent_console_widget*), 480
- answer_prompt() (*spinetoolbox.server.engine_client.EngineClient* method), 233
- answer_prompt() (*spinetoolbox.spine_engine_manager.LocalSpineEngineManager* method), 620
- answer_prompt() (*spinetoolbox.spine_engine_manager.RemoteSpineEngineManager* method), 622
- answer_prompt() (*spinetoolbox.spine_engine_manager.SpineEngineManagerBase* method), 618
- any_checked() (*spinetoolbox.spine_db_editor.widgets.mass_select_items_dialogs._SelectDatabaseItems* method), 491
- any_structural_item_checked() (*spinetoolbox.widgets.select_database_items.SelectDatabaseItems* method), 491

- method), 491
- app_settings (spinetoolbox.project.SpineToolboxProject property), 558
- append() (spinetoolbox.widgets.custom_qtextbrowser.CustomButtonWidget method), 433
- append_arg() (spinetoolbox.mvcmodels.file_list_models.CommandLineArgsModel method), 183
- append_children() (spinetoolbox.mvcmodels.minimal_tree_model.TreeItem method), 197
- append_children_by_id() (spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem method), 268
- append_column() (spinetoolbox.mvcmodels.map_model.MapModel method), 189
- APPLICATION_PATH (in module spinetoolbox.config), 504
- apply_filter() (spinetoolbox.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel method), 185
- apply_graph_style() (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 383
- apply_pivot_style() (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 383
- apply_rotation() (spinetoolbox.spine_db_editor.graphics_items.EntityItem method), 396
- apply_stacked_style() (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 383
- apply_zoom() (spinetoolbox.spine_db_editor.graphics_items.ArcItem method), 399
- apply_zoom() (spinetoolbox.spine_db_editor.graphics_items.EntityItem method), 396
- apply_zoom() (spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 338
- ArcItem (class in spinetoolbox.spine_db_editor.graphics_items), 398
- area (spinetoolbox.spine_db_editor.widgets.custom_qtableview.FrozenTableView property), 346
- area (spinetoolbox.spine_db_editor.widgets.pivot_table_header_view.PivotTableHeaderView property), 373
- area (spinetoolbox.spine_db_editor.widgets.tabular_view_header_widget.TabularViewHeaderWidget property), 384
- args (spinetoolbox.mvcmodels.file_list_models.CommandLineArgsModel property), 183
- args() (spinetoolbox.execution_managers.QProcessExecutionManager method), 506
- args_updated (spinetoolbox.mvcmodels.file_list_models.CommandLineArgsModel attribute), 183
- ATTRIBUTE_ERROR (spinetoolbox.headless.Status attribute), 516
- ARRAY (spinetoolbox.widgets.parameter_value_editor_base.ValueType attribute), 475
- array() (spinetoolbox.mvcmodels.array_model.ArrayModel method), 176
- ArrayEditor (class in spinetoolbox.widgets.array_editor), 408
- ArrayModel (class in spinetoolbox.mvcmodels.array_model), 175
- ArrayContextMenu (class in spinetoolbox.widgets.indexed_value_table_context_menu), 447
- ArrayTableView (class in spinetoolbox.widgets.custom_qtableview), 431
- ArrayValueEditor (class in spinetoolbox.widgets.parameter_value_editor), 409
- asyncio_logger (in module spinetoolbox.widgets.jupyter_console_widget), 453
- AutoFillerCopyPasteTableView (class in spinetoolbox.widgets.custom_qtableview), 428
- axes (spinetoolbox.widgets.plot_canvas.PlotCanvas property), 482
- ## B
- backup_project_file() (spinetoolbox.project_upgrader.ProjectUpgrader method), 588
- BAR (spinetoolbox.plotting.PlotType attribute), 547
- BaseProjectTreeItem (class in spinetoolbox.mvcmodels.project_tree_item), 205
- batch_set_check_state() (in module spinetoolbox.widgets.select_database_items), 491
- batch_set_data() (spinetoolbox.mvcmodels.array_model.ArrayModel method), 176
- batch_set_data() (spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel method), 179
- batch_set_data() (spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel method), 194
- batch_set_data() (spinetoolbox.mvcmodels.time_pattern_model.TimePatternModel method), 211
- batch_set_data() (spinetoolbox.mvcmodels.time_series_model_fixed_resolution.TimeSeriesModel method), 213
- batch_set_data() (spinetoolbox.mvcmodels.time_series_model_variable_resolution.TimeSeriesModel method), 213

<code>call_on_focused_widget()</code> (in module <code>spinetool-box.helpers</code>), 526	<code>box.spine_db_editor.mvcmodels.entity_tree_item.RelationshipItem</code> method), 254
<code>call_open_console()</code> (<code>spinetool-box.widgets.custom_menus.KernelsPopupMenu</code> method), 419	<code>can_fetch_more()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem</code> method), 268
<code>call_open_project()</code> (<code>spinetool-box.widgets.custom_menus.RecentProjectsPopupMenu</code> method), 419	<code>can_fetch_more()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.tree_item_utility.FetchMoreMixin</code> method), 309
<code>call_reset_model()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.pivot_table_models.IndexEditorSpineDBTableModel</code> method), 291	<code>can_fetch_more()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.tree_item_utility.LeafItem</code> method), 311
<code>call_reset_model()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.pivot_table_models.ParameterSpineDBTableModel</code> method), 290	<code>can_fetch_more()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.spine_db_manager.SpineDBManager</code> method), 597
<code>call_reset_model()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModelBase</code> method), 286	<code>can_fetch_more()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.spine_db_worker.SpineDBWorker</code> method), 615
<code>call_reset_model()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.pivot_table_models.RelationshipPivotTableModel</code> method), 292	<code>can_paste()</code> (<code>spinetool-box.widgets.custom_qtreeview.AlternativeTreeView</code> method), 352
<code>call_reset_model()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.pivot_table_models.ScenarioAlternativePivotTableModel</code> method), 292	<code>can_paste()</code> (<code>spinetool-box.widgets.custom_qtreeview.ScenarioTreeView</code> method), 353
<code>call_set_description()</code> (<code>spinetool-box.project.SpineToolboxProject</code> method), 559	<code>can_paste()</code> (<code>spinetool-box.widgets.custom_qtableview.CopyPasteTableView</code> method), 428
<code>can_be_filtered</code> (<code>spinetool-box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel</code> property), 245	<code>can_paste()</code> (<code>spinetool-box.widgets.custom_qtreeview.CopyPasteTreeView</code> method), 434
<code>can_be_filtered</code> (<code>spinetool-box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel</code> property), 298	<code>CANCEL_OPERATION</code> (<code>spinetool-box.widgets.scenario_generator._ScenarioNameModel</code> attribute), 375
<code>can_copy()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_qtreeview.AlternativeTreeView</code> method), 352	<code>cancelPressed</code> (<code>spinetool-box.widgets.custom_qwidgets.FilterWidget</code> attribute), 437
<code>can_copy()</code> (<code>spinetool-box.widgets.custom_qtableview.CopyPasteTableView</code> method), 428	<code>canDropMimeData()</code> (<code>spinetool-box.mvcmodels.file_list_models.JumpCommandLineArgsModel</code> method), 184
<code>can_copy()</code> (<code>spinetool-box.widgets.custom_qtreeview.CopyPasteTreeView</code> method), 434	<code>canDropMimeData()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.scenario_model.ScenarioModel</code> method), 296
<code>can_fetch_more()</code> (<code>spinetool-box.mvcmodels.minimal_tree_model.TreeItem</code> method), 197	<code>canDropMimeData()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.tool_feature_model.ToolFeatureModel</code> method), 307
<code>can_fetch_more()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.entity_tree_item.EntityItem</code> method), 252	<code>canFetchMore()</code> (<code>spinetool-box.mvcmodels.compound_table_model.CompoundTableModel</code> method), 179
<code>can_fetch_more()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.entity_tree_item.MemberObjectModel</code> method), 253	<code>canFetchMore()</code> (<code>spinetool-box.mvcmodels.empty_row_model.EmptyRowModel</code> method), 181
<code>can_fetch_more()</code> (<code>spinetool-box.spine_db_editor.mvcmodels.entity_tree_item.ObjectClassModel</code> method), 250	<code>canFetchMore()</code> (<code>spinetool-box.mvcmodels.filter_checkbox_list_model.LazyFilterCheckboxList</code> method), 185
<code>can_fetch_more()</code> (<code>spinetool-</code>	<code>canFetchMore()</code> (<code>spinetool-</code>

box.mvcmodels.minimal_table_model.MinimalTableModel *box.widgets.custom_editors*), 415
method), 193
canFetchMore() (*spinetool-* **CheckPreviousInstallPage** (*class in spinetool-*
box.mvcmodels.minimal_tree_model.MinimalTreeModel), 196
method), 199
canFetchMore() (*spinetool-* **child()** (*spinetoolbox.mvcmodels.minimal_tree_model.TreeItem*
box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel
method), 239
canFetchMore() (*spinetool-* **child_count()** (*spinetool-*
box.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase
method), 263
canFetchMore() (*spinetool-* **child_count()** (*spinetool-*
box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModelBase
method), 285
canvas (*spinetoolbox.widgets.plot_widget.PlotWidget* at-
tribute), 483
category_of_item() (*spinetool-* **child_item_class** (*spinetool-*
box.mvcmodels.project_item_model.ProjectItemModel *box.spine_db_editor.mvcmodels.entity_tree_item.MembersItem*
method), 201
CategoryProjectTreeItem (*class in spinetool-* **child_item_class** (*spinetool-*
box.mvcmodels.project_tree_item), 207
center_items() (*spinetool-* **child_item_class** (*spinetool-*
box.widgets.custom_qgraphicsscene.CustomGraphicsScene *box.spine_db_editor.mvcmodels.entity_tree_item.ObjectClassItem*
method), 421
change_filter() (*spinetool-* **child_item_class** (*spinetool-*
box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin *box.spine_db_editor.mvcmodels.entity_tree_item.ObjectItem*
method), 389
ChangeNotifier (*class in spinetool-* **child_item_class** (*spinetool-*
box.widgets.notification), 470
ChangeSpecPropertyCommand (*class in spinetool-* **child_item_class** (*spinetool-*
box.project_item.specification_editor_window), 228
CharIconEngine (*class in spinetoolbox.helpers*), 523
check_options() (*spinetool-* **child_item_class** (*spinetool-*
box.widgets.kernel_editor.KernelEditorBase *box.spine_db_editor.mvcmodels.entity_tree_item.RelationshipClass*
method), 455
CHECK_PREVIOUS_INSTALL (*spinetool-* **child_item_class** (*spinetool-*
box.widgets.add_up_spine_opt_wizard.PageId *box.spine_db_editor.mvcmodels.entity_tree_item.RelationshipTree*
attribute), 406
CheckBoxDelegate (*class in spinetool-* **child_item_class** (*spinetool-*
box.widgets.custom_delegates), 412
checked_state_changed (*spinetool-* **child_number()** (*spinetool-*
box.spine_db_editor.widgets.mass_select_items_dialogs.SelectDatabaseItems
attribute), 365
checked_state_changed (*spinetool-* **ChildCyclingKeyPressFilter** (*class in spinetool-*
box.widgets.select_database_items.SelectDatabaseItems *box.helpers*), 526
attribute), 491
checked_states() (*spinetool-* **children()** (*spinetool-*
box.spine_db_editor.widgets.mass_select_items_dialogs.SelectDatabaseItems *spinetoolbox.mvcmodels.minimal_tree_model.TreeItem*
method), 365
checked_states() (*spinetool-* **children_ids** (*spinetool-*
box.widgets.select_database_items.SelectDatabaseItems *box.spine_db_editor.mvcmodels.project_tree_item.BaseProjectTreeItem*
method), 491
CheckListEditor (*class in spinetool-* **class_renderer()** (*spinetool-*

<i>box.spine_db_icon_manager.SpineDBIconManager</i> method), 595	<i>box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTree</i> method), 269
<code>clean_up()</code> (<i>spinetool-</i> <i>box.plugin_manager.PluginWorker</i> method), 557	<code>clear_cross_hairs_items()</code> (<i>spinetool-</i> <i>box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphi</i> method), 337
<code>clean_up()</code> (<i>spinetool-</i> <i>box.spine_db_manager.SpineDBManager</i> method), 601	<code>clear_filter()</code> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableSo</i> method), 293
<code>clean_up()</code> (<i>spinetool-</i> <i>box.spine_db_worker.SpineDBWorker</i> method), 614	<code>clear_filter()</code> (<i>spinetool-</i> <i>box.widgets.custom_qwidgets.FilterWidget</i> method), 437
<code>clean_up()</code> (<i>spinetool-</i> <i>box.spine_engine_manager.RemoteSpineEngineManager</i> method), 622	<code>clear_icons_and_links()</code> (<i>spinetool-</i> <i>box.widgets.custom_qgraphicsscene.DesignGraphicsScene</i> method), 421
<code>clean_up()</code> (<i>spinetool-</i> <i>box.spine_engine_worker.SpineEngineWorker</i> method), 627	<code>clear_model()</code> (<i>spinetool-</i> <i>box.mvcmodels.compound_table_model.CompoundWithEmptyTab</i> method), 180
<code>cleanupPage()</code> (<i>spinetool-</i> <i>box.widgets.add_up_spine_opt_wizard.CheckPreviousInstallPage</i> method), 406	<code>clear_model()</code> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTable</i> method), 255
<code>cleanupPage()</code> (<i>spinetool-</i> <i>box.widgets.add_up_spine_opt_wizard.TroubleshootSolutionPage</i> method), 407	<code>clear_model()</code> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.pivot_model.PivotModel</i> method), 279
<code>cleanupPage()</code> (<i>spinetool-</i> <i>box.widgets.custom_qwidgets.QWizardProcessPage</i> method), 441	<code>clear_model()</code> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableM</i> method), 286
<code>cleanupPage()</code> (<i>spinetool-</i> <i>box.widgets.install_julia_wizard.InstallJuliaPage</i> method), 451	<code>clear_notifications()</code> (<i>spinetool-</i> <i>box.project_item.project_item.ProjectItem</i> method), 221
<code>clear()</code> (<i>spinetoolbox.mvcmodels.empty_row_model.EmptyRowModel</i> method), 181	<code>clear_notifications()</code> (<i>spinetool-</i> <i>box.project_item_icon.ExclamationIcon</i> method), 582
<code>clear()</code> (<i>spinetoolbox.mvcmodels.map_model.MapModel</i> method), 189	<code>clear_pivot_table()</code> (<i>spinetool-</i> <i>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMi</i> method), 388
<code>clear()</code> (<i>spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel</i> method), 193	<code>clear_pivot_table()</code> (<i>spinetool-</i> <i>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMi</i> method), 388
<code>clear()</code> (<i>spinetoolbox.mvcmodels.project_item_specification_model.ProjectItemSpecificationModel</i> method), 204	<code>clear_selected()</code> (<i>spinetool-</i> <i>box.ui_main.ToolboxUI</i> method), 637
<code>clear()</code> (<i>spinetoolbox.spine_db_editor.mvcmodels.item_model.ItemModel</i> method), 258	<code>clear_sessions()</code> (<i>spinetool-</i> <i>box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphi</i> method), 337
<code>clear()</code> (<i>spinetoolbox.widgets.custom_qtextbrowser.CustomQTextBrowser</i> method), 433	<code>clear_selected()</code> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTable</i> method), 255
<code>clear_all_filters()</code> (<i>spinetool-</i> <i>box.spine_db_editor.widgets.parameter_view_mixin.ParameterViewMi</i> method), 371	<code>clear_ui()</code> (<i>spinetoolbox.ui_main.ToolboxUI</i> method), 631
<code>clear_any_selections()</code> (<i>spinetool-</i> <i>box.spine_db_editor.widgets.custom_qtreeview.EntityTreeView</i> method), 349	<code>ClientSecurityModel</code> (class in <i>spinetool-</i> <i>box.server.engine_client</i>), 231
<code>clear_auto_filter()</code> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel</i> method), 240	<code>clone()</code> (<i>spinetoolbox.spine_db_commands.AgedUndoCommand</i> method), 504
<code>clear_children()</code> (<i>spinetool-</i> <i>box.mvcmodels.minimal_tree_model.TreeItem</i> method), 197	<code>clone()</code> (<i>spinetoolbox.widgets.project_item_drag.ShadeProjectItemSpecBu</i> method), 487
<code>clear_children()</code> (<i>spinetool-</i> <i>box.mvcmodels.minimal_tree_model.TreeItem</i> method), 197	<code>close()</code> (<i>spinetoolbox.server.engine_client.EngineClient</i> method), 234
<code>clear_children()</code> (<i>spinetool-</i> <i>box.mvcmodels.minimal_tree_model.TreeItem</i> method), 197	<code>close_all_sessions()</code> (<i>spinetool-</i> <i>box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphi</i> method), 337

box.spine_db_manager.SpineDBManager (class in *spinetool-*
method), 599

close_db_map() (*spinetool-*
box.spine_db_worker.SpineDBWorker *method*),
616

close_editor() (*spinetool-*
box.spine_db_editor.widgets.custom_delegates.ManageFromIndex
method), 331

close_editor() (*spinetool-*
box.spine_db_editor.widgets.object_name_list_editor.ObjectNameListEditor
method), 370

close_project() (*spinetoolbox.ui_main.ToolboxUI*
method), 630

close_session() (*spinetool-*
box.spine_db_manager.SpineDBManager
method), 599

closeEvent() (*spinetool-*
box.project_item.specification_editor_window.SpecificationEditorWindow
method), 230

closeEvent() (*spinetool-*
box.spine_db_editor.widgets.commit_viewer.CommitViewer *method*), 291
method), 321

closeEvent() (*spinetool-*
box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin
method), 362

closeEvent() (*spinetool-*
box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor
method), 382

closeEvent() (*spinetoolbox.ui_main.ToolboxUI*
method), 637

closeEvent() (*spinetool-*
box.widgets.about_widget.AboutWidget
method), 403

closeEvent() (*spinetool-*
box.widgets.add_project_item_widget.AddProjectItemWidget
method), 404

closeEvent() (*spinetool-*
box.widgets.jupyter_console_widget.JupyterConsoleWidget
method), 454

closeEvent() (*spinetool-*
box.widgets.multi_tab_window.MultiTabWindow
method), 468

closeEvent() (*spinetool-*
box.widgets.open_project_widget.OpenProjectDialog
method), 473

closeEvent() (*spinetool-*
box.widgets.persistent_console_widget.PersistentConsoleWidget
method), 477

closeEvent() (*spinetool-*
box.widgets.plot_widget.PlotWidget *method*),
483

closeEvent() (*spinetool-*
box.widgets.settings_widget.SettingsWidget
method), 497

CodeTextEdit (class in *spinetool-*
box.widgets.code_text_edit), 410

color() (*spinetoolbox.helpers.ColoredIcon* *method*),
523

color() (*spinetoolbox.helpers.ColoredIconEngine*
method), 524

color_from_index() (in module *spinetoolbox.helpers*),
528

color_pixmap() (in module *spinetoolbox.helpers*), 524

ColoredIconDelegate (in *spinetoolbox.helpers*), 523

ColoredIconEngine (class in *spinetoolbox.helpers*),
524

Column (class in *spinetool-*
box.spine_db_editor.mvcmodels.metadata_table_model_base),
262

COLUMN_COUNT (*spinetool-*
box.widgets.select_database_items.SelectDatabaseItems
method), 491

column_is_index_column() (*spinetool-*
box.spine_db_editor.mvcmodels.pivot_table_models.IndexExpans
method), 291

column_is_index_column() (*spinetool-*
box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableM
method), 288

column_key() (*spinetool-*
box.spine_db_editor.mvcmodels.pivot_model.PivotModel
method), 281

column_name() (*spinetool-*
box.spine_db_editor.mvcmodels.pivot_table_models.ParameterVa
method), 290

columnCount() (*spinetool-*
box.mvcmodels.array_model.ArrayModel
method), 176

columnCount() (*spinetool-*
box.mvcmodels.file_list_models.FileListModel
method), 182

columnCount() (*spinetool-*
box.mvcmodels.indexed_value_table_model.IndexedValueTableM
method), 187

columnCount() (*spinetool-*
box.mvcmodels.map_model.MapModel
method), 189

columnCount() (*spinetool-*
box.mvcmodels.minimal_table_model.MinimalTableModel
method), 194

columnCount() (*spinetool-*
box.mvcmodels.minimal_tree_model.MinimalTreeModel
method), 198

columnCount() (*spinetool-*
box.mvcmodels.project_item_model.ProjectItemModel
method), 200

columnCount() (*spinetool-*
box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTable
method), 256

columnCount() (spinetool- CompoundObjectParameterMixin (class in spinetool-
box.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase
method), 263 242

columnCount() (spinetool- CompoundObjectParameterValueModel
box.spine_db_editor.mvcmodels.multi_db_tree_model.MultiDBTreeModel in spinetool-
method), 270 box.spine_db_editor.mvcmodels.compound_parameter_models),

columnCount() (spinetool- 243
box.spine_db_editor.mvcmodels.parameter_value.CompoundParameterDefinitionModel
method), 279 (class in spinetool-
box.spine_db_editor.mvcmodels.compound_parameter_models),

columnCount() (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models),
box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModelBase
method), 287 242

columnCount() (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models),
box.spine_db_editor.mvcmodels.tree_model_base.TreeModelBase
method), 311 CompoundParameterValueMixin (class in spinetool-
box.spine_db_editor.mvcmodels.compound_parameter_models),

columnCount() (spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models),
box.widgets.open_project_widget.CustomQFileSystemModel 242
method), 473 CompoundRelationshipParameterDefinitionModel

columns (spinetoolbox.spine_db_editor.mvcmodels.pivot_model.PivotModel in spinetool-
property), 279 box.spine_db_editor.mvcmodels.compound_parameter_models),

combine_data_with_same_indexes() (in module 243
spinetoolbox.plotting), 549 CompoundRelationshipParameterMixin

combobox_text_edited() (spinetool- (class in spinetool-
box.widgets.open_project_widget.OpenProjectDialog box.spine_db_editor.mvcmodels.compound_parameter_models),
method), 472 242

ComboBoxDelegate (class in spinetool- CompoundRelationshipParameterValueModel
box.widgets.custom_delegates), 412 (class in spinetool-
box.mvcmodels.file_list_models), 183 box.spine_db_editor.mvcmodels.compound_parameter_models),

CommandLineArgItem (class in spinetool- 243
box.mvcmodels.file_list_models), 183 CompoundTableModel (class in spinetool-
box.mvcmodels.file_list_models), 183 box.mvcmodels.compound_table_model),

commands() (spinetool- 177
box.spine_db_commands.AgedUndoStack CompoundWithEmptyTableModel (class in spinetool-
method), 590 box.mvcmodels.compound_table_model), 179

commit() (spinetoolbox.spine_db_commands.AgedUndoStack.commit_button() (spinetool-
method), 590 box.project_item_icon.ProjectItemIcon
method), 578

commit_session() (spinetool- box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorEditor_signals() (spinetool-
method), 380 box.spine_db_editor.widgets.custom_delegates.ManageItemsDeleteDialog), 331

commit_session() (spinetool-
box.spine_db_manager.SpineDBManager connect_pull_socket() (spinetool-
method), 601 box.server.engine_client.EngineClient method),
232

commit_session() (spinetool-
box.spine_db_worker.SpineDBWorker method), connect_signals() (spinetool-
617 box.spine_db_editor.widgets.add_items_dialogs.AddItemDialog
method), 315

CommitDialog (class in spinetool- connect_signals() (spinetool-
box.widgets.commit_dialog), 411 box.spine_db_editor.widgets.add_items_dialogs.AddObjectClassesDialog), 316

CommitViewer (class in spinetool- connect_signals() (spinetool-
box.spine_db_editor.widgets.commit_viewer), 321 box.spine_db_editor.widgets.add_items_dialogs.AddOrManageRelationshipsDialog), 317

CompoundObjectParameterDefinitionModel connect_signals() (spinetool-
(class in spinetool- box.spine_db_editor.widgets.add_items_dialogs.AddReadyRelationshipsDialog), 317
box.spine_db_editor.mvcmodels.compound_parameter_models), 243

method), 315

connect_signals() (spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddRelationshipClassesDialog method), 317

connect_signals() (spinetoolbox.spine_db_editor.widgets.add_items_dialogs.ManageRelationshipClassesDialog method), 318

connect_signals() (spinetoolbox.spine_db_editor.widgets.add_items_dialogs.ObjectGroupDialogBase method), 319

connect_signals() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.AlternativeTreeView method), 352

connect_signals() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349

connect_signals() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ItemTreeView method), 351

connect_signals() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView method), 350

connect_signals() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ScenarioTreeView method), 353

connect_signals() (spinetoolbox.spine_db_editor.widgets.edit_or_remove_items_dialogs.EditObjectClassesDialog method), 356

connect_signals() (spinetoolbox.spine_db_editor.widgets.edit_or_remove_items_dialogs.EditRelationshipClassesDialog method), 357

connect_signals() (spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin method), 359

connect_signals() (spinetoolbox.spine_db_editor.widgets.item_metadata_editor.ItemMetadataEditor method), 362

connect_signals() (spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ManageItemsDialog method), 364

connect_signals() (spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.ManageItemsDialogBase method), 364

connect_signals() (spinetoolbox.spine_db_editor.widgets.metadata_editor.MetadataEditor method), 367

connect_signals() (spinetoolbox.spine_db_editor.widgets.parameter_view_mixin.ParameterViewMixin method), 371

connect_signals() (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 382

connect_signals() (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase method), 379

connect_signals() (spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin method), 385

connect_signals() (spinetoolbox.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin method), 390

connect_signals() (spinetoolbox.ui_main.ToolboxUI method), 408

connect_signals() (spinetoolbox.widgets.add_project_item_widget.AddProjectItemWidget method), 404

connect_signals() (spinetoolbox.widgets.custom_editors.IconColorEditor method), 417

connect_signals() (spinetoolbox.widgets.custom_menus.FilterMenuBase method), 420

connect_signals() (spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene method), 422

connect_signals() (spinetoolbox.widgets.custom_qwidgets.FilterWidget method), 437

connect_signals() (spinetoolbox.widgets.kernel_editor.KernelEditorBase method), 444

connect_signals() (spinetoolbox.widgets.multi_tab_window.MultiTabWindow method), 465

connect_signals() (spinetoolbox.widgets.open_project_widget.OpenProjectDialog method), 471

connect_signals() (spinetoolbox.widgets.settings_widget.SettingsWidget method), 494

connect_signals() (spinetoolbox.widgets.settings_widget.SettingsWidgetBase method), 493

connect_signals() (spinetoolbox.widgets.settings_widget.SpineDBEditorSettingsMixin method), 494

connect_spine_db_editor() (spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 336

connect_spine_db_editor() (spinetoolbox.spine_db_editor.widgets.custom_qtableview.MetadataTableView method), 337

connect_spine_db_editor() (spinetoolbox.spine_db_editor.widgets.custom_qtableview.ParameterTableView method), 339

connect_spine_db_editor() (spinetoolbox.spine_db_editor.widgets.custom_qtableview.ParameterValueTableView method), 341

<code>connect_spine_db_editor()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtableview.PivotTableView method), 346	564	<code>connections_to_dict()</code>	(spinetool- box.headless.ModifiableProject method),
<code>connect_spine_db_editor()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtreeview.AllocatorButton method), 352	513	<code>ConnectorButton</code>	(class in spinetool- box.project_item_icon), 580
<code>connect_spine_db_editor()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349	<code>console_closed</code>	(spinetool- box.widgets.jupyter_console_widget.JupyterConsoleWidget attribute), 453	
<code>connect_spine_db_editor()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtreeview.ItemTreeView method), 351	<code>content</code>	(spinetoolbox.plotting.TreeNode attribute), 547	
<code>connect_spine_db_editor()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtreeview.PivotTableHeaderView method), 354	<code>contextMenuRequested</code>	(spinetool- box.spine_db_editor.widgets.pivot_table_header_view.ScenarioAl attribute), 374	
<code>connect_spine_db_editor()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtreeview.ScenarioTreeView method), 353	<code>ContextMenuBaseView</code>	(class in spinetool- box.widgets.indexed_value_table_context_menu),	
<code>connect_spine_db_editor()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtreeview.ScenarioTreeView method), 353	<code>ContextMenuEvent()</code>	(spinetoolbox.link.JumpOrLink method), 538	
<code>connect_spine_db_editor()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtreeview.ToolFeatureTreeView method), 351	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtreeview.ToolFeatureTreeView method), 579	
<code>connect_to_kernel()</code>	(spinetool- box.widgets.jupyter_console_widget.JupyterConsoleWidget method), 454	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.graphics_items.CrossHairsItem method), 400	
<code>connect_to_project()</code>	(spinetool- box.mvcmodels.project_item_model.ProjectItemModel method), 199	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.graphics_items.CrossHairsRelationshipItem method), 400	
<code>connect_to_project()</code>	(spinetool- box.mvcmodels.project_item_specification_models.ProjectItemSpecificationModel method), 204	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.graphics_items.EntityItem method), 396	
<code>CONNECTION</code>	(spinetoolbox.helpers.LinkType attribute), 519	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraph method), 338	
<code>connection</code>	(spinetoolbox.link.Link property), 539	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtableview.MetadataTableVi method), 347	
<code>connection</code>	(spinetool- box.mvcmodels.resource_filter_model.ResourceFilterModel property), 208	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtableview.ParameterTableV method), 340	
<code>connection_about_to_be_removed</code>	(spinetool- box.project.SpineToolboxProject attribute), 558	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtableview.PivotTableView method), 346	
<code>connection_established</code>	(spinetool- box.project.SpineToolboxProject attribute), 558	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtableview.PivotTableView method), 346	
<code>connection_from_dict()</code>	(spinetool- box.project.SpineToolboxProject method), 560	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349	
<code>connection_updated</code>	(spinetool- box.project.SpineToolboxProject attribute), 558	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtreeview.ItemTreeView method), 351	
<code>ConnectionLinkDrawer</code>	(class in spinetoolbox.link), 540	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.widgets.pivot_table_header_view.ParameterV method), 374	
<code>connections</code>	(spinetoolbox.project.SpineToolboxProject property), 558	<code>ContextMenuEvent()</code>	(spinetool- box.spine_db_editor.widgets.pivot_table_header_view.ScenarioAl method), 374	
<code>connections_for_item()</code>	(spinetool- box.project.SpineToolboxProject method),			

contextMenuEvent() (spinetool- method), 224
 box.widgets.custom_qgraphicsviews.DesignQGraphicsView.plot_data() (spinetool-
 method), 426 box.widgets.plot_widget.PlotWidget method),
 contextMenuEvent() (spinetool- 483
 box.widgets.custom_qtextbrowser.CustomQTextBrowser.copy_to_clipboard() (spinetool-
 method), 433 box.widgets.about_widget.AboutWidget
 contextMenuEvent() (spinetool- method), 403
 box.widgets.multi_tab_window.TabBarPlus CopyPasteTableView (class in spinetool-
 method), 469 box.widgets.custom_qtableview), 427
 contextMenuEvent() (spinetool- CopyPasteTreeView (class in spinetool-
 box.widgets.persistent_console_widget.PersistentConsoleWidget), 434
 method), 480 create_data_dir() (spinetool-
 contextMenuEvent() (spinetool- box.project_item.project_item.ProjectItem
 method), 484 method), 220
 box.widgets.plot_widget._PlotDataView create_db_file() (spinetool-
 contextMenuEvent() (spinetool- box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase
 method), 483 method), 379
 box.widgets.plot_widget.PlotWidget create_delegates() (spinetool-
 contextMenuEvent() (spinetool- box.spine_db_editor.widgets.custom_qtableview.ObjectParameter
 method), 487 method), 341
 box.widgets.project_item_drag.ProjectItemSpecButton create_delegates() (spinetool-
 method), 487 method), 342
 convert_indexed_value_to_tree() (in module box.spine_db_editor.widgets.custom_qtableview.ObjectParameter
 spinetoolbox.plotting), 548 method), 342
 convert_leaf_maps() (spinetool- create_delegates() (spinetool-
 box.mvcmodels.map_model.MapModel box.spine_db_editor.widgets.custom_qtableview.ParameterDefini
 method), 189 method), 341
 ConvertToDBMixin (class in spinetool- create_delegates() (spinetool-
 box.spine_db_editor.mvcmodels.parameter_mixins), box.spine_db_editor.widgets.custom_qtableview.ParameterTableV
 271 method), 340
 copy() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.create_delegatesView (spinetool-
 method), 353 box.spine_db_editor.widgets.custom_qtableview.ParameterValueT
 copy() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.SceneWidgetView
 method), 353 create_delegates() (spinetool-
 copy() (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase editor.widgets.custom_qtableview.RelationshipPara
 method), 379 method), 341
 copy() (spinetoolbox.widgets.custom_qtableview.ArrayTableView create_delegates() (spinetool-
 method), 431 box.spine_db_editor.widgets.custom_qtableview.RelationshipPara
 copy() (spinetoolbox.widgets.custom_qtableview.CopyPasteTableView method), 342
 method), 428 create_dir() (in module spinetoolbox.helpers), 520
 copy() (spinetoolbox.widgets.custom_qtableview.IndexedParameterTableView create_delegatesView (spinetool-
 method), 429 box.spine_db_editor.widgets.custom_qtableview.ParameterValueT
 copy() (spinetoolbox.widgets.custom_qtableview.MapTableView method), 479
 method), 431 create_engine_worker() (spinetool-
 copy() (spinetoolbox.widgets.custom_qtreeview.CopyPasteTreeView box.project.SpineToolboxProject method),
 method), 434 566
 copy_action (spinetool- create_filter_menu() (spinetool-
 box.widgets.custom_qtableview.CopyPasteTableView box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMi
 property), 428 method), 389
 copy_files() (in module spinetoolbox.helpers), 521 create_header_widget() (spinetool-
 copy_input() (spinetool- box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMi
 box.widgets.jupyter_console_widget.JupyterConsoleWidget method), 389
 method), 454 create_new_spine_database() (spinetool-
 copy_local_data() (spinetool- box.spine_db_manager.SpineDBManager
 box.project_item.project_item.ProjectItem method), 599

`create_project()` (`spinetoolbox.ui_main.ToolboxUI` `box.spine_db_editor.widgets.custom_delegates.RelationshipClass`
`method`), 629 `method`), 327
`createEditor()` (`spinetool- createEditor()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.AlternativeDbDelegate` `box.spine_db_editor.widgets.custom_delegates.RelationshipPivot`
`method`), 330 `method`), 323
`createEditor()` (`spinetool- createEditor()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.AlternativeDbDelegate` `box.spine_db_editor.widgets.custom_delegates.RemoveEntitiesDe`
`method`), 328 `method`), 333
`createEditor()` (`spinetool- createEditor()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.DatabaseNameDelegate` `box.spine_db_editor.widgets.custom_delegates.ScenarioAlternativ`
`method`), 325 `method`), 324
`createEditor()` (`spinetool- createEditor()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ItemMetadataDelegate` `box.spine_db_editor.widgets.custom_delegates.ScenarioDelegate`
`method`), 333 `method`), 330
`createEditor()` (`spinetool- createEditor()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ManageItemDelegate` `box.spine_db_editor.widgets.custom_delegates.ToolFeatureDeleg`
`method`), 332 `method`), 329
`createEditor()` (`spinetool- createEditor()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ManageObjectClassDelegate` `box.spine_db_editor.widgets.custom_delegates.ValueListDelegate`
`method`), 332 `method`), 327
`createEditor()` (`spinetool- createEditor()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ManageObjectClassDelegate` `box.spine_db_editor.widgets.object_name_list_editor.SearchBarD`
`method`), 332 `method`), 370
`createEditor()` (`spinetool- createEditor()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ManageRelationshipClassDelegate` `box.spine_db_editor.widgets.select_position_parameters_dialog.P`
`method`), 332 `method`), 378
`createEditor()` (`spinetool- createEditor()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ManageRelationshipClassDelegate` `box.spine_db_editor.widgets.CheckBoxDelegate`
`method`), 333 `method`), 412
`createEditor()` (`spinetool- createEditor()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ObjectClassNameDelegate` `box.spine_db_editor.widgets.ComboBoxDelegate`
`method`), 327 `method`), 412
`createEditor()` (`spinetool- createEditor()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ObjectNameDelegate` `box.spine_db_editor.widgets.custom_editors._CustomLineEditDelegate`
`method`), 328 `method`), 414
`createEditor()` (`spinetool- CrossHairsArcItem` (`class in spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ObjectNameDelegate` `box.spine_db_editor.graphics_items`), 400
`method`), 328 `CrossHairsItem` (`class in spinetool-`
`createEditor()` (`spinetool- box.spine_db_editor.graphics_items`), 399
`box.spine_db_editor.widgets.custom_delegates.PanCrossHairsDelegate` `CrossHairsRelationshipItem` (`class in spinetool-`
`method`), 327 `box.spine_db_editor.graphics_items`), 400
`createEditor()` (`spinetool- current_changed()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ParameterPivotWidgetDelegate` `box.spine_db_editor.widgets.OpenProjectDialog`
`method`), 324 `method`), 472
`createEditor()` (`spinetool- current_index_changed()` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ParameterValueDelegate` `box.spine_db_editor.widgets.OpenProjectDialog`
`method`), 324 `method`), 471
`createEditor()` (`spinetool- current_object_class_id_list` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ParameterValueDelegate` `box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMi`
`method`), 331 `property`), 384
`createEditor()` (`spinetool- current_object_class_ids` (`spinetool-`
`box.spine_db_editor.widgets.custom_delegates.ParameterValueDelegate` `box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMi`
`method`), 326 `property`), 384
`createEditor()` (`spinetool- current_object_class_name_list` (`spinetool-`

box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin (class in spinetoolbox.widgets.tabular_view_mixin), 384

currentChanged() (spinetoolbox.widgets.custom_editors.SearchBarEditor method), 415

custom_context_menu() (spinetoolbox.mvcmodels.project_tree_item.BaseProjectTreeItem method), 206

custom_context_menu() (spinetoolbox.mvcmodels.project_tree_item.CategoryProjectTreeItem method), 207

custom_context_menu() (spinetoolbox.mvcmodels.project_tree_item.LeafProjectTreeItem method), 208

custom_context_menu() (spinetoolbox.mvcmodels.project_tree_item.RootProjectTreeItem method), 207

CustomContextMenu (class in spinetoolbox.widgets.custom_menus), 417

CustomGraphicsScene (class in spinetoolbox.widgets.custom_qgraphicsscene), 421

CustomLineEditor (class in spinetoolbox.widgets.custom_editors), 413

CustomPopupMenu (class in spinetoolbox.widgets.custom_menus), 418

CustomQComboBox (class in spinetoolbox.widgets.custom_qcombobox), 420

CustomQFileSystemModel (class in spinetoolbox.widgets.open_project_widget), 473

CustomQGraphicsView (class in spinetoolbox.widgets.custom_qgraphicsviews), 423

CustomQTextBrowser (class in spinetoolbox.widgets.custom_qtextbrowser), 432

CustomSyntaxHighlighter (class in spinetoolbox.helpers), 531

CustomTreeView (class in spinetoolbox.widgets.custom_qtreeview), 435

CustomWidgetAction (class in spinetoolbox.widgets.custom_qwidgets), 437

D

dag_with_node() (spinetoolbox.project.SpineToolboxProject method), 565

data (spinetoolbox.spine_db_parcel.SpineDBParcel property), 611

data() (spinetoolbox.mvcmodels.array_model.ArrayModel method), 176

data() (spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel method), 179

data() (spinetoolbox.mvcmodels.file_list_models.FileListModel method), 182

data() (spinetoolbox.mvcmodels.filter_checkbox_list_model.FilterCheckboxListModel method), 186

data() (spinetoolbox.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel method), 185

data() (spinetoolbox.mvcmodels.filter_execution_model.FilterExecutionModel method), 186

data() (spinetoolbox.mvcmodels.indexed_value_table_model.IndexedValueTableModel method), 187

data() (spinetoolbox.mvcmodels.map_model.MapModel method), 189

data() (spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel method), 194

data() (spinetoolbox.mvcmodels.minimal_tree_model.MinimalTreeModel method), 199

data() (spinetoolbox.mvcmodels.minimal_tree_model.TreeItem method), 197

data() (spinetoolbox.mvcmodels.project_item_model.ProjectItemModel method), 200

data() (spinetoolbox.mvcmodels.project_item_specification_models.ProjectItemSpecificationModel method), 204

data() (spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_model.EmptyParameterModel method), 245

data() (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.EntityTreeItem method), 249

data() (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.EntityTreeItem method), 252

data() (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.MemberEntityTreeItem method), 252

data() (spinetoolbox.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel method), 257

data() (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel method), 263

data() (spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDbTreeItem method), 269

data() (spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item.ParameterValueListItem method), 278

data() (spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item.ParameterValueListItem method), 278

data() (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 289

data() (spinetoolbox.spine_db_editor.mvcmodels.single_parameter_model.SingleParameterModel method), 299

data() (spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 305

data() (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.BoldTextTreeItemUtility method), 309

data() (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.GrayIfLeafTreeItemUtility method), 309

data() (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafIfBoldTextTreeItemUtility method), 310

data() (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeItemUtility method), 310

data() (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeItemUtility method), 308

data() (spinetoolbox.widgets.custom_editors.CheckListEditor method), 416

data() (spinetoolbox.widgets.custom_editors.CustomLineEditBase (spinetool-
 method), 413 box.project_item.logging_connection.HeadlessConnection
 data() (spinetoolbox.widgets.custom_editors.IconColorEditor property), 216
 method), 417 DatabaseNameDelegate (class in spinetool-
 data() (spinetoolbox.widgets.custom_editors.ParameterValueLineEdit box.spine_db_editor.widgets.custom_delegates),
 method), 414 325
 data() (spinetoolbox.widgets.custom_editors.SearchBarEditor dataColumnCount() (spinetool-
 method), 415 box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableM
 data() (spinetoolbox.widgets.plugin_manager_widgets.InstallPluginModel), 287
 method), 485 dataRowCount() (spinetool-
 data_committed (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableM
 box.spine_db_editor.widgets.custom_delegates.AlternativeDele (class in spinetool-
 attribute), 329 DataToValueFilterCheckboxListModel
 data_committed (spinetool- (class in spinetool-
 box.spine_db_editor.widgets.custom_delegates.ManageItemDelegates (class in spinetool-
 attribute), 331 186
 data_committed (spinetool- DATETIME (spinetoolbox.widgets.parameter_value_editor_base.ValueType
 box.spine_db_editor.widgets.custom_delegates.ParameterDataDelegate), 475
 attribute), 325 DatetimeEditor (class in spinetool-
 data_committed (spinetool- box.widgets.datetime_editor), 443
 box.spine_db_editor.widgets.custom_delegates.ParameterEditorBaseDelegate changed (spinetool-
 attribute), 324 box.spine_db_editor.widgets.url_toolbar._DBListWidget
 data_committed (spinetool- attribute), 393
 box.spine_db_editor.widgets.custom_delegates.ParameterEditorBaseDelegate (spinetool-
 attribute), 331 db_item() (spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_m
 data_committed (spinetool- db_item() (spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_m
 box.spine_db_editor.widgets.custom_delegates.RelationshipFieldDelegate
 attribute), 323 db_item() (spinetoolbox.spine_db_editor.mvcmodels.single_parameter_m
 data_committed (spinetool- method), 299
 box.spine_db_editor.widgets.custom_delegates.ScenarioDelegate (spinetool-
 attribute), 323 db_items() (spinetoolbox.spine_db_editor.mvcmodels.tree_model_base.Tree
 data_committed (spinetool- db_item_from_id() (spinetool-
 box.spine_db_editor.widgets.custom_delegates.ScenarioDelegate (spinetool-
 attribute), 330 method), 299
 data_committed (spinetool- DB_ITEM_SEPARATOR (in module spinetoolbox.helpers),
 box.spine_db_editor.widgets.custom_delegates.ToolFeatureDelegate
 attribute), 328 db_items() (spinetool-
 data_committed (spinetool- box.spine_db_editor.mvcmodels.single_parameter_models.Single
 box.spine_db_editor.widgets.object_name_list_editor.SearchBarEditor (spinetool-
 attribute), 370 200
 data_committed (spinetool- DB_MAP (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_b
 box.widgets.custom_delegates.CheckBoxDelegate db_map (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem
 attribute), 412 property), 310
 data_committed (spinetool- db_map (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.RootItem
 box.widgets.custom_editors.SearchBarEditor property), 310
 attribute), 414 db_map() (spinetoolbox.spine_db_manager.SpineDBManager
 data_files() (spinetool- method), 599
 box.project_item.project_item.ProjectItem db_map_class_ids() (spinetool-
 method), 220 box.spine_db_manager.SpineDBManager
 data_index (spinetoolbox.plotting.XYData attribute), static method), 609
 547 db_map_data() (spinetool-
 data_submitted (spinetool- box.spine_db_editor.graphics_items.EntityItem
 box.spine_db_editor.widgets.mass_select_items_dialogs.MassSelectItemsDialog
 attribute), 367 db_map_data() (spinetool-

box.spine_db_editor.mvcmodels.entity_tree_item.MultiDBTreeItem (spinetoolbox.spine_db_editor.widgets.custom_delegates.ParameterModel (class in spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView), 325
 method), 251
 db_map_data() (spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView), 336
 box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem (class in spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item), 268
 method), 268
 db_map_data_field() (spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableModel (class in spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models), 346
 box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem (class in spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item), 494
 method), 268
 db_map_from_key() (spinetoolbox.spine_db_editor.widgets.settings_widget.SettingsWidget (class in spinetoolbox.spine_db_editor.widgets.settings_widget), 494
 box.spine_db_manager.SpineDBManager (class in spinetoolbox.spine_db_manager), 494
 method), 598
 db_map_id() (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorSettingsWidget (class in spinetoolbox.spine_db_editor.widgets.spine_db_editor), 378
 box.spine_db_editor.graphics_items.EntityItem (class in spinetoolbox.spine_db_editor.graphics_items), 378
 method), 395
 db_map_id() (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorSettingsWidget (class in spinetoolbox.spine_db_editor.widgets.spine_db_editor), 398
 box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel (class in spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_model), 397
 method), 241
 db_map_id() (spinetoolbox.spine_db_editor.mvcmodels.relationship_item.RelationshipItem (class in spinetoolbox.spine_db_editor.mvcmodels.relationship_item), 397
 box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem (class in spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item), 312
 method), 267
 db_map_ids (spinetoolbox.spine_db_editor.mvcmodels.tree_model_base.TreeModelBase (class in spinetoolbox.spine_db_editor.mvcmodels.tree_model_base), 395
 box.spine_db_editor.graphics_items.EntityItem (class in spinetoolbox.spine_db_editor.graphics_items), 378
 property), 395
 db_map_ids (spinetoolbox.spine_db_editor.mvcmodels.tree_model_base.TreeModelBase (class in spinetoolbox.spine_db_editor.mvcmodels.tree_model_base), 596
 box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem (class in spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item), 235
 property), 267
 db_map_ids() (spinetoolbox.spine_db_editor.mvcmodels.alternative_item.AlternativeItem (class in spinetoolbox.spine_db_editor.mvcmodels.alternative_item), 277
 box.spine_db_manager.SpineDBManager (class in spinetoolbox.spine_db_manager), 221
 static method), 609
 db_map_key() (spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item.ParameterValueListItem (class in spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item), 524
 box.spine_db_manager.SpineDBManager (class in spinetoolbox.spine_db_manager), 221
 static method), 598
 db_map_listeners() (spinetoolbox.spine_db_editor.mvcmodels.project_item.ProjectItem (class in spinetoolbox.spine_db_editor.mvcmodels.project_item), 221
 box.spine_db_manager.SpineDBManager (class in spinetoolbox.spine_db_manager), 221
 method), 600
 db_map_object_ids() (spinetoolbox.spine_db_editor.mvcmodels.entity_qgraphics_view.EntityQGraphicsView (class in spinetoolbox.spine_db_editor.widgets.entity_qgraphics_view), 336
 box.spine_db_editor.mvcmodels.pivot_table_models.ParameterModel (class in spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models), 336
 method), 290
 DB_MAP_ROLE (in module spinetoolbox.spine_db_editor.mvcmodels.shared), 210
 db_maps (spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem (class in spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item), 267
 spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem (class in spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item), 267
 property), 267
 db_maps (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel (class in spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models), 267
 spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel (class in spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models), 267
 property), 285
 db_maps (spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem (class in spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item), 267
 spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem (class in spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item), 267
 property), 596
 db_mgr (spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem (class in spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item), 267
 spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem (class in spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item), 267
 property), 266
 db_mgr (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel (class in spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models), 267
 spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel (class in spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models), 267
 property), 282
 db_mgr (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.TreeItemUtility (class in spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility), 398
 spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.TreeItemUtility (class in spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility), 398
 property), 308

<code>default_parameter_data()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.graphics_items.RelationshipItem</i> <i>method</i>), 397	<i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 601
<code>default_parameter_data()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.entity_tree_item.Miscellaneous()</i> (in module <i>spinetoolbox.helpers</i>), 530 <i>method</i>), 252	<i>DirValidator</i> (class in <i>spinetool-</i> <i>box.widgets.open_project_widget</i>), 473
<code>default_parameter_data()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.entity_tree_item.ObjectClassProperty()</i> , 196 <i>method</i>), 250	<i>display_data</i> (<i>spinetool-</i> <i>box.mvcmodels.minimal_tree_model.TreeItem</i> <i>property</i>), 196
<code>default_parameter_data()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.entity_tree_item.ObjectItemProperty()</i> , 395 <i>method</i>), 253	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.graphics_items.EntityItem</i> <i>property</i>), 395
<code>default_parameter_data()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.entity_tree_item.RelationshipProperty()</i> , 248 <i>method</i>), 251	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.entity_tree_item.EntityRootItem</i> <i>property</i>), 248
<code>default_parameter_data()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.entity_tree_item.RelationshipProperty()</i> , 253 <i>method</i>), 253	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.entity_tree_item.MemberObjectItem</i> <i>property</i>), 253
<code>default_parameter_data()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem()</i> , 251 <i>method</i>), 269	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.entity_tree_item.RelationshipItem</i> <i>property</i>), 253
<code>DEFAULT_WORK_DIR</code> (in module <i>spinetoolbox.config</i>), 504		<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem</i> <i>property</i>), 266
<code>del_key_pressed</code>	(<i>spinetool-</i> <i>box.widgets.custom_qtreeview.CustomTreeView</i> <i>attribute</i>), 435	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tool_feature_item.FeatureRootItem</i> <i>property</i>), 303
<code>del_key_pressed</code>	(<i>spinetool-</i> <i>box.widgets.custom_qtreeview.SourcesTreeView</i> <i>attribute</i>), 435	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureMe</i> <i>property</i>), 305
<code>delete_content()</code>	(<i>spinetool-</i> <i>box.widgets.custom_qtableview.CopyPasteTableView</i> <i>method</i>), 428	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureRo</i> <i>property</i>), 304
<code>delete_content()</code>	(<i>spinetool-</i> <i>box.widgets.custom_qtableview.MapTableView</i> <i>method</i>), 431	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tool_feature_item.ToolRootItem</i> <i>property</i>), 303
<code>descendant_names()</code>	(<i>spinetool-</i> <i>box.project.SpineToolboxProject</i> <i>method</i>), 568	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeIte</i> <i>property</i>), 308
<code>description</code>	(<i>spinetool-</i> <i>box.widgets.set_description_dialog.SetDescriptionDialog</i> <i>property</i>), 492	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeIte</i> <i>property</i>), 308
<code>description()</code>	(<i>spinetool-</i> <i>box.project_item.specification_editor_window.SpecificationEditorWindow</i> <i>method</i>), 231	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeIte</i> <i>property</i>), 308
<code>DesignGraphicsScene</code>	(class in <i>spinetool-</i> <i>box.widgets.custom_qgraphicsscene</i>), 421	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeIte</i> <i>property</i>), 308
<code>DesignQGraphicsView</code>	(class in <i>spinetool-</i> <i>box.widgets.custom_qgraphicsviews</i>), 425	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeIte</i> <i>property</i>), 308
<code>detach()</code>	(<i>spinetoolbox.widgets.multi_tab_window.MultiTabWindow</i> <i>method</i>), 466	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeIte</i> <i>property</i>), 308
<code>dir_is_valid()</code> (in module <i>spinetoolbox.helpers</i>), 527		<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeIte</i> <i>property</i>), 308
<code>dirname</code> (in module <i>spinetoolbox.main</i>), 544		<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeIte</i> <i>property</i>), 308
<code>dirty()</code>	(<i>spinetoolbox.spine_db_manager.SpineDBManager</i> <i>method</i>), 600	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeIte</i> <i>property</i>), 308
<code>dirty_and_without_editors()</code>	(<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.entity_tree_item.EntityClassItem</i> <i>property</i>), 249	<i>display_data</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.entity_tree_item.EntityClassItem</i> <i>property</i>), 249

box.spine_db_editor.mvcmodels.entity_tree_item.EntityItem method), 426
 property), 252 do_update_link() (spinetool-
 display_icon (spinetool- box.widgets.custom_qgraphicsviews.DesignQGraphicsView
 box.spine_db_editor.mvcmodels.entity_tree_item.EntityRootItem method), 425
 property), 248 do_work() (spinetoolbox.spine_engine_worker.SpineEngineWorker
 display_icon (spinetool- method), 626
 box.spine_db_editor.mvcmodels.entity_tree_item.EntityRootItem method), 425
 property), 253 DOCUMENTATION_PATH (in module spinetoolbox.config),
 504
 display_icon (spinetool- done() (spinetoolbox.widgets.open_project_widget.OpenProjectDialog
 box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem method), 472
 property), 266 double_clicked (spinetool-
 display_icon (spinetool- box.widgets.project_item_drag.ProjectItemButton
 box.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeItem method), 487
 property), 308 download_files() (spinetool-
 display_id (spinetool- box.server.engine_client.EngineClient method),
 box.spine_db_editor.mvcmodels.entity_tree_item.EntityRootItem method), 425
 property), 248 downstream_resources_updated() (spinetool-
 display_id (spinetool- box.project_item.project_item.ProjectItem
 box.spine_db_editor.mvcmodels.entity_tree_item.MembersItem method), 223
 property), 251 drag_about_to_start (spinetool-
 display_id (spinetool- box.widgets.project_item_drag.ProjectItemDragMixin
 box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem method), 486
 property), 266 dragEnterEvent() (spinetool-
 do_add_jump() (spinetool- box.spine_db_editor.widgets.custom_qtableview.FrozenTableView
 box.widgets.custom_qgraphicsviews.DesignQGraphicsView method), 346
 method), 426 dragEnterEvent() (spinetool-
 do_add_link() (spinetool- box.spine_db_editor.widgets.custom_qtreeview.ScenarioTreeView
 box.widgets.custom_qgraphicsviews.DesignQGraphicsView method), 353
 method), 425 dragEnterEvent() (spinetool-
 do_create_new_spine_database() (in module spine- box.spine_db_editor.widgets.custom_qtreeview.ToolFeatureTreeVi
 toolbox.spine_db_manager), 596 method), 352
 do_reload_pivot_table() (spinetool- dragEnterEvent() (spinetool-
 box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin method), 373
 method), 388 do_reload_pivot_table() (in module spine-
 box.spine_db_editor.widgets.pivot_table_header_view.PivotTable
 method), 373
 do_remove_items() (spinetool- dragEnterEvent() (spinetool-
 box.spine_db_manager.SpineDBManager box.spine_db_editor.widgets.tabular_view_header_widget.Tabula
 method), 609 method), 384
 do_remove_jump() (spinetool- dragEnterEvent() (spinetool-
 box.widgets.custom_qgraphicsviews.DesignQGraphicsView box.widgets.custom_qgraphicsscene.DesignGraphicsScene
 method), 426 method), 422
 do_remove_link() (spinetool- dragEnterEvent() (spinetool-
 box.widgets.custom_qgraphicsviews.DesignQGraphicsView box.widgets.custom_qtreeview.SourcesTreeView
 method), 425 method), 435
 do_set_description() (spinetool- dragEnterEvent() (spinetool-
 box.project_item.specification_editor_window.SpecNameDescriptionEditor console_widget.JupyterConsoleWidget
 method), 231 method), 454
 do_set_name() (spinetool- dragEnterEvent() (spinetool-
 box.project_item.specification_editor_window.SpecNameDescriptionEditor console_widget.JupyterConsoleWidget
 method), 230 method), 503
 do_set_specification() (spinetool- dragLeaveEvent() (spinetool-
 box.project_item.project_item.ProjectItem box.widgets.custom_qgraphicsscene.DesignGraphicsScene
 method), 221 method), 422
 do_update_jump() (spinetool- dragLeaveEvent() (spinetool-
 box.widgets.custom_qgraphicsviews.DesignQGraphicsView box.widgets.toolbars.MainToolBar method),

503
[dragMoveEvent\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtableview.FrozenTableView method), 346
[dragMoveEvent\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ScenarioTreeView method), 353
[dragMoveEvent\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ToolFeatureTreeView method), 352
[dragMoveEvent\(\)](#) (spinetoolbox.spine_db_editor.widgets.pivot_table_header_view.PivotTableHeaderView method), 373
[dragMoveEvent\(\)](#) (spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene method), 422
[dragMoveEvent\(\)](#) (spinetoolbox.widgets.custom_qtreeview.SourcesTreeView method), 435
[dragMoveEvent\(\)](#) (spinetoolbox.widgets.toolbars.MainToolBar method), 503
[drain\(\)](#) (spinetoolbox.helpers.HTMLTagFilter method), 533
[drawBackground\(\)](#) (spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene method), 422
[dropEvent\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtableview.FrozenTableView method), 346
[dropEvent\(\)](#) (spinetoolbox.spine_db_editor.widgets.pivot_table_header_view.PivotTableHeaderView method), 373
[dropEvent\(\)](#) (spinetoolbox.spine_db_editor.widgets.tabular_view_header_widget.TabularHeaderViewHeaderWidget method), 384
[dropEvent\(\)](#) (spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene method), 422
[dropEvent\(\)](#) (spinetoolbox.widgets.custom_qtreeview.SourcesTreeView method), 435
[dropEvent\(\)](#) (spinetoolbox.widgets.toolbars.MainToolBar method), 503
[dropMimeData\(\)](#) (spinetoolbox.mvcmodels.file_list_models.CommandLineArgsModel method), 184
[dropMimeData\(\)](#) (spinetoolbox.spine_db_editor.mvcmodels.scenario_model.ScenarioModel method), 296
[dropMimeData\(\)](#) (spinetoolbox.spine_db_editor.mvcmodels.tool_feature_model.ToolFeatureModel method), 307
[dst_center](#) (spinetoolbox.link.LinkBase property), 536
[dst_center](#) (spinetoolbox.link.LinkDrawerBase property), 540
[dst_rect](#) (spinetoolbox.link.LinkBase property), 536
[dst_rect](#) (spinetoolbox.link.LinkDrawerBase property), 540
[duplicate_object\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView method), 350
[duplicate_object\(\)](#) (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase method), 386
[duplicate_object\(\)](#) (spinetoolbox.spine_db_manager.SpineDBManager method), 610
[duplicate_paths\(\)](#) (spinetoolbox.mvcmodels.file_list_models.FileListModel method), 183
[duplicate_project_item\(\)](#) (spinetoolbox.ui_main.ToolboxUI method), 637
[duplicate_scenario\(\)](#) (spinetoolbox.spine_db_editor.mvcmodels.scenario_model.ScenarioModel method), 297
[duplicate_scenario\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView method), 345
[duplicate_scenario\(\)](#) (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase method), 380
[duplicate_scenario\(\)](#) (spinetoolbox.spine_db_manager.SpineDBManager method), 610
[DURATION](#) (spinetoolbox.widgets.parameter_value_editor_base.ValueType attribute), 475
[DurationEditor](#) (spinetoolbox.widgets.duration_editor widget in spinetoolbox.widgets.duration_editor), 444
[E](#)
[edit\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349
[edit_data](#) (spinetoolbox.mvcmodels.minimal_tree_model.TreeItem property), 196
[edit_data](#) (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.Relation property), 253
[edit_entity_tree_items\(\)](#) (spinetoolbox.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin method), 391
[edit_first_index\(\)](#) (spinetoolbox.widgets.custom_editors.SearchBarEditor method), 415
[edit_selected\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 336

edit_selected() (spinetool- 188
 box.spine_db_editor.widgets.custom_qtreeview.EmptyChild() (spinetool-
 method), 350 box.spine_db_editor.mvcmodels.alternative_item.DBItem
 edit_specification() (spinetool- method), 235
 box.ui_main.ToolboxUI method), 633 empty_child() (spinetool-
 EditableMixin (class in spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_item.DBItem
 box.spine_db_editor.mvcmodels.tree_item_utility), method), 277
 309 empty_child() (spinetool-
 EditObjectClassesDialog (class in spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_item.ListItem
 box.spine_db_editor.widgets.edit_or_remove_items_dialogs) method), 277
 356 empty_child() (spinetool-
 EditObjectsDialog (class in spinetool- box.spine_db_editor.mvcmodels.scenario_item.ScenarioDBItem
 box.spine_db_editor.widgets.edit_or_remove_items_dialogs) method), 294
 356 empty_child() (spinetool-
 editorEvent() (spinetool- box.spine_db_editor.mvcmodels.scenario_item.ScenarioItem
 box.spine_db_editor.widgets.custom_delegates.RelationshipPivotTableDelegate
 method), 323 empty_child() (spinetool-
 editorEvent() (spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.FeatureRootItem
 box.spine_db_editor.widgets.custom_delegates.ScenarioAlternativeTableDelegate
 method), 324 empty_child() (spinetool-
 editorEvent() (spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureMe
 box.widgets.custom_delegates.CheckBoxDelegate method), 305
 method), 412 empty_child() (spinetool-
 EditOrRemoveItemsDialog (class in spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureRo
 box.spine_db_editor.widgets.edit_or_remove_items_dialogs) method), 304
 355 empty_child() (spinetool-
 EditRelationshipClassesDialog (class in spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.ToolRootItem
 box.spine_db_editor.widgets.edit_or_remove_items_dialogs) method), 303
 356 empty_child() (spinetool-
 EditRelationshipsDialog (class in spinetool- box.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildMixi
 box.spine_db_editor.widgets.edit_or_remove_items_dialogs) method), 309
 357 empty_child() (spinetool-
 ElidedCombobox (class in spinetool- box.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildRoot
 box.widgets.custom_combobox), 411 method), 310
 ElidedLabel (class in spinetool- empty_model (spinetool-
 box.widgets.custom_qwidgets), 441 box.mvcmodels.compound_table_model.CompoundWithEmptyTab
 ElidedTextMixin (class in spinetool- property), 180
 box.widgets.custom_qwidgets), 436 EmptyChildMixin (class in spinetool-
 emit_connection_failed() (spinetool- box.spine_db_editor.mvcmodels.tree_item_utility),
 box.widgets.custom_qgraphicsscene.DesignGraphicsScene 309
 method), 421 EmptyChildRootItem (class in spinetool-
 emit_filter_changed() (spinetool- box.spine_db_editor.mvcmodels.tree_item_utility),
 box.spine_db_editor.widgets.custom_menus.ParameterViewFilterMenu
 method), 334 emptyColumnCount() (spinetool-
 emit_filter_changed() (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableM
 box.spine_db_editor.widgets.custom_menus.TabularViewFilterMenu
 method), 335 method), 287
 EmptyObjectParameterDefinitionModel
 emit_filter_changed() (spinetool- (class in spinetool-
 box.widgets.custom_menus.FilterMenuBase box.spine_db_editor.mvcmodels.empty_parameter_models),
 method), 420 246
 emit_pinned_values_updated() (spinetool- EmptyObjectParameterValueModel
 box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor
 method), 382 (class in spinetool-
 empty (in module spinetoolbox.mvcmodels.map_model), 247 box.spine_db_editor.mvcmodels.empty_parameter_models),
 247

EmptyParameterDefinitionModel (class in spinetool-
box.spine_db_editor.mvcmodels.empty_parameter_models), method), 602
245 entity_class_id() (spinetool-
EmptyParameterModel (class in spinetool-
box.spine_db_editor.mvcmodels.empty_parameter_models), method), 395
244 entity_class_id_key (spinetool-
EmptyParameterValueModel (class in spinetool-
box.spine_db_editor.mvcmodels.compound_parameter_models.C
box.spine_db_editor.mvcmodels.empty_parameter_models), property), 239
246 entity_class_id_key (spinetool-
EmptyRelationshipParameterDefinitionModel
(class in spinetool-
box.spine_db_editor.mvcmodels.empty_parameter_models.Empty
property), 244
246 entity_class_id_key (spinetool-
EmptyRelationshipParameterValueModel
(class in spinetool-
box.spine_db_editor.mvcmodels.empty_parameter_models), box.spine_db_editor.graphics_items.CrossHairsItem
property), 298
247 entity_class_name (spinetool-
emptyRowCount() (spinetool-
box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModelBoxEditor.graphics_items.EntityItem
method), 287 property), 395
EmptyRowModel (class in spinetool-
box.mvcmodels.empty_row_model), 181
entity_class_name (spinetool-
enable_edit_actions() (spinetool-
box.ui_main.ToolboxUI method), 636 property), 298
enable_execute_all (spinetool-
box.project_settings.ProjectSettings attribute),
583 property), 298
entity_class_name_key (spinetool-
enabled_changed (spinetool-
box.widgets.custom_qwidgets._MenuToolBar
attribute), 439 property), 245
entity_class_renderer() (spinetool-
end_style_change() (spinetool-
box.spine_db_editor.widgets.spine_db_editor.SpineDBEditomethod), 601
method), 383 entity_class_type (spinetool-
endFormatScope() (spinetool-
box.widgets.persistent_console_widget.AnsiEscapeCodeHandlerproperty), 395
method), 480 entity_class_type (spinetool-
engine_data (spinetool-
box.spine_engine_worker.SpineEngineWorker
property), 625 property), 242
entity_class_type (spinetool-
engine_final_state() (spinetool-
box.spine_engine_worker.SpineEngineWorker
method), 626 property), 238
entity_class_type (spinetool-
engine_server_settings() (spinetool-
box.ui_main.ToolboxUI method), 635 property), 242
EntityClient (class in spinetool-
box.server.engine_client), 232
entity_class_type (spinetool-
ensure_window_is_on_screen() (in module spine-
toolbox.helpers), 525 property), 246
entity_class_type (spinetool-
enterEvent() (spinetool-
box.widgets.notification.Notification method),
470 property), 247
entity_class_type (spinetool-
ENTITY (spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_namespace_item_type
attribute), 258 property), 246
entity_class_icon() (spinetool-
entity_class_type (spinetool-

box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel (class in spine-
toolbox), 244

entity_class_type (spinetool- entity_type (spinetool-
box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel (class in spine-
toolbox), 246

entity_class_type (spinetool- entity_type (spinetool-
box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel (class in spine-
toolbox), 247

entity_class_type (spinetool- entity_type (spinetool-
box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel (class in spine-
toolbox), 243

entity_class_type (spinetool- entity_type (spinetool-
box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel (class in spine-
toolbox), 300

entity_class_type (spinetool- entity_type (spinetool-
box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel (class in spine-
toolbox), 298

entity_class_type (spinetool- entity_type (spinetool-
box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel (class in spine-
toolbox), 300

entity_id() (spinetool- entity_type (spinetool-
box.spine_db_editor.graphics_items.EntityItem (class in spine-toolbox), 395

entity_id_key (spinetool- entity_type (spinetool-
box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel (class in spine-
toolbox), 247

entity_id_key (spinetool- entity_type (spinetool-
box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel (class in spine-
toolbox), 300

entity_items (spinetool- entity_type (spinetool-
box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView (class in spine-
toolbox), 336

entity_name (spinetool- entity_type (spinetool-
box.spine_db_editor.graphics_items.CrossHairsItem (class in spine-toolbox), 399

entity_name (spinetool- EntityClassItem (class in spine-
box.spine_db_editor.graphics_items.EntityItem (class in spine-toolbox), 395

entity_name_edited (spinetool- EntityItem (class in spine-
box.spine_db_editor.graphics_items.ObjectLabelItem (class in spine-toolbox), 401

entity_name_key (spinetool- box.spine_db_editor.mvcmodels.entity_tree_item),
box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterValueModel (class in spine-
toolbox), 247

entity_name_key (spinetool- box.spine_db_editor.widgets.custom_qgraphicsviews),
box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterValueMixin (class in spine-
toolbox), 301

entity_name_key_in_cache (spinetool- box.spine_db_editor.mvcmodels.entity_tree_item),
box.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterValueModel (class in spine-
toolbox), 247

entity_name_key_in_cache (spinetool- box.spine_db_editor.widgets.custom_qtreeview),
box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterValueMixin (class in spine-
toolbox), 301

entity_type (spinetool- ERROR (spinetoolbox.headless.Status attribute), 516
box.spine_db_editor.graphics_items.EntityItem (class in spine-toolbox), 395

entity_type (spinetool- error_box (spinetoolbox.headless.HeadlessLogger attribute), 512
error_box (spinetoolbox.logger_interface.LoggerInterface (class in spine-toolbox), 521

attribute), 543

error_box (spinetoolbox.ui_main.ToolboxUI attribute), 627

error_msg (spinetoolbox.spine_db_manager.SpineDBManager attribute), 596

event() (spinetoolbox.headless.ActionsWithProject method), 514

event() (spinetoolbox.spine_db_editor.widgets.custom_menu_method), 334

event() (spinetoolbox.spine_db_editor.widgets.custom_menu_method), 335

event() (spinetoolbox.widgets.custom_qgraphicsscene.DesignedAndScreen method), 422

eventFilter() (spinetoolbox.helpers.ChildCyclingKeyPressFilter method), 526

eventFilter() (spinetoolbox.spine_db_editor.widgets.object_name_list_editor_method), 370

eventFilter() (spinetoolbox.ui_main.ToolboxUI method), 628

eventFilter() (spinetoolbox.widgets.custom_editors._CustomLineEditDelegate method), 414

eventFilter() (spinetoolbox.widgets.custom_qwidgets._MenuToolBar method), 440

eventFilter() (spinetoolbox.widgets.custom_qwidgets.ToolBarWidgetAction method), 438

eventFilter() (spinetoolbox.widgets.properties_widget.PropertiesWidgetBase method), 489

eventFilter() (spinetoolbox.widgets.settings_widget.SettingsWidget method), 494

exception() (spinetoolbox.qthread_pool_executor.QtBasedFuture method), 589

ExclamationIcon (class in spinetoolbox.project_item_icon), 582

executable_class (spinetoolbox.project_item.project_item.ProjectItem property), 220

execute_dags() (spinetoolbox.project.SpineToolboxProject method), 566

execute_project() (spinetoolbox.project.SpineToolboxProject method), 567

execute_selected() (spinetoolbox.project.SpineToolboxProject method), 566

execution_finished (spinetoolbox.execution_managers.ExecutionManager attribute), 505

execution_timestamps() (spinetoolbox.widgets.custom_qtextbrowser.CustomQTextBrowser method), 433

ExecutionIcon (class in spinetoolbox.project_item_icon), 581

ExecutionManager (class in spinetoolbox.execution_managers), 505

EXISTINGVIEWFILTERMENU (spinetoolbox.project.ItemNameStatus attribute), 557

expand_and_resize() (spinetoolbox.widgets.open_project_widget.OpenProjectDialog method), 471

EXPANSE_COLOR (in module spinetoolbox.mvcmodels.indexed_value_table_model), 187

export_data_delegate (spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 337

export_data() (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase method), 380

export_data() (spinetoolbox.spine_db_manager.SpineDBManager method), 610

export_selected() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349

export_selected() (spinetoolbox.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin method), 391

export_session() (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase method), 380

export_to_excel() (spinetoolbox.spine_db_manager.SpineDBManager method), 610

export_to_json() (spinetoolbox.spine_db_manager.SpineDBManager method), 610

export_to_sqlite() (spinetoolbox.spine_db_manager.SpineDBManager method), 610

ExtraColumn (class in spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model), 257

ExtraColumn (class in spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model), 260

F

failed (spinetoolbox.plugin_manager._PluginWorker attribute), 556

FAILURE (spinetoolbox.widgets.add_up_spine_opt_wizard._PageId attribute), 406

FAILURE (spinetoolbox.widgets.install_julia_wizard._PageId attribute), 450

FailurePage (class in spinetoolbox.widgets.add_up_spine_opt_wizard), 407

FailurePage (class in spinetoolbox.widgets.install_julia_wizard), 451

feature_id_list (spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem property), 304

FeatureLeafItem (class in spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item), 303

FeatureRootItem (class in spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item), 303

fetch_all() (spinetoolbox.spine_db_worker.SpineDBWorker method), 616

fetch_filters() (spinetoolbox.mvcmodels.resource_filter_model.ResourceFilterModel method), 209

fetch_item_type (spinetoolbox.fetch_parent.FetchParent property), 507

fetch_item_type (spinetoolbox.fetch_parent.ItemTypeFetchParent property), 509

fetch_item_type (spinetoolbox.spine_db_editor.mvcmodels.alternative_item.DBItem property), 235

fetch_item_type (spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem property), 267

fetch_item_type (spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item.DBItem property), 277

fetch_item_type (spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item.ListItem property), 277

fetch_item_type (spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioDBItem property), 294

fetch_item_type (spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioItem property), 294

fetch_item_type (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel property), 285

fetch_kernels() (spinetoolbox.ui_main.ToolboxUI method), 630

fetch_more() (spinetoolbox.mvcmodels.minimal_tree_model.TreeItem method), 197

fetch_more() (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.EntityItem method), 252

fetch_more() (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.ObjectClassItem method), 250

fetch_more() (spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem method), 268

fetch_more() (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.FetchMoreMixin method), 309

fetch_more() (spinetoolbox.spine_db_manager.SpineDBManager method), 597

fetch_more_if_possible() (spinetoolbox.spine_db_worker.SpineDBWorker method), 615

fetch_more_if_possible() (spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem method), 268

fetcher_waiting_over (spinetoolbox.spine_db_manager.SpineDBManager attribute), 597

fetchMore() (spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel method), 179

fetchMore() (spinetoolbox.mvcmodels.empty_row_model.EmptyRowModel method), 181

fetchMore() (spinetoolbox.mvcmodels.filter_checkbox_list_model.LazyFilterCheckboxList method), 185

fetchMore() (spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel method), 193

fetchMore() (spinetoolbox.mvcmodels.minimal_tree_model.MinimalTreeModel method), 199

fetchMore() (spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModels method), 239

fetchMore() (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase method), 263

fetchMore() (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 285

FetchMoreMixin (class in spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility), 309

FetchParent (class in spinetoolbox.fetch_parent), 507

- 570
- `find_cascading_parameter_data()` (*spinetoolbox.spine_db_manager.SpineDBManager* method), 609
- `find_cascading_parameter_values_by_definition()` (*spinetoolbox.spine_db_manager.SpineDBManager* method), 610
- `find_cascading_parameter_values_by_entity()` (*spinetoolbox.spine_db_manager.SpineDBManager* method), 610
- `find_cascading_relationship_classes()` (*spinetoolbox.spine_db_manager.SpineDBManager* method), 609
- `find_cascading_relationships()` (*spinetoolbox.spine_db_manager.SpineDBManager* method), 609
- `find_cascading_scenario_alternatives_by_scenario_id()` (*spinetoolbox.spine_db_manager.SpineDBManager* method), 610
- `find_category()` (*spinetoolbox.mvcmodels.project_item_model.ProjectItemModel* method), 201
- `find_child()` (*spinetoolbox.mvcmodels.minimal_tree_model.TreeItem* method), 196
- `find_children()` (*spinetoolbox.mvcmodels.minimal_tree_model.TreeItem* method), 196
- `find_children_by_id()` (*spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem* method), 269
- `find_connection()` (*spinetoolbox.headless.ModifiableProject* method), 513
- `find_connection()` (*spinetoolbox.project.SpineToolboxProject* method), 563
- `find_frozen_values()` (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* method), 390
- `find_groups_by_entity()` (*spinetoolbox.spine_db_manager.SpineDBManager* method), 610
- `find_index()` (*spinetoolbox.mvcmodels.filter_execution_model.FilterExecutionModel* method), 186
- `find_item()` (*spinetoolbox.headless.ModifiableProject* method), 513
- `find_item()` (*spinetoolbox.mvcmodels.project_item_model.ProjectItemModel* method), 201
- `find_items()` (*spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem* method), 270
- `find_jump()` (*spinetoolbox.project.SpineToolboxProject* method), 565
- `find_next_relationship()` (*spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView* method), 350
- `find_next_relationship_index()` (*spinetoolbox.spine_db_editor.mvcmodels.entity_tree_models.ObjectTreeModel* method), 254
- `find_row()` (*spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem* method), 269
- `find_rows_by_id()` (*spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem* method), 269
- `finished` (*spinetoolbox.plugin_manager.PluginWorker* attribute), 556
- `finished` (*spinetoolbox.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGenerator* attribute), 358
- `finished` (*spinetoolbox.spine_engine_worker.SpineEngineWorker* attribute), 625
- `first_db_map` (*spinetoolbox.spine_db_editor.graphics_items.EntityItem* property), 395
- `first_db_map` (*spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem* property), 266
- `first_db_map` (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase* property), 378
- `first_entity_class_id` (*spinetoolbox.spine_db_editor.graphics_items.EntityItem* property), 395
- `first_id` (*spinetoolbox.spine_db_editor.graphics_items.EntityItem* property), 395
- `first_non_null()` (in module *spinetoolbox.helpers*), 522
- `fix_geometry()` (*spinetoolbox.widgets.custom_editors.PivotHeaderTableLineEditor* method), 414
- `fix_lightness_color()` (in module *spinetoolbox.helpers*), 532
- `FIXED_FIELD_COLOR` (in module *spinetoolbox.spine_db_editor.mvcmodels.colors*), 237
- `fixed_fields` (*spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel* property), 298
- `flags()` (*spinetoolbox.mvcmodels.array_model.ArrayModel* method), 176
- `flags()` (*spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel* method), 179

- ul style="list-style-type: none; padding-left: 0;">
- flags() (spinetoolbox.mvcmodels.empty_row_model.EmptyRowModel method), 181
- flags() (spinetoolbox.mvcmodels.empty_row_model.EmptyRowModel method), 181
- flags() (spinetoolbox.mvcmodels.file_list_models.FileListModel method), 182
- flags() (spinetoolbox.mvcmodels.map_model.MapModel method), 189
- flags() (spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel method), 193
- flags() (spinetoolbox.mvcmodels.minimal_tree_model.MinimalTreeModel method), 199
- flags() (spinetoolbox.mvcmodels.minimal_tree_model.TreeItem method), 197
- flags() (spinetoolbox.mvcmodels.project_item_model.ProjectItemModel method), 200
- flags() (spinetoolbox.mvcmodels.project_item_specification_model.ProjectItemSpecificationModel method), 204
- flags() (spinetoolbox.mvcmodels.project_tree_item.BaseProjectTreeItem method), 205
- flags() (spinetoolbox.mvcmodels.project_tree_item.CategoryProjectTreeItem method), 207
- flags() (spinetoolbox.mvcmodels.project_tree_item.LeafProjectTreeItem method), 208
- flags() (spinetoolbox.mvcmodels.time_pattern_model.TimePatternModel method), 210
- flags() (spinetoolbox.mvcmodels.time_series_model_fixed_resolution_model.FixedResolutionTimeSeriesModel method), 212
- flags() (spinetoolbox.mvcmodels.time_series_model_variable_resolution_model.VariableResolutionTimeSeriesModel method), 214
- flags() (spinetoolbox.spine_db_editor.mvcmodels.alternative_model.AlternativeItem method), 236
- flags() (spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_model.EmptyRelationshipParameterDefinitionModel method), 246
- flags() (spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel method), 259
- flags() (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel method), 261
- flags() (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_model.PivotTableModel method), 291
- flags() (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_model.PivotTableModelBase method), 287
- flags() (spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioItem method), 295
- flags() (spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioItem method), 294
- flags() (spinetoolbox.spine_db_editor.mvcmodels.single_parameter_model.SingleParameterModel method), 299
- flags() (spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 303
- flags() (spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 305
- flags() (spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 306
- flags() (spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem method), 305

box.spine_db_icon_manager.SceneSvgRenderer *get_all_feature_names()* (*spinetool-*
class method), 595 *box.spine_db_editor.mvcmodels.tool_feature_model.ToolFeatureModel*
method), 307
frozen_values() (*spinetool-* *get_all_multi_spine_db_editors()* (*spinetool-*
method), 280 *box.spine_db_manager.SpineDBManager*
static method), 610
frozen_values_added (*spinetool-* *get_all_multi_db_spec_editors()* (*spinetool-*
box.spine_db_editor.mvcmodels.pivot_table_model.PivotTableModel *static method*), 635
attribute), 285 *box.ui_main.ToolboxUI*
frozen_values_removed (*spinetool-* *get_all_regular_kernels()* (*spinetool-*
box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel *method*),
attribute), 285 *box.model_fetcher.KernelFetcher*
method), 534
FrozenTableModel (*class in spinetool-* *get_all_spine_db_editors()* (*spinetool-*
box.spine_db_editor.mvcmodels.frozen_table_model), *box.spine_db_manager.SpineDBManager*
method), 610
FrozenTableView (*class in spinetool-* *get_auto_filter_menu()* (*spinetool-*
box.spine_db_editor.widgets.custom_qtableview), *box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel*
method), 239
full_push_object_class_ids() (*spinetool-* *get_checkbox_rect()* (*spinetool-*
box.spine_db_parcel.SpineDBParcel *method*), *box.widgets.custom_delegates.CheckBoxDelegate*
method), 413
full_push_object_ids() (*spinetool-* *get_checked_states()* (*spinetool-*
box.spine_db_parcel.SpineDBParcel *method*), *box.widgets.custom_qwidgets.SelectDatabaseItemsDialog*
method), 442
full_push_relationship_class_ids() (*spinetool-* *get_console()* (*spinetool-*
box.spine_db_parcel.SpineDBParcel *method*), *box.mvcmodels.filter_execution_model.FilterExecutionModel*
method), 186
full_push_relationship_ids() (*spinetool-* *get_datetime()* (*in module spinetoolbox.helpers*), 521
box.spine_db_parcel.SpineDBParcel *method*), *get_db_map()* (*spinetool-*
box.spine_db_manager.SpineDBManager
method), 599
full_push_scenario_ids() (*spinetool-* *get_db_map()* (*spinetool-*
box.spine_db_parcel.SpineDBParcel *method*), *box.spine_db_worker.SpineDBWorker* *method*),
method), 614
fully_collapse() (*spinetool-* *get_db_map_cache()* (*spinetool-*
box.spine_db_editor.widgets.custom_qtreeview.EngineTreeView *method*), 349 *box.spine_db_manager.SpineDBManager*
method), 598
fully_expand() (*spinetool-* *get_elapsed_time()* (*spinetool-*
box.spine_db_editor.widgets.custom_qtreeview.EngineTreeView *method*), 349 *box.server.engine_client.EngineClient* *method*),
method), 234
G *get_engine_data()* (*spinetool-*
box.spine_engine_worker.SpineEngineWorker
method), 625
gentle_zoom() (*spinetool-* *get_engine_event()* (*spinetool-*
box.widgets.custom_qgraphicsviews.CustomQGraphicsView *method*), 424 *box.spine_engine_manager.LocalSpineEngineManager*
method), 620
get() (*spinetoolbox.qthread_pool_executor.QtBasedQueue* *box.spine_engine_manager.RemoteSpineEngineManager*
method), 589 *method*), 622
get_action() (*spinetool-* *get_engine_event()* (*spinetool-*
box.widgets.custom_menus.CustomContextMenu *box.spine_engine_manager.SpineEngineManagerBase*
method), 418 *method*), 618
get_all_conda_kernels() (*spinetool-* *get_engine_event()* (*spinetool-*
box.kernel_fetcher.KernelFetcher *method*), *box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel*
method), 534 *method*), 242
get_all_feature_methods() (*spinetool-* *get_entity_class_id()* (*spinetool-*
box.spine_db_editor.mvcmodels.tool_feature_model.ToolFeatureModel *box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel*
method), 307 *method*), 242

<code>get_feature_data()</code> <i>box.spine_db_editor.mvcmodels.tool_feature_model.ToolFeatureModel</i> <i>method</i>), 307	<i>(spinetool-</i>	<code>get_mime_data_text()</code> <i>box.spine_db_editor.mvcmodels.project_item_specification_models.FilteredSpecificationModel</i> <i>method</i>), 205	<i>(spinetool-</i>
<code>get_field()</code> <i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 602	<i>(spinetool-</i>	<code>get_next_urls()</code> <i>box.spine_db_editor.widgets.url_toolbar.UrlToolBar</i> <i>method</i>), 392	<i>(spinetool-</i>
<code>get_field_item()</code> <i>box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel</i> <i>method</i>), 299	<i>(spinetool-</i>	<code>get_not_selected()</code> <i>box.spine_db_editor.widgets.checkbox_list_model.SimpleFilterCheckboxListModel</i> <i>method</i>), 185	<i>(spinetool-</i>
<code>get_field_item_data()</code> <i>box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel</i> <i>method</i>), 299	<i>(spinetool-</i>	<code>get_opacity()</code> <i>box.spine_db_editor.widgets.notification_notification_model.NotificationModel</i> <i>method</i>), 469	<i>(spinetool-</i>
<code>get_frozen_value()</code> <i>box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel</i> <i>method</i>), 256	<i>(spinetool-</i>	<code>get_open_file_name_in_last_dir()</code> (in module <i>box.helpers</i>), 525	<i>(in module</i>
<code>get_icon()</code> (<i>spinetoolbox.kernel_fetcher.KernelFetcher</i> <i>static method</i>), 534		<code>get_parameter_value_list()</code> <i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 604	<i>(spinetool-</i>
<code>get_icon()</code> <i>box.project_item.project_item.ProjectItem</i> <i>method</i>), 221	<i>(spinetool-</i>	<code>get_pdf_file_path()</code> <i>box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin</i> <i>method</i>), 362	<i>(spinetool-</i>
<code>get_icon_mgr()</code> <i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 598	<i>(spinetool-</i>	<code>get_persistent_completions()</code> <i>box.spine_engine_manager.LocalSpineEngineManager</i> <i>method</i>), 621	<i>(spinetool-</i>
<code>get_id_key()</code> <i>box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel</i> <i>method</i>), 299	<i>(spinetool-</i>	<code>get_persistent_completions()</code> <i>box.spine_engine_manager.RemoteSpineEngineManager</i> <i>method</i>), 623	<i>(spinetool-</i>
<code>get_item()</code> <i>box.mvcmodels.project_item_model.ProjectItemModel</i> <i>method</i>), 201	<i>(spinetool-</i>	<code>get_persistent_completions()</code> <i>box.spine_engine_manager.SpineEngineManagerBase</i> <i>method</i>), 619	<i>(spinetool-</i>
<code>get_item()</code> (<i>spinetoolbox.project.SpineToolboxProject</i> <i>method</i>), 562		<code>get_persistent_history_item()</code> <i>box.spine_engine_manager.LocalSpineEngineManager</i> <i>method</i>), 621	<i>(spinetool-</i>
<code>get_item()</code> <i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 602	<i>(spinetool-</i>	<code>get_persistent_history_item()</code> <i>box.spine_engine_manager.RemoteSpineEngineManager</i> <i>method</i>), 623	<i>(spinetool-</i>
<code>get_item_by_field()</code> <i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 603	<i>(spinetool-</i>	<code>get_persistent_history_item()</code> <i>box.spine_engine_manager.SpineEngineManagerBase</i> <i>method</i>), 620	<i>(spinetool-</i>
<code>get_items()</code> (<i>spinetoolbox.project.SpineToolboxProject</i> <i>method</i>), 563		<code>get_pivot_preferences()</code> <i>box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin</i> <i>method</i>), 388	<i>(spinetool-</i>
<code>get_items()</code> <i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 602	<i>(spinetool-</i>	<code>get_pivoted_data()</code> <i>box.spine_db_editor.mvcmodels.pivot_model.PivotModel</i> <i>method</i>), 281	<i>(spinetool-</i>
<code>get_items_by_field()</code> <i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 602	<i>(spinetool-</i>	<code>get_previous_urls()</code> <i>box.spine_db_editor.widgets.url_toolbar.UrlToolBar</i> <i>method</i>), 392	<i>(spinetool-</i>
<code>get_items_for_commit()</code> <i>box.spine_db_manager.SpineDBManager</i> <i>method</i>), 610	<i>(spinetool-</i>	<code>get_project_directory()</code> <i>box.project_upgrader.ProjectUpgrader</i> <i>method</i>), 587	<i>(spinetool-</i>
<code>get_kernel_deats()</code> <i>box.kernel_fetcher.KernelFetcher</i> <i>static method</i>), 534	<i>(spinetool-</i>	<code>get_save_file_name_in_last_dir()</code> (in module <i>spinetoolbox.helpers</i>), 525	<i>(in module</i>
<code>get_method_index()</code> <i>box.spine_db_editor.mvcmodels.tool_feature_model.ToolFeatureModel</i> <i>method</i>), 307	<i>(spinetool-</i>	<code>get_to_scenario_model_alternative_id_list()</code> <i>spinetoolbox.spine_db_manager.SpineDBManager</i> <i>method</i>), 610	<i>(spine-</i>

method), 604

get_scenario_names() (spinetool-
box.project_item.logging_connection.LoggingConnection
method), 217

get_selected() (spinetool-
box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel
method), 185

get_set_data_delayed() (spinetool-
box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel
method), 242

get_set_data_delayed() (spinetool-
box.spine_db_editor.mvcmodels.parameter_value_list_model.ParameterValueListModel
method), 279

get_set_data_delayed() (spinetool-
box.spine_db_editor.mvcmodels.pivot_table_models.ParameterPivotTableModel
method), 291

get_specification() (spinetool-
box.project.SpineToolboxProject
method), 561

get_tool_names() (spinetool-
box.project_item.logging_connection.LoggingConnection
method), 217

get_upgrade_db_prompt_text() (in module spinetool-
box.helpers), 528

get_value() (spinetool-
box.spine_db_manager.SpineDBManager
method), 603

get_value_from_data() (spinetool-
box.spine_db_manager.SpineDBManager
method), 603

get_value_index() (spinetool-
box.spine_db_manager.SpineDBManager
method), 604

get_value_indexes() (spinetool-
box.spine_db_manager.SpineDBManager
method), 604

get_value_list_item() (spinetool-
box.spine_db_manager.SpineDBManager
method), 604

GetObjectClassesMixin (class in spinetool-
box.spine_db_editor.widgets.manage_items_dialogs),
364

GetObjectsMixin (class in spinetool-
box.spine_db_editor.widgets.manage_items_dialogs),
364

GetRelationshipClassesMixin (class in spinetool-
box.spine_db_editor.widgets.manage_items_dialogs),
365

go_desktop() (spinetool-
box.widgets.open_project_widget.OpenProjectDialog
method), 472

go_documents() (spinetool-
box.widgets.open_project_widget.OpenProjectDialog
method), 472

go_home() (spinetoolbox.widgets.open_project_widget.OpenProjectDialog
method), 472

go_project() (spinetoolbox.widgets.open_project_widget.OpenProjectDialog
method), 472

graph_selection_changed (spinetool-
box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView
attribute), 336

graphics_item (spinetool-
box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView
property), 217

graphics_item (spinetool-
box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView
property), 219

GraphLayoutGeneratorRunnable (class in spinetool-
box.spine_db_editor.widgets.graph_layout_generator),
358

GraphLayoutGeneratorRunnable.Signals
(class in spinetool-
box.spine_db_editor.widgets.graph_layout_generator),
358

GraphViewMixin (class in spinetool-
box.spine_db_editor.widgets.graph_view_mixin),
359

GrayIfLastMixin (class in spinetool-
box.spine_db_editor.mvcmodels.tree_item_utility),
309

group_fields (spinetool-
box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel
property), 298

group_renderer() (spinetool-
box.spine_db_icon_manager.SpineDBIconManager
method), 595

guide_path() (spinetoolbox.link.LinkBase method),
536

H

H_MARGIN (spinetoolbox.widgets.custom_qwidgets.TitleWidgetAction
attribute), 440

HalfSortedTableModel (class in spinetool-
box.spine_db_editor.mvcmodels.single_parameter_models),
297

handle_close_request_from_tab() (spinetool-
box.widgets.multi_tab_window.MultiTabWindow
method), 467

handle_data() (spinetoolbox.helpers.HTMLTagFilter
method), 533

handle_execution_successful() (spinetool-
box.project_item.project_item.ProjectItem
method), 222

handle_header_dropped() (spinetool-
box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin
method), 389

handle_julia_install_finished() (spinetool-
box.widgets.kernel_editor.KernelEditorBase
method), 472

- method*), 457
- `handle_ijulia_rebuild_finished()` (*spinetoolbox.widgets.kernel_editor.KernelEditorBase* *method*), 458
- `handle_installkernel_process_finished()` (*spinetoolbox.widgets.kernel_editor.KernelEditorBase* *method*), 458
- `handle_installkernel_process_finished()` (*spinetoolbox.widgets.kernel_editor.MinijuliaKernelEditor* *method*), 459
- `handle_items_added()` (*spinetoolbox.fetch_parent.FetchParent* *method*), 508
- `handle_items_added()` (*spinetoolbox.fetch_parent.FlexibleFetchParent* *method*), 510
- `handle_items_added()` (*spinetoolbox.fetch_parent.ItemTypeFetchParent* *method*), 509
- `handle_items_added()` (*spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel* *method*), 241
- `handle_items_added()` (*spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel* *method*), 245
- `handle_items_added()` (*spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem* *method*), 268
- `handle_items_added()` (*spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioItem* *method*), 294
- `handle_items_added()` (*spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.TreeItemUtility* *method*), 309
- `handle_items_removed()` (*spinetoolbox.fetch_parent.FetchParent* *method*), 509
- `handle_items_removed()` (*spinetoolbox.fetch_parent.FlexibleFetchParent* *method*), 510
- `handle_items_removed()` (*spinetoolbox.fetch_parent.ItemTypeFetchParent* *method*), 509
- `handle_items_removed()` (*spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel* *method*), 241
- `handle_items_removed()` (*spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem* *method*), 268
- `handle_items_removed()` (*spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioItem* *method*), 295
- `handle_items_removed()` (*spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.TreeItemUtility* *method*), 309
- `handle_items_updated()` (*spinetoolbox.fetch_parent.FetchParent* *method*), 509
- `handle_items_updated()` (*spinetoolbox.fetch_parent.FlexibleFetchParent* *method*), 510
- `handle_items_updated()` (*spinetoolbox.fetch_parent.ItemTypeFetchParent* *method*), 510
- `handle_items_updated()` (*spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel* *method*), 241
- `handle_items_updated()` (*spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem* *method*), 268
- `handle_items_updated()` (*spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioItem* *method*), 295
- `handle_items_updated()` (*spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.TreeItemUtility* *method*), 310
- `handle_items_updated()` (*spinetoolbox.widgets.kernel_editor.KernelEditorBase* *method*), 457
- `handle_items_updated()` (*spinetoolbox.widgets.kernel_editor.MinijuliaKernelEditor* *method*), 459
- `handle_items_updated()` (*spinetoolbox.widgets.add_project_item_widget.AddProjectItemWidget* *method*), 404
- `handle_items_updated()` (*spinetoolbox.widgets.add_project_item_widget.AddProjectItemWidget* *method*), 404
- `handle_install_package_install_process_finished()` (*spinetoolbox.widgets.kernel_editor.KernelEditorBase* *method*), 456
- `handle_redo_complete()` (*spinetoolbox.spine_db_commands.SpineDBCommand* *method*), 592
- `handle_scene_selection_changed()` (*spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView* *method*), 336
- `handle_selection_changed()` (*spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene* *method*), 422
- `handle_starttag()` (*spinetoolbox.helpers.HTMLTagFilter* *method*), 533
- `handle_undo_complete()` (*spinetoolbox.spine_db_commands.SpineDBCommand* *method*), 592
- `handle_updated_in_db()` (*spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioItem* *method*), 294
- `handle_updated_in_db()` (*spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.TreeItemUtility* *method*), 311
- `handle_updated_in_db()` (*spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem* *method*), 311

has_children()	(spinetool- box.mvcmodels.minimal_tree_model.TreeItem method), 196	box.spine_db_editor.mvcmodels.tree_item_utility.LeafItem method), 310
has_children()	(spinetool- box.spine_db_editor.mvcmodels.entity_tree_item.MemberObjectAttribute), 346	header_dropped (spinetool- box.spine_db_editor.widgets.custom_qtableview.FrozenTableView method), 253
has_children()	(spinetool- box.spine_db_editor.mvcmodels.entity_tree_item.RelationshipAttribute), 373	header_dropped (spinetool- box.spine_db_editor.widgets.pivot_table_header_view.PivotTable method), 254
has_children()	(spinetool- box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureRequiredItem method), 305	header_dropped (spinetool- box.spine_db_editor.widgets.tabular_view_header_widget.Tabular method), 305
has_filter()	(spinetool- box.widgets.custom_qwidgets.FilterWidget method), 437	header_name() (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableM method), 288
has_filters()	(spinetool- box.project_item.logging_connection.LoggingConnection method), 217	header_type (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftAlter property), 283
has_items()	(spinetoolbox.project.SpineToolboxProject method), 562	header_type (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftData property), 284
has_recents()	(spinetool- box.widgets.custom_menus.RecentProjectsPopupMenu method), 419	header_type (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftObje property), 282
has_twinned_axes()	(spinetool- box.widgets.plot_canvas.PlotCanvas method), 482	header_type (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftParam property), 283
hasChildren()	(spinetool- box.mvcmodels.minimal_tree_model.MinimalTreeModel method), 199	header_type (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftParam property), 283
header_changed	(spinetool- box.spine_db_editor.widgets.custom_qtableview.PivotTable attribute), 346	header_type (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftScene property), 284
header_data()	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftAttributeHeaderItem method), 284	headerColumnCount() (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableM method), 287
header_data()	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftDatabaseHeaderItem method), 284	headerData() (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftDatabaseHeaderItem method), 176
header_data()	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftFileHeaderItem method), 282	headerData() (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftFileHeaderItem method), 182
header_data()	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftObjectHeaderItem method), 282	headerData() (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftObjectHeaderItem method), 186
header_data()	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftParameterHeaderItem method), 283	headerData() (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftParameterHeaderItem method), 187
header_data()	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftTableHeaderItem method), 283	headerData() (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftTableHeaderItem method), 189
header_data()	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftValueHeaderItem method), 284	headerData() (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftValueHeaderItem method), 194
header_data()	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftValueHeaderItem method), 284	headerData() (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.TopLeftValueHeaderItem method), 194

<code>box.spine_db_editor.mvcmodels.compound_parameter_model_base.CompoundParameterModelBase</code>	(<code>spinetool- box.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase</code> <code>method</code>), 239	<code>box.project_item_icon.ProjectItemIcon</code> <code>method</code>), 582
<code>headerData()</code>	(<code>spinetool- box.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase</code> <code>method</code>), 264	<code>hoverEnterEvent()</code> (<code>spinetool- box.project_item_icon.ProjectItemIcon</code> <code>method</code>), 579
<code>headerData()</code>	(<code>spinetool- box.spine_db_editor.mvcmodels.multi_db_tree_model.MultiDBTreeModelBase</code> <code>method</code>), 270	<code>hoverLeaveEvent()</code> (<code>spinetoolbox.link._IconButton</code> <code>method</code>), 582
<code>headerData()</code>	(<code>spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableMultiBase</code> <code>method</code>), 288	<code>hoverLeaveEvent()</code> (<code>spinetool- box.project_item_icon.ConnectorButton</code> <code>method</code>), 582
<code>headerData()</code>	(<code>spinetool- box.spine_db_editor.mvcmodels.tree_model_base.TreeModelBase</code> <code>method</code>), 312	<code>hoverLeaveEvent()</code> (<code>spinetool- box.project_item_icon.ExclamationIcon</code> <code>method</code>), 582
<code>headerRowCount()</code>	(<code>spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModelBase</code> <code>method</code>), 287	<code>hoverLeaveEvent()</code> (<code>spinetool- box.project_item_icon.ExecutionIcon</code> <code>method</code>), 582
<code>headers</code> (<code>spinetoolbox.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel</code> <code>property</code>), 255	<code>box.project_item_icon.ProjectItemIcon</code> <code>method</code>), 579	
<code>headless_main()</code> (in module <code>spinetoolbox.headless</code>), 515	<code>hoverMoveEvent()</code> (<code>spinetool- box.project_item_icon.ProjectItemIcon</code> <code>method</code>), 579	
<code>HeadlessConnection</code> (class in <code>spinetool- box.project_item.logging_connection</code>), 215	<code>HTMLTagFilter</code> (class in <code>spinetoolbox.helpers</code>), 533	
<code>HeadlessLogger</code> (class in <code>spinetoolbox.headless</code>), 512	<code>HyperTextLabel</code> (class in <code>spinetool- box.widgets.custom_qwidgets</code>), 440	
<code>hide_removed_entities()</code> (<code>spinetool- box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin</code> <code>method</code>), 360		
<code>hide_selected_items()</code> (<code>spinetool- box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView</code> <code>method</code>), 336	<code>icon()</code> (<code>spinetoolbox.helpers.ProjectDirectoryIconProvider</code> <code>method</code>), 524	
<code>hideEvent()</code> (<code>spinetool- box.widgets.custom_qwidgets._MenuToolBar</code> <code>method</code>), 440	<code>icon()</code> (<code>spinetoolbox.project_item.project_item_factory.ProjectItemFactory</code> <code>static method</code>), 226	
<code>highlightBlock()</code> (<code>spinetool- box.helpers.CustomSyntaxHighlighter</code> <code>method</code>), 531	<code>icon_code</code> (<code>spinetoolbox.spine_db_editor.mvcmodels.alternative_item.AlternativeItem</code> <code>property</code>), 236	
<code>home_dir()</code> (in module <code>spinetoolbox.helpers</code>), 520	<code>icon_code</code> (<code>spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioItem</code> <code>property</code>), 294	
<code>horizontal_header_labels()</code> (<code>spinetool- box.mvcmodels.minimal_table_model.MinimalTableModel</code> <code>method</code>), 194	<code>icon_code</code> (<code>spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem</code> <code>property</code>), 303	
<code>HorizontalSpinBox</code> (class in <code>spinetool- box.widgets.custom_qwidgets</code>), 441	<code>icon_code</code> (<code>spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem</code> <code>property</code>), 305	
<code>hover_brush</code> (<code>spinetool- box.project_item_icon.ConnectorButton</code> <code>attribute</code>), 580	<code>icon_code</code> (<code>spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem</code> <code>property</code>), 304	
<code>hoverEnterEvent()</code> (<code>spinetoolbox.link._IconButton</code> <code>method</code>), 537	<code>icon_code</code> (<code>spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem</code> <code>property</code>), 303	
<code>hoverEnterEvent()</code> (<code>spinetool- box.project_item_icon.ConnectorButton</code> <code>method</code>), 581	<code>icon_code</code> (<code>spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.StarItemUtility</code> <code>property</code>), 308	
<code>hoverEnterEvent()</code> (<code>spinetool- box.project_item_icon.ExclamationIcon</code> <code>method</code>), 582	<code>icon_color()</code> (<code>spinetool- box.project_item.project_item_factory.ProjectItemFactory</code> <code>static method</code>), 226	
<code>hoverEnterEvent()</code> (<code>spinetool-</code>	<code>icon_color_editor_requested</code> (<code>spinetool- box.spine_db_editor.widgets.custom_delegates.ManageObjectClassDialog</code> <code>attribute</code>), 332	
	<code>icon_color_editor_requested</code> (<code>spinetool- box.spine_db_editor.widgets.custom_delegates.ManageRelationshipDialog</code> <code>attribute</code>), 332	

<code>icon_from_renderer()</code>	(spinetool- box.spine_db_icon_manager.SpineDBIconManager static method), 595	<code>increase_arc_length()</code>	(spinetool- box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 579
<code>icon_ordering()</code>	(spinetool- box.widgets.toolbars.MainToolBar method), 503	<code>increment_position()</code>	(spinetool- box.fetch_parent.FetchParent method), 507
<code>icon_renderer()</code>	(spinetool- box.spine_db_icon_manager.SpineDBIconManager method), 595	<code>index()</code>	(spinetoolbox.mvcmodels.file_list_models.FileListModel method), 183
<code>IColorEditor</code>	(class in spinetool- box.widgets.custom_editors), 416	<code>index()</code>	(spinetoolbox.mvcmodels.minimal_tree_model.MinimalTreeModel method), 198
<code>IconListManager</code>	(class in spinetoolbox.helpers), 523	<code>index()</code>	(spinetoolbox.mvcmodels.minimal_tree_model.TreeItem method), 197
<code>id</code>	(spinetoolbox.plotting.IndexName attribute), 547	<code>index()</code>	(spinetoolbox.mvcmodels.project_item_model.ProjectItemModel method), 200
<code>ID</code>	(spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.EntityColumn attribute), 260	<code>index_from_item()</code>	(spinetool- box.mvcmodels.minimal_tree_model.MinimalTreeModel method), 198
<code>id</code>	(spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem property), 310	<code>index_in_column_headers()</code>	(spinetool- box.spine_db_editor.widgets.tabular_view_header_widget.TabularHeaderViewHeaderWidget method), 287
<code>identifier</code>	(spinetool- box.spine_db_editor.widgets.tabular_view_header_widget.TabularHeaderViewHeaderWidget property), 383	<code>index_in_data()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 288
<code>import_data()</code>	(spinetool- box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 379	<code>index_in_empty_column_headers()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 287
<code>import_data()</code>	(spinetool- box.spine_db_manager.SpineDBManager method), 604	<code>index_in_empty_row_headers()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 287
<code>import_data_commands()</code>	(spinetool- box.spine_db_manager.SpineDBManager method), 605	<code>index_in_headers()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 287
<code>import_file()</code>	(spinetool- box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 379	<code>index_in_left()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 287
<code>import_from_excel()</code>	(spinetool- box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 380	<code>index_in_row_headers()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 287
<code>import_from_json()</code>	(spinetool- box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 379	<code>index_in_top()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 287
<code>import_from_sqlite()</code>	(spinetool- box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 380	<code>index_in_top_left()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 287
<code>import_specification()</code>	(spinetool- box.ui_main.ToolboxUI method), 632	<code>index_name()</code>	(spinetool- box.mvcmodels.map_model.MapModel method), 192
<code>ImposeEntityClassIdMixin</code>	(class in spinetool- box.spine_db_editor.mvcmodels.parameter_mixins), 275	<code>index_name()</code>	(spinetool- box.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel method), 241
<code>incoming_connections()</code>	(spinetool- box.project.SpineToolboxProject method), 569	<code>index_name()</code>	(spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_model.ParameterValueListModel method), 279
<code>incoming_links()</code>	(spinetool- box.project_item_icon.ConnectorButton method), 580	<code>index_name()</code>	(spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_model.ParameterValueListModel method), 279
<code>incoming_links()</code>	(spinetool- box.project_item_icon.ProjectItemIcon method), 580	<code>index_name()</code>	(spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_model.ParameterValueListModel method), 279

`box.spine_db_editor.mvcmodels.pivot_table_models.ParameterValueTableModel` (class in `box.spine_db_editor.mvcmodels.pivot_table_models`), 290

`box.spine_db_editor.widgets.compound_parameter_models.CompoundParameterTableModel` (class in `box.spine_db_editor.widgets.compound_parameter_models`), 242

`index_names` (`spinetoolbox.plotting.XYData` attribute), 547

`index_under_mouse()` (`spinetoolbox.widgets.multi_tab_window.TabBarPlus` method), 468

`index_within_top_left()` (`spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel` method), 287

`IndexedParameterValueTableViewBase` (class in `spinetoolbox.widgets.custom_qtableview`), 429

`IndexedValueTableContextMenu` (class in `spinetoolbox.widgets.indexed_value_table_context_menu`), 447

`IndexedValueTableModel` (class in `spinetoolbox.mvcmodels.indexed_value_table_model`), 187

`IndexedValueTableView` (class in `spinetoolbox.widgets.custom_qtableview`), 430

`indexes` (`spinetoolbox.mvcmodels.time_series_model_fixed_resolution_model.FixedResolutionSeriesModelFixedResolution` property), 212

`indexes` (`spinetoolbox.mvcmodels.time_series_model_variable_resolution_model.VariableResolutionSeriesModelVariableResolution` property), 213

`IndexExpansionPivotTableModel` (class in `spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models`), 291

`IndexName` (class in `spinetoolbox.plotting`), 547

`InferEntityClassIdMixin` (class in `spinetoolbox.spine_db_editor.mvcmodels.parameter_mixins`), 275

`information_box` (`spinetoolbox.headless.HeadlessLogger` attribute), 512

`information_box` (`spinetoolbox.logger_interface.LoggerInterface` attribute), 543

`information_box` (`spinetoolbox.ui_main.ToolboxUI` attribute), 627

`init_add_undo_redo_actions()` (`spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase` method), 379

`init_copy_and_paste_actions()` (`spinetoolbox.widgets.custom_qtableview.CopyPasteTableView` method), 428

`init_model()` (`spinetoolbox.helpers.IconListManager` method), 523

`init_model()` (`spinetoolbox.mvcmodels.compound_table_model.CompoundWithEmptyTableModel` method), 180

`init_model()` (`spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterTableModel` method), 239

`init_model()` (`spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel` method), 287

`init_model()` (`spinetoolbox.widgets.compound_parameter_models.CompoundParameterTableModel` method), 242

`init_model()` (`spinetoolbox.widgets.object_name_list_editor.ObjectNameListEditor` method), 371

`init_models()` (`spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin` method), 359

`init_models()` (`spinetoolbox.spine_db_editor.widgets.item_metadata_editor.ItemMetadataEditor` method), 363

`init_models()` (`spinetoolbox.spine_db_editor.widgets.metadata_editor.MetadataEditor` method), 367

`init_models()` (`spinetoolbox.spine_db_editor.widgets.parameter_view_mixin.ParameterViewMixin` method), 371

`init_models()` (`spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase` method), 382

`init_models()` (`spinetoolbox.widgets.spine_db_editor.SpineDBEditorBase` method), 382

`init_models()` (`spinetoolbox.widgets.tabular_view_mixin.TabularViewMixin` method), 385

`init_models()` (`spinetoolbox.widgets.tree_view_mixin.TreeViewMixin` method), 390

`init_project()` (`spinetoolbox.ui_main.ToolboxUI` method), 629

`init_project_item_model()` (`spinetoolbox.ui_main.ToolboxUI` method), 630

`init_specification_model()` (`spinetoolbox.ui_main.ToolboxUI` method), 630

`initial_entity_id()` (`spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectGroupDialog` method), 319

`initial_entity_id()` (`spinetoolbox.spine_db_editor.widgets.add_items_dialogs.ManageMembersDialog` method), 320

`initial_entity_id()` (`spinetoolbox.spine_db_editor.widgets.add_items_dialogs.ObjectGroupDialog` method), 319

`initial_member_ids()` (`spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddObjectGroupDialog` method), 319

`initial_member_ids()` (`spinetoolbox.spine_db_editor.widgets.add_items_dialogs.ManageMembersDialog` method), 320

`initial_member_ids()` (`spinetoolbox.spine_db_editor.widgets.add_items_dialogs.ObjectGroupDialog` method), 319

`initializePage()` (`spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterTableModel` method), 239

`box.widgets.add_up_spine_opt_wizard.AddUpSpineOptPage`method), 317

`method`), 407

`insert_column_data()` (spinetool-
`box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTable`
`initializePage()` (spinetool-
`box.widgets.add_up_spine_opt_wizard.CheckPreviousInstallPage`method), 256
`method`), 406

`insert_file_open_button()` (spinetool-
`box.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDB`
`initializePage()` (spinetool-
`box.widgets.add_up_spine_opt_wizard.FailurePage`method), 369
`method`), 407

`insert_horizontal_header_labels()` (spinetool-
`box.mvcmodels.minimal_table_model.MinimalTableModel`
`initializePage()` (spinetool-
`box.widgets.add_up_spine_opt_wizard.ResetRegistryPage`method), 194
`method`), 407

`insert_item()` (spinetool-
`box.mvcmodels.project_item_model.ProjectItemModel`
`initializePage()` (spinetool-
`box.widgets.add_up_spine_opt_wizard.SelectJuliaPage`method), 202
`method`), 406

`insert_new_tab()` (spinetool-
`box.widgets.multi_tab_window.MultiTabWindow`
`initializePage()` (spinetool-
`box.widgets.add_up_spine_opt_wizard.SuccessPage`method), 465
`method`), 407

`insert_sqlite_file_open_button()` (spinetool-
`box.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDB`
`initializePage()` (spinetool-
`box.widgets.add_up_spine_opt_wizard.TroubleshootSolutionPage`method), 369
`method`), 407

`insert_text_to_console()` (spinetool-
`box.widgets.jupyter_console_widget.JupyterConsoleWidget`
`initializePage()` (spinetool-
`box.widgets.install_julia_wizard.FailurePage`method), 454
`method`), 451

`insertColumns()` (spinetool-
`box.mvcmodels.map_model.MapModel`
`initializePage()` (spinetool-
`box.widgets.install_julia_wizard.InstallJuliaPage`method), 189
`method`), 451

`insertColumns()` (spinetool-
`box.mvcmodels.minimal_table_model.MinimalTableModel`
`initializePage()` (spinetool-
`box.widgets.install_julia_wizard.SelectDirsPage`method), 195
`method`), 451

`insertFromMimeData()` (spinetool-
`box.widgets.code_text_edit.CodeTextEdit`
`initializePage()` (spinetool-
`box.widgets.install_julia_wizard.SuccessPage`method), 410
`method`), 451

`insertRow()` (spinetool-
`box.mvcmodels.project_item_specification_models.ProjectItemSp`
`inner_push_object_ids()` (spinetool-
`box.spine_db_parcel.SpineDBParcel`method), 204
612

`insertRows()` (spinetool-
`box.mvcmodels.array_model.ArrayModel`
`inner_push_parameter_value_ids()` (spinetool-
`box.spine_db_parcel.SpineDBParcel`method), 176
612

`insertRows()` (spinetool-
`box.mvcmodels.compound_table_model.CompoundTableModel`
`inner_push_relationship_ids()` (spinetool-
`box.spine_db_parcel.SpineDBParcel`method), 179
612

`insertRows()` (spinetool-
`box.mvcmodels.map_model.MapModel`
`inquire_index_name()` (in module spinetool-
`box.helpers`), 531

`insertRows()` (spinetool-
`box.mvcmodels.minimal_tree_model.TreeItem`
`insert_children()` (spinetool-
`box.mvcmodels.minimal_table_model.MinimalTableModel`
`method`), 197
`method`), 194

`insertRows()` (spinetool-
`box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem`
`insert_children()` (spinetool-
`box.mvcmodels.time_pattern_model.TimePatternModel`
`method`), 269
`method`), 210

`insertRows()` (spinetool-
`box.mvcmodels.time_series_model_fixed_resolution.TimeSeriesM`
`insert_children_sorted()` (spinetool-
`box.spine_db_editor.mvcmodels.tree_item_utility.SortChildrenMixin`
`method`), 309
`method`), 212

`insertRows()` (spinetool-
`box.spine_db_editor.widgets.add_items_dialogs.AddRelationshipDialog`
`insert_column()` (spinetool-
`box.spine_db_editor.widgets.add_items_dialogs.AddRelationshipDialog`
`insert_row_class_dialog_series_model_variable_resolution.TimeSeries`

method), 214

insertRows() (spinetool-
box.spine_db_editor.mvcmodels.metadata_table_model_base.
method), 264

INSTALL (spinetoolbox.widgets.install_julia_wizard._PageId
attribute), 450

InstallJuliaPage (class in spinetool-
box.widgets.install_julia_wizard), 451

InstallJuliaWizard (class in spinetool-
box.widgets.install_julia_wizard), 450

InstallPluginDialog (class in spinetool-
box.widgets.plugin_manager_widgets), 485

interpret_icon_id() (in module spinetool-
box.helpers), 524

interrupt_persistent() (spinetool-
box.spine_engine_manager.LocalSpineEngineManager
method), 621

interrupt_persistent() (spinetool-
box.spine_engine_manager.RemoteSpineEngineManager
method), 623

interrupt_persistent() (spinetool-
box.spine_engine_manager.SpineEngineManagerBase
method), 619

INTRO (spinetoolbox.widgets.add_up_spine_opt_wizard._PageId
attribute), 405

INTRO (spinetoolbox.widgets.install_julia_wizard._PageId
attribute), 450

IntroPage (class in spinetool-
box.widgets.add_up_spine_opt_wizard),
406

IntroPage (class in spinetool-
box.widgets.install_julia_wizard), 451

INVALID (spinetoolbox.project.ItemNameStatus at-
tribute), 557

INVALID_CHARS (in module spinetoolbox.config), 504

INVALID_FILENAME_CHARS (in module spinetool-
box.config), 504

invalidate_workflow() (spinetool-
box.project_item.project_item.ProjectItem
method), 223

is_busy (spinetoolbox.fetch_parent.FetchParent prop-
erty), 507

is_clone() (spinetool-
box.spine_db_commands.AgedUndoCommand
method), 591

is_critical (spinetool-
box.project_commands.RenameProjectItemCommand
property), 573

is_critical (spinetool-
box.project_commands.ReplaceSpecificationCommand
property), 576

is_critical (spinetool-
box.project_commands.SpineToolboxCommand
property), 571

is_db_map_editor() (spinetool-
box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase
method), 264

is_deprecated() (spinetool-
box.project_item.project_item_factory.ProjectItemFactory
static method), 226

is_dirty() (spinetool-
box.spine_db_manager.SpineDBManager
method), 600

is_enabled() (spinetool-
box.widgets.custom_qwidgets._MenuToolBar
method), 439

is_expense_column() (spinetool-
box.mvcmodels.map_model.MapModel
method), 190

is_expense_row() (spinetool-
box.mvcmodels.array_model.ArrayModel
method), 176

is_expense_row() (spinetool-
box.mvcmodels.indexed_value_table_model.IndexedValueTableM
method), 187

is_expense_row() (spinetool-
box.mvcmodels.map_model.MapModel
method), 190

is_fetched (spinetoolbox.fetch_parent.FetchParent
property), 507

is_ijulia_installed() (spinetool-
box.widgets.kernel_editor.KernelEditorBase
method), 457

is_leaf_value() (spinetool-
box.mvcmodels.map_model.MapModel
method), 190

is_obsolete (spinetoolbox.fetch_parent.FetchParent
property), 507

is_package_installed() (spinetool-
box.widgets.kernel_editor.KernelEditorBase
static method), 456

is_persistent_command_complete() (spinetool-
box.spine_engine_manager.LocalSpineEngineManager
method), 621

is_persistent_command_complete() (spinetool-
box.spine_engine_manager.RemoteSpineEngineManager
method), 622

is_persistent_command_complete() (spinetool-
box.spine_engine_manager.SpineEngineManagerBase
method), 619

is_specification_name_reserved() (spinetool-
box.project.SpineToolboxProject
method),
560

is_valid() (spinetool-
box.project_upgrader.ProjectUpgrader
method), 587

is_valid() (spinetool-
box.spine_db_editor.mvcmodels.entity_tree_item.RelationshipItem

<i>method</i>), 254	<i>item_category_context_menu()</i> (<i>spinetool-</i> <i>box.ui_main.ToolboxUI method</i>), 638
<i>is_valid()</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem</i> <i>method</i>), 268	<i>item_data</i> (<i>spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioItem</i> <i>static method</i>), 226
<i>is_valid_v1()</i> (<i>spinetool-</i> <i>box.project_upgrader.ProjectUpgrader</i> <i>method</i>), 588	<i>item_data</i> (<i>spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioItem</i> <i>property</i>), 295
<i>is_valid_v11()</i> (<i>spinetool-</i> <i>box.project_upgrader.ProjectUpgrader</i> <i>method</i>), 588	<i>item_data</i> (<i>spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem</i> <i>property</i>), 303
<i>is_valid_v2_to_v8()</i> (<i>spinetool-</i> <i>box.project_upgrader.ProjectUpgrader</i> <i>method</i>), 588	<i>item_data</i> (<i>spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem</i> <i>property</i>), 304
<i>is_valid_v9_to_v10()</i> (<i>spinetool-</i> <i>box.project_upgrader.ProjectUpgrader</i> <i>method</i>), 588	<i>item_data</i> (<i>spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureItem</i> <i>property</i>), 306
<i>isComplete()</i> (<i>spinetool-</i> <i>box.widgets.add_up_spine_opt_wizard.CheckPreviousInstallation</i> <i>method</i>), 406	<i>item_data</i> (<i>spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItem</i> <i>property</i>), 310
<i>isComplete()</i> (<i>spinetool-</i> <i>box.widgets.add_up_spine_opt_wizard.TroubleshootProblemsDialog</i> <i>method</i>), 407	<i>item_dict()</i> (<i>spinetool-</i> <i>box.project_item.project_item.ProjectItem</i> <i>method</i>), 223
<i>isComplete()</i> (<i>spinetool-</i> <i>box.widgets.add_up_spine_opt_wizard.TroubleshootProblemsDialog</i> <i>method</i>), 407	<i>item_dict_local_entries()</i> (<i>spinetool-</i> <i>box.project_item.project_item.ProjectItem</i> <i>static method</i>), 223
<i>isComplete()</i> (<i>spinetool-</i> <i>box.widgets.custom_qwidgets.QWizardProcessPage</i> <i>method</i>), 441	ITEM_EXTENT (<i>spinetool-</i> <i>box.project_item_icon.ProjectItemIcon</i> <i>attribute</i>), 578
<i>isSeparator()</i> (<i>spinetool-</i> <i>box.widgets.custom_qwidgets.TitleWidgetAction</i> <i>method</i>), 440	<i>item_from_index()</i> (<i>spinetool-</i> <i>box.mvcmodels.minimal_tree_model.MinimalTreeModel</i> <i>method</i>), 198
<i>issue_persistent_command()</i> (<i>spinetool-</i> <i>box.spine_engine_manager.LocalSpineEngineManager</i> <i>method</i>), 620	<i>item_id()</i> (<i>spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_model.EmptyParameterModel</i> <i>method</i>), 245
<i>issue_persistent_command()</i> (<i>spinetool-</i> <i>box.spine_engine_manager.RemoteSpineEngineManager</i> <i>method</i>), 623	<i>item_id()</i> (<i>spinetoolbox.spine_db_editor.mvcmodels.single_parameter_model.SingleParameterModel</i> <i>method</i>), 298
<i>issue_persistent_command()</i> (<i>spinetool-</i> <i>box.spine_engine_manager.SpineEngineManagerBase</i> <i>method</i>), 619	<i>item_ids()</i> (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModels</i> <i>method</i>), 299
<i>issues()</i> (<i>spinetoolbox.link.JumpLink</i> <i>method</i>), 539	ITEM_METADATA_ID (<i>spinetool-</i> <i>box.spine_db_editor.mvcmodels.item_metadata_table_model.ExtraItemMetadataTableModel</i> <i>attribute</i>), 257
<i>item</i> (<i>spinetoolbox.link.JumpLink</i> <i>property</i>), 539	<i>item_move_finished</i> (<i>spinetool-</i> <i>box.widgets.custom_qgraphicsscene.CustomGraphicsScene</i> <i>attribute</i>), 421
<i>item</i> (<i>spinetoolbox.link.JumpOrLink</i> <i>property</i>), 538	<i>item_name()</i> (<i>spinetool-</i> <i>box.project_item_icon.ExecutionIcon</i> <i>method</i>), 581
<i>item</i> (<i>spinetoolbox.link.Link</i> <i>property</i>), 539	<i>item_names()</i> (<i>spinetool-</i> <i>box.mvcmodels.project_item_model.ProjectItemModel</i> <i>method</i>), 203
<i>item()</i> (<i>spinetoolbox.mvcmodels.project_item_model.ProjectItemModel</i> <i>method</i>), 201	<i>item_removed</i> (<i>spinetool-</i> <i>box.widgets.plugin_manager_widgets.ManagePluginsDialog</i> <i>attribute</i>), 485
<i>item_about_to_be_removed</i> (<i>spinetool-</i> <i>box.project.SpineToolboxProject</i> <i>attribute</i>), 558	<i>item_renamed</i> (<i>spinetool-</i> <i>box.project.SpineToolboxProject</i> <i>attribute</i>), 558
<i>item_added</i> (<i>spinetoolbox.project.SpineToolboxProject</i> <i>attribute</i>), 558	<i>item_selected</i> (<i>spinetool-</i> <i>box.widgets.plugin_manager_widgets.InstallPluginDialog</i> <i>attribute</i>), 485
<i>item_at_row()</i> (<i>spinetool-</i> <i>box.mvcmodels.compound_table_model.CompoundTableModel</i> <i>method</i>), 178	
<i>item_category()</i> (<i>spinetool-</i> <i>box.project_item.project_item.ProjectItem</i> <i>static method</i>), 220	

- `itemChange()` (*spinetoolbox.project_item_icon.ConnectorButton* method), 581
- `itemChange()` (*spinetoolbox.project_item_icon.ProjectItemIcon* method), 579
- `itemChange()` (*spinetoolbox.spine_db_editor.graphics_items.EntityItem* method), 396
- `itemChange()` (*spinetoolbox.spine_db_editor.graphics_items.RelationshipItem* method), 397
- `ItemMetadataDelegate` (class in *spinetoolbox.spine_db_editor.widgets.custom_delegates*), 333
- `ItemMetadataEditor` (class in *spinetoolbox.spine_db_editor.widgets.item_metadata_editor*), 362
- `ItemMetadataTableModel` (class in *spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model*), 258
- `ItemMetadataTableView` (class in *spinetoolbox.spine_db_editor.widgets.custom_qtableview*), 347
- `ItemNameStatus` (class in *spinetoolbox.project*), 557
- `items()` (*spinetoolbox.mvcmodels.project_item_model.ProjectItemModel* method), 202
- `items_added` (*spinetoolbox.spine_db_manager.SpineDBManager* attribute), 596
- `items_per_category()` (*spinetoolbox.mvcmodels.project_item_model.ProjectItemModel* method), 203
- `items_removed` (*spinetoolbox.spine_db_manager.SpineDBManager* attribute), 597
- `items_to_dict()` (*spinetoolbox.headless.ModifiableProject* method), 513
- `items_updated` (*spinetoolbox.spine_db_manager.SpineDBManager* attribute), 596
- `ItemSpecificationMenu` (class in *spinetoolbox.widgets.custom_menus*), 418
- `ItemTreeView` (class in *spinetoolbox.spine_db_editor.widgets.custom_qtreeview*), 351
- `ItemType` (class in *spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model*), 258
- `ItemTypeFetchParent` (class in *spinetoolbox.fetch_parent*), 509
- J**
- `jill_install` (in module *spinetoolbox.widgets.install_julia_wizard*), 450
- `JillNotFoundPage` (class in *spinetoolbox.widgets.install_julia_wizard*), 451
- `job_id` (*spinetoolbox.spine_engine_worker.SpineEngineWorker* property), 625
- `julia_exe_selected` (*spinetoolbox.widgets.install_julia_wizard.InstallJuliaWizard* attribute), 450
- `JUMP` (*spinetoolbox.helpers.LinkType* attribute), 519
- `jump` (*spinetoolbox.link.JumpLink* property), 539
- `jump_about_to_be_removed` (*spinetoolbox.project.SpineToolboxProject* attribute), 558
- `jump_added` (*spinetoolbox.project.SpineToolboxProject* attribute), 558
- `JUMP_COLOR` (in module *spinetoolbox.link*), 535
- `jump_from_dict()` (*spinetoolbox.project.SpineToolboxProject* method), 560
- `jump_issues()` (*spinetoolbox.project.SpineToolboxProject* method), 565
- `jump_updated` (*spinetoolbox.project.SpineToolboxProject* attribute), 558
- `JumpCommandLineArgsModel` (class in *spinetoolbox.mvcmodels.file_list_models*), 184
- `JumpLink` (class in *spinetoolbox.link*), 539
- `JumpLinkDrawer` (class in *spinetoolbox.link*), 540
- `JumpOrLink` (class in *spinetoolbox.link*), 538
- `JumpPropertiesWidget` (class in *spinetoolbox.widgets.jump_properties_widget*), 452
- `jumps_for_item()` (*spinetoolbox.project.SpineToolboxProject* method), 564
- `jupyter_console_requested` (*spinetoolbox.ui_main.ToolboxUI* attribute), 627
- `JUPYTER_KERNEL_TIME_TO_DEAD` (in module *spinetoolbox.config*), 504
- `JupyterConsoleWidget` (class in *spinetoolbox.widgets.jupyter_console_widget*), 453
- K**
- `kernel_found` (*spinetoolbox.kernel_fetcher.KernelFetcher* attribute), 534
- `kernel_managers()` (*spinetoolbox.spine_engine_manager.LocalSpineEngineManager* method), 620
- `kernel_shutdown` (*spinetoolbox.ui_main.ToolboxUI* attribute), 627

KernelEditorBase (class in <i>spinetoolbox.widgets.kernel_editor</i>), 455	kill_persistent() (<i>spinetoolbox.spine_engine_manager.LocalSpineEngineManager</i> method), 621
KernelFetcher (class in <i>spinetoolbox.kernel_fetcher</i>), 534	kill_persistent() (<i>spinetoolbox.spine_engine_manager.RemoteSpineEngineManager</i> method), 623
KernelsPopupMenu (class in <i>spinetoolbox.widgets.custom_menus</i>), 419	kill_persistent() (<i>spinetoolbox.spine_engine_manager.SpineEngineManagerBase</i> method), 619
key_press_event() (<i>spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget</i> method), 479	Label (class in <i>spinetoolbox.widgets.custom_qgraphicsviews.EntityQGraphicsView</i> method), 338
keyPressEvent() (<i>spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView</i> method), 338	label (<i>spinetoolbox.plotting.IndexName</i> attribute), 547
keyPressEvent() (<i>spinetoolbox.widgets.about_widget.AboutWidget</i> method), 403	label (<i>spinetoolbox.plotting.ParameterTableHeaderSection</i> attribute), 548
keyPressEvent() (<i>spinetoolbox.widgets.add_project_item_widget.AddProjectItemWidget</i> method), 404	label (<i>spinetoolbox.plotting.TreeNode</i> attribute), 547
keyPressEvent() (<i>spinetoolbox.widgets.custom_combobox.OpenProjectDialogComboBox</i> method), 411	LabelWithCopyButton (class in <i>spinetoolbox.widgets.custom_qwidgets</i>), 441
keyPressEvent() (<i>spinetoolbox.widgets.custom_editors.CheckListEditor</i> method), 416	last_child() (<i>spinetoolbox.mvcmodels.minimal_tree_model.TreeItem</i> method), 196
keyPressEvent() (<i>spinetoolbox.widgets.custom_editors.CustomLineEditor</i> method), 414	last_db_map (<i>spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem</i> property), 266
keyPressEvent() (<i>spinetoolbox.widgets.custom_editors.SearchBarEditor</i> method), 415	LATEST_PROJECT_VERSION (in module <i>spinetoolbox.config</i>), 504
keyPressEvent() (<i>spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene</i> method), 421	layout_available (<i>spinetoolbox.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGenerator</i> attribute), 358
keyPressEvent() (<i>spinetoolbox.widgets.custom_qgraphicsviews.CustomQGraphicsView</i> method), 423	LazyFilterCheckboxListModel (class in <i>spinetoolbox.mvcmodels.filter_checkbox_list_model</i>), 185
keyPressEvent() (<i>spinetoolbox.widgets.custom_qtreeview.CustomTreeView</i> method), 435	leaf_indexes() (<i>spinetoolbox.mvcmodels.project_item_model.ProjectItemModel</i> method), 203
keyPressEvent() (<i>spinetoolbox.widgets.custom_qtreeview.SourcesTreeView</i> method), 435	LeafItem (class in <i>spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility</i>), 310
keyPressEvent() (<i>spinetoolbox.widgets.custom_qwidgets._MenuToolBar</i> method), 440	LeafProjectTreeItem (class in <i>spinetoolbox.mvcmodels.project_tree_item</i>), 207
keyPressEvent() (<i>spinetoolbox.widgets.custom_qwidgets.UndoRedoMixin</i> method), 436	LEAVE_AS_IS (<i>spinetoolbox.spine_db_editor.widgets.scenario_generator._ScenarioNameGenerator</i> attribute), 375
keyPressEvent() (<i>spinetoolbox.widgets.persistent_console_widget._CustomLineEdit</i> method), 477	leaveEvent() (<i>spinetoolbox.widgets.notification.Notification</i> method), 470
keyPressEvent() (<i>spinetoolbox.widgets.settings_widget.SettingsWidgetBase</i> method), 493	legend_axes (<i>spinetoolbox.widgets.plot_canvas.PlotCanvas</i> property), 482
	LEGEND_PLACEMENT_THRESHOLD (in module <i>spinetoolbox.plotting</i>), 546
	LegendPosition (class in <i>spinetoolbox.widgets.plot_canvas</i>), 481
	LINE (<i>spinetoolbox.plotting.PlotType</i> attribute), 547

[line_edit](#) (*spinetoolbox.spine_db_editor.widgets.url_toolbar.UrlToolBar* *property*), 392
[line_number_area_paint_event\(\)](#) (*spinetoolbox.widgets.code_text_edit.CodeTextEdit* *method*), 410
[line_number_area_width\(\)](#) (*spinetoolbox.widgets.code_text_edit.CodeTextEdit* *method*), 410
[LineNumberArea](#) (class in *spinetoolbox.widgets.code_text_edit*), 410
[Link](#) (class in *spinetoolbox.link*), 538
[LINK_COLOR](#) (in module *spinetoolbox.link*), 535
[LinkBase](#) (class in *spinetoolbox.link*), 535
[LinkDrawerBase](#) (class in *spinetoolbox.link*), 540
[LinkNotification](#) (class in *spinetoolbox.widgets.notification*), 470
[LinkPropertiesWidget](#) (class in *spinetoolbox.widgets.link_properties_widget*), 460
[LinkType](#) (class in *spinetoolbox.helpers*), 519
[list_index\(\)](#) (*spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item.LinkItem* *method*), 278
[ListItem](#) (class in *spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item.LinkItem*), 277
[load\(\)](#) (*spinetoolbox.project.SpineToolboxProject* *method*), 560
[load_connection_options\(\)](#) (*spinetoolbox.widgets.link_properties_widget.LinkPropertiesWidget* *method*), 461
[load_db_urls\(\)](#) (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase* *method*), 378
[load_empty_expanded_parameter_value_data\(\)](#) (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* *method*), 387
[load_empty_parameter_value_data\(\)](#) (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* *method*), 386
[load_empty_relationship_data\(\)](#) (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* *method*), 385
[load_expanded_parameter_value_data\(\)](#) (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* *method*), 388
[load_full_expanded_parameter_value_data\(\)](#) (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* *method*), 387
[load_full_parameter_value_data\(\)](#) (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* *method*), 387
[load_full_relationship_data\(\)](#) (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* *method*), 386
[load_installed_plugins\(\)](#) (*spinetoolbox.plugin_manager.PluginManager* *method*), 556
[load_local_project_data\(\)](#) (in module *spinetoolbox.helpers*), 532
[load_next_urls\(\)](#) (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase* *method*), 379
[load_parameter_value_data\(\)](#) (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* *method*), 388
[load_plugin_dict\(\)](#) (in module *spinetoolbox.helpers*), 529
[load_plugin_specifications\(\)](#) (in module *spinetoolbox.helpers*), 529
[load_previous_urls\(\)](#) (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase* *method*), 379
[load_project_dict\(\)](#) (in module *spinetoolbox.helpers*), 532
[load_project_items\(\)](#) (in module *spinetoolbox.load_project_items*), 541
[load_relationship_data\(\)](#) (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* *method*), 386
[load_scenario_alternative_data\(\)](#) (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* *method*), 386
[load_specification_from_file\(\)](#) (in module *spinetoolbox.helpers*), 528
[load_specification_local_data\(\)](#) (in module *spinetoolbox.helpers*), 530
[LOCAL_EXECUTION_JOB_ID](#) (*spinetoolbox.project.SpineToolboxProject* *attribute*), 559
[LocalSpineEngineManager](#) (class in *spinetoolbox.spine_engine_manager*), 620
[Logger](#) (*spinetoolbox.project_item.project_item.ProjectItem* *property*), 220
[LoggerInterface](#) (class in *spinetoolbox.logging_connection*), 542
[LoggingConnection](#) (class in *spinetoolbox.project_item.logging_connection*), 216
[LoggingTabularViewMixin](#) (class in *spinetoolbox.project_item.logging_connection*), 219
[LogMixin](#) (class in *spinetoolbox.log_mixin*), 542
[M](#)
[magic_number](#) (*spinetoolbox.link.LinkBase* *property*), 538
[main\(\)](#) (in module *spinetoolbox.main*), 544

main() (in module *spinetoolbox.spine_db_editor.main*), 401
 MainMenu (class in *spinetoolbox.spine_db_editor.widgets.custom_menus*), 333
 MainStatusBar (class in *spinetoolbox.widgets.statusbars*), 498
 MainToolBar (class in *spinetoolbox.widgets.toolbars*), 502
 MAINWINDOW_SS (in module *spinetoolbox.config*), 505
 major (*spinetoolbox.version.VersionInfo* attribute), 641
 make_add_item_widget() (*spinetoolbox.project_item.project_item_factory.ProjectItemFactory* static method), 226
 make_context_menu() (*spinetoolbox.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDBEditor* method), 369
 make_context_menu() (*spinetoolbox.widgets.multi_tab_window.MultiTabWindow* method), 467
 make_db_map_obj_cls_lookup() (*spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetObjectClassMixin* method), 364
 make_db_map_obj_lookup() (*spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetObjectMixin* method), 364
 make_db_map_rel_cls_lookup() (*spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetRelationshipClassesMixin* method), 365
 make_delegate() (*spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ParameterizedPivotTableModel* static method), 290
 make_delegate() (*spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModelBase* static method), 286
 make_delegate() (*spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.RelationshipPivotTableModel* static method), 292
 make_delegate() (*spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ScenarioPivotTableModel* static method), 292
 make_engine_client() (*spinetoolbox.spine_engine_manager.RemoteSpineEngineManager* method), 622
 make_engine_manager() (in module *spinetoolbox.spine_engine_manager*), 623
 make_feature_name() (*spinetoolbox.spine_db_editor.mvcmodels.tool_feature_model.ToolFeatureModel* static method), 307
 make_figure_graphics_item() (in module *spinetoolbox.spine_db_editor.graphics_items*), 394
 make_heat_map() (in module *spinetoolbox.spine_db_editor.widgets.graph_layout_generator*), 358
 make_icon() (*spinetoolbox.project_item.project_item_factory.ProjectItemFactory* static method), 227
 make_icon_background() (in module *spinetoolbox.helpers*), 528
 make_icon_id() (in module *spinetoolbox.helpers*), 524
 make_icon_toolbar_ss() (in module *spinetoolbox.helpers*), 528
 make_item() (*spinetoolbox.project_item.project_item_factory.ProjectItemFactory* static method), 227
 make_item_properties_uis() (*spinetoolbox.ui_main.ToolboxUI* method), 630
 make_items_menu() (*spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView* method), 366
 make_julia_kernel() (*spinetoolbox.widgets.kernel_editor.KernelEditorBase* method), 457
 make_julia_kernel() (*spinetoolbox.widgets.settings_widget.SettingsWidget* method), 455
 make_kernel() (*spinetoolbox.widgets.kernel_editor.KernelEditorBase* method), 455
 make_model() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddOrManageRelationships* method), 365
 make_model() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddRelationships* method), 365
 make_model() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.ManageRelationships* method), 365
 make_pivot_headers() (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* method), 365
 make_properties_widget() (*spinetoolbox.project_item.project_item_factory.ProjectItemFactory* static method), 227
 make_python_kernel() (*spinetoolbox.widgets.kernel_editor.KernelEditorBase* method), 456
 make_python_kernel() (*spinetoolbox.widgets.settings_widget.SettingsWidget* method), 495
 make_room_for_item() (*spinetoolbox.project_item.project_item_factory.ProjectItemFactory* static method), 227
 make_settings_dict_for_engine() (in module *spinetoolbox.helpers*), 527
 make_signal_handler_dict() (*spinetoolbox.project_item.project_item_factory.ProjectItemFactory* static method), 221

make_specification_editor() (spinetool- **ManageRelationshipsDialog** (class in spinetool-
 box.project_item.project_item_factory.ProjectItemFactory box.spine_db_editor.widgets.add_items_dialogs),
 static method), 227 318
make_specification_menu() (spinetool- **MAP** (spinetoolbox.widgets.parameter_value_editor_base.ValueType
 box.project_item.project_item_factory.ProjectItemFactory attribute), 475
 static method), 227 **map_from_sub()** (spinetool-
make_table_view() (spinetool- box.mvcmodels.compound_table_model.CompoundTableModel
 box.spine_db_editor.widgets.add_items_dialogs.AddReadyRelationshipDialog
 method), 315 **map_to_pivot()** (spinetool-
make_table_view() (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableM
 box.spine_db_editor.widgets.manage_items_dialogs.ManageItemsDialogBase
 method), 364 **map_to_sub()** (spinetool-
make_unique_importer_specification_name() box.mvcmodels.compound_table_model.CompoundTableModel
 (spinetoolbox.project_upgrader.ProjectUpgrader method), 177
 static method), 587 **MapEditor** (class in spinetoolbox.widgets.map_editor),
MakeRelationshipOnTheFlyMixin (class in spinetool- 461
 box.spine_db_editor.mvcmodels.parameter_mixinMapModel (class in spinetool-
 276 box.mvcmodels.map_model), 188
manage_members() (spinetool- **MapTableContextMenu** (class in spinetool-
 box.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView box.widgets.indexed_value_table_context_menu),
 method), 350 447
manage_relationships() (spinetool- **MapView** (class in spinetool-
 box.spine_db_editor.widgets.custom_qtreeview.EntityTreeView box.widgets.custom_qtableview), 431
 method), 349 **MapValueEditor** (class in spinetool-
ManageEntityClassesDelegate (class in spinetool- box.widgets.map_value_editor), 462
 box.spine_db_editor.widgets.custom_delegates), 332 **mark_execution_finished()** (spinetool-
ManageItemsDelegate (class in spinetool- box.project_item_icon.ExecutionIcon method),
 box.spine_db_editor.widgets.custom_delegates), 331 582
ManageItemsDialog (class in spinetool- **mark_execution_ignored()** (spinetool-
 box.spine_db_editor.widgets.manage_items_dialogmark 582
 364 box.project_item_icon.ExecutionIcon method),
ManageItemsDialogBase (class in spinetool- **mark_execution_started()** (spinetool-
 box.spine_db_editor.widgets.manage_items_dialogmark 582
 363 box.project_item_icon.ExecutionIcon method),
ManageMembersDialog (class in spinetool- **mark_execution_waiting()** (spinetool-
 box.spine_db_editor.widgets.add_items_dialogs), 582
 319 box.project_item_icon.ExecutionIcon method),
ManageObjectClassesDelegate (class in spinetool- **mass_export_items()** (spinetool-
 box.spine_db_editor.widgets.custom_delegates), 332 box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase
 method), 380
ManageObjectsDelegate (class in spinetool- **MassExportItemsDialog** (class in spinetool-
 box.spine_db_editor.widgets.custom_delegates), 332 box.spine_db_editor.widgets.mass_select_items_dialogs),
 366
ManagePluginsDialog (class in spinetool- **MassRemoveItemsDialog** (class in spinetool-
 box.widgets.plugin_manager_widgets), 485 box.spine_db_editor.widgets.mass_select_items_dialogs),
 366
ManageRelationshipClassesDelegate (class in spinetool- **MassSelectItemsDialog** (class in spinetool-
 box.spine_db_editor.widgets.custom_delegates), 332 box.spine_db_editor.widgets.mass_select_items_dialogs),
 366
ManageRelationshipsDelegate (class in spinetool- **max()** (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_b
 box.spine_db_editor.widgets.custom_delegates), 333 static method), 262
 333 **may_have_filters()** (spinetool-
 box.spine_db_editor.widgets.custom_delegates), 333 box.project_item.logging_connection.LoggingConnection
 method), 217

[may_have_write_index\(\)](#) (*spinetoolbox.project_item.logging_connection.LoggingConnection* method), 296
[may_purge_before_writing\(\)](#) (*spinetoolbox.project_item.logging_connection.LoggingConnection* method), 217
[may_use_datapackage\(\)](#) (*spinetoolbox.project_item.logging_connection.LoggingConnection* method), 218
[may_use_memory_db\(\)](#) (*spinetoolbox.project_item.logging_connection.LoggingConnection* method), 217
[MemberObjectItem](#) (class in *spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item*), 253
[MembersItem](#) (class in *spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item*), 251
[MenuItemToolBarWidget](#) (class in *spinetoolbox.widgets.custom_qwidgets*), 438
[merge_dicts\(\)](#) (in module *spinetoolbox.helpers*), 532
[METADATA_ID](#) (*spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model.ExtraColumn* attribute), 258
[metadata_model\(\)](#) (*spinetoolbox.spine_db_editor.widgets.metadata_editor.MetadataEditor* method), 368
[MetadataEditor](#) (class in *spinetoolbox.spine_db_editor.widgets.metadata_editor*), 367
[MetadataTableModel](#) (class in *spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model*), 260
[MetadataTableModelBase](#) (class in *spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model*), 262
[MetadataTableView](#) (class in *spinetoolbox.spine_db_editor.widgets.custom_qtableview*), 347
[MetadataTableViewBase](#) (class in *spinetoolbox.spine_db_editor.widgets.custom_qtableview*), 347
[MetaObject](#) (class in *spinetoolbox.metaobject*), 544
[micro](#) (*spinetoolbox.version.VersionInfo* attribute), 641
[mimeType\(\)](#) (*spinetoolbox.mvcmodels.file_list_models.CommandLineArgsModel* method), 184
[mimeType\(\)](#) (*spinetoolbox.mvcmodels.file_list_models.FileListModel* method), 182
[mimeType\(\)](#) (*spinetoolbox.spine_db_editor.mvcmodels.alternative_model.AlternativeModel* method), 236
[mimeType\(\)](#) (*spinetoolbox.spine_db_editor.mvcmodels.scenario_model.ScenarioModel* method), 296
[mimeType\(\)](#) (*spinetoolbox.spine_db_editor.mvcmodels.tool_feature_model.ToolFeatureModel* method), 307
[min_pos](#) (*spinetoolbox.widgets.persistent_console_widget._CustomLineEdit* property), 476
[MinimalJuliaKernelEditor](#) (class in *spinetoolbox.widgets.kernel_editor*), 459
[MinimalTableModel](#) (class in *spinetoolbox.mvcmodels.minimal_table_model*), 193
[MinimalTreeModel](#) (class in *spinetoolbox.mvcmodels.minimal_tree_model*), 198
[MiniPythonKernelEditor](#) (class in *spinetoolbox.widgets.kernel_editor*), 458
[minor](#) (*spinetoolbox.version.VersionInfo* attribute), 641
[model](#) (*spinetoolbox.mvcmodels.minimal_tree_model.TreeItem* property), 196
[model](#) (*spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLevelTableModel* property), 282
[model_data_changed](#) (*spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel* attribute), 285
[model_data_changed](#) (*spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel* attribute), 293
[ModifiableProject](#) (class in *spinetoolbox.headless*), 512
[module](#)
 spinetoolbox, 175
 spinetoolbox.__main__, 503
 spinetoolbox._version, 504
 spinetoolbox.config, 504
 spinetoolbox.execution_managers, 505
 spinetoolbox.fetch_parent, 507
 spinetoolbox.headless, 511
 spinetoolbox.helpers, 516
 spinetoolbox.kernel_fetcher, 534
 spinetoolbox.link, 535
 spinetoolbox.load_project_items, 541
 spinetoolbox.log_mixin, 541
 spinetoolbox.logger_interface, 542
 spinetoolbox.main, 543
 spinetoolbox.metaobject, 544
 spinetoolbox.mvcmodels, 175
 spinetoolbox.mvcmodels.array_model, 175
 spinetoolbox.mvcmodels.compound_table_model, 177
 spinetoolbox.mvcmodels.empty_row_model, 181
 spinetoolbox.mvcmodels.file_list_models, 182
 spinetoolbox.mvcmodels.filter_checkbox_list_model, 184

[spinetoolbox.mvcmodels.filter_execution_model](#), [spinetoolbox.spine_db_editor.mvcmodels.colors](#),
 186 [237](#)
[spinetoolbox.mvcmodels.indexed_value_table_model](#), [spinetoolbox.spine_db_editor.mvcmodels.compound_paramete](#)
 187 [237](#)
[spinetoolbox.mvcmodels.map_model](#), [188](#) [spinetoolbox.spine_db_editor.mvcmodels.empty_parameter](#)
[spinetoolbox.mvcmodels.minimal_table_model](#), [244](#)
 193 [spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item](#)
[spinetoolbox.mvcmodels.minimal_tree_model](#), [248](#)
 196 [spinetoolbox.spine_db_editor.mvcmodels.entity_tree_model](#)
[spinetoolbox.mvcmodels.project_item_model](#), [254](#)
 199 [spinetoolbox.spine_db_editor.mvcmodels.frozen_table_model](#)
[spinetoolbox.mvcmodels.project_item_specification_model](#), [255](#)
 203 [spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table](#)
[spinetoolbox.mvcmodels.project_tree_item](#), [257](#)
 205 [spinetoolbox.spine_db_editor.mvcmodels.metadata_table](#)
[spinetoolbox.mvcmodels.resource_filter_model](#), [260](#)
 208 [spinetoolbox.spine_db_editor.mvcmodels.metadata_table](#)
[spinetoolbox.mvcmodels.shared](#), [210](#)
[spinetoolbox.mvcmodels.time_pattern_model](#), [spinetoolbox.spine_db_editor.mvcmodels.mime_types](#),
 210 [265](#)
[spinetoolbox.mvcmodels.time_series_model_fixed_resolution](#), [spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item](#)
 211 [266](#)
[spinetoolbox.mvcmodels.time_series_model_variable_resolution](#), [spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_model](#)
 213 [270](#)
[spinetoolbox.plotting](#), [545](#) [spinetoolbox.spine_db_editor.mvcmodels.parameter_mixin](#)
[spinetoolbox.plugin_manager](#), [555](#) [271](#)
[spinetoolbox.project](#), [557](#) [spinetoolbox.spine_db_editor.mvcmodels.parameter_value](#)
[spinetoolbox.project_commands](#), [571](#) [277](#)
[spinetoolbox.project_item](#), [215](#) [spinetoolbox.spine_db_editor.mvcmodels.parameter_value](#)
[spinetoolbox.project_item.logging_connection](#), [278](#)
 215 [spinetoolbox.spine_db_editor.mvcmodels.pivot_model](#),
[spinetoolbox.project_item.project_item](#), [279](#)
 219 [spinetoolbox.spine_db_editor.mvcmodels.pivot_table_model](#)
[spinetoolbox.project_item.project_item_factory](#), [281](#)
 226 [spinetoolbox.spine_db_editor.mvcmodels.scenario_item](#),
[spinetoolbox.project_item.specification_editor_window](#), [281](#)
 228 [spinetoolbox.spine_db_editor.mvcmodels.scenario_model](#),
[spinetoolbox.project_item_icon](#), [577](#) [296](#)
[spinetoolbox.project_settings](#), [583](#) [spinetoolbox.spine_db_editor.mvcmodels.single_parameter](#)
[spinetoolbox.project_upgrader](#), [583](#) [297](#)
[spinetoolbox.qthread_pool_executor](#), [589](#) [spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item](#)
[spinetoolbox.server](#), [231](#) [302](#)
[spinetoolbox.server.engine_client](#), [231](#) [spinetoolbox.spine_db_editor.mvcmodels.tool_feature_model](#)
[spinetoolbox.spine_db_commands](#), [590](#) [306](#)
[spinetoolbox.spine_db_editor](#), [235](#) [spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility](#)
[spinetoolbox.spine_db_editor.graphics_items](#), [307](#)
 394 [spinetoolbox.spine_db_editor.mvcmodels.tree_model_base](#)
[spinetoolbox.spine_db_editor.main](#), [401](#) [311](#)
[spinetoolbox.spine_db_editor.mvcmodels](#), [spinetoolbox.spine_db_editor.mvcmodels.utils](#),
 235 [312](#)
[spinetoolbox.spine_db_editor.mvcmodels.alternative_model](#), [spinetoolbox.spine_db_editor.scenario_generation](#),
 235 [402](#)
[spinetoolbox.spine_db_editor.mvcmodels.alternative_model](#), [spinetoolbox.spine_db_editor.ui](#), [313](#)
 236 [spinetoolbox.spine_db_editor.ui.scenario_generator](#),

313 spinetoolbox.spine_db_editor.widgets.tree_view_mixin,
 spinetoolbox.spine_db_editor.ui.select_databases, 390
 313 spinetoolbox.spine_db_editor.widgets.url_toolbar,
 spinetoolbox.spine_db_editor.ui.spine_db_editor_window,
 313 spinetoolbox.spine_db_icon_manager, 594
 spinetoolbox.spine_db_editor.widgets, 314 spinetoolbox.spine_db_manager, 596
 spinetoolbox.spine_db_editor.widgets.add_items_dialog, spinetoolbox.spine_db_parcel, 611
 314 spinetoolbox.spine_db_worker, 613
 spinetoolbox.spine_db_editor.widgets.commit_widget, spinetoolbox.spine_engine_manager, 618
 320 spinetoolbox.spine_engine_worker, 624
 spinetoolbox.spine_db_editor.widgets.custom_delegates, spinetoolbox.ui_main, 627
 321 spinetoolbox.version, 641
 spinetoolbox.spine_db_editor.widgets.custom_menus, spinetoolbox.widgets, 402
 333 spinetoolbox.widgets.about_widget, 402
 spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews, spinetoolbox.widgets.add_project_item_widget,
 335 404
 spinetoolbox.spine_db_editor.widgets.custom_qtableview, spinetoolbox.widgets.add_up_spine_opt_wizard,
 338 405
 spinetoolbox.spine_db_editor.widgets.custom_qtreeview, spinetoolbox.widgets.array_editor, 408
 348 spinetoolbox.widgets.array_value_editor,
 spinetoolbox.spine_db_editor.widgets.custom_qwidgets, 408
 354 spinetoolbox.widgets.code_text_edit, 409
 spinetoolbox.spine_db_editor.widgets.edit_or_remove_dialogs, spinetoolbox.widgets.commit_dialog, 410
 355 spinetoolbox.widgets.custom_combobox, 411
 spinetoolbox.spine_db_editor.widgets.graph_layout_generator, spinetoolbox.widgets.custom_delegates,
 358 412
 spinetoolbox.spine_db_editor.widgets.graph_view_expansion, spinetoolbox.widgets.custom_editors, 413
 359 spinetoolbox.widgets.custom_menus, 417
 spinetoolbox.spine_db_editor.widgets.item_metadata_editor, spinetoolbox.widgets.custom_qcombobox,
 362 420
 spinetoolbox.spine_db_editor.widgets.manage_items_dialog, spinetoolbox.widgets.custom_qgraphicsscene,
 363 421
 spinetoolbox.spine_db_editor.widgets.mass_select_items_dialog, spinetoolbox.widgets.custom_qgraphicsviews,
 365 423
 spinetoolbox.spine_db_editor.widgets.metadata_editor, spinetoolbox.widgets.custom_qlineedit, 426
 367 spinetoolbox.spine_db_editor.widgets.custom_qtableview,
 368 427
 spinetoolbox.spine_db_editor.widgets.object_name_editor, spinetoolbox.widgets.custom_qtextbrowser,
 370 432
 spinetoolbox.spine_db_editor.widgets.parameters_dialog, spinetoolbox.widgets.custom_qtreeview,
 371 434
 spinetoolbox.spine_db_editor.widgets.pivot_table_metadata_editor, spinetoolbox.widgets.custom_qwidgets, 435
 373 spinetoolbox.widgets.datetime_editor, 443
 spinetoolbox.spine_db_editor.widgets.scenario_editor, spinetoolbox.widgets.duration_editor, 444
 375 spinetoolbox.widgets.indexed_value_table_context_menu,
 spinetoolbox.spine_db_editor.widgets.select_position_parameters_dialog,
 377 spinetoolbox.widgets.install_julia_wizard,
 spinetoolbox.spine_db_editor.widgets.spine_db_editor, 449
 378 spinetoolbox.widgets.jump_properties_widget,
 spinetoolbox.spine_db_editor.widgets.tabular_view_header_widget,
 383 spinetoolbox.widgets.jupyter_console_widget,
 spinetoolbox.spine_db_editor.widgets.tabular_view_mixin,
 384 spinetoolbox.widgets.kernel_editor, 455

spinetoolbox.widgets.link_properties_widget, 460
 spinetoolbox.widgets.map_editor, 461
 spinetoolbox.widgets.map_value_editor, 462
 spinetoolbox.widgets.multi_tab_spec_editor, 462
 spinetoolbox.widgets.multi_tab_window, 463
 spinetoolbox.widgets.notification, 469
 spinetoolbox.widgets.open_project_widget, 471
 spinetoolbox.widgets.parameter_value_editor, 474
 spinetoolbox.widgets.parameter_value_editor_base, 474
 spinetoolbox.widgets.persistent_console_widget, 476
 spinetoolbox.widgets.plain_parameter_value_editor, 481
 spinetoolbox.widgets.plot_canvas, 481
 spinetoolbox.widgets.plot_widget, 482
 spinetoolbox.widgets.plugin_manager_widgets, 484
 spinetoolbox.widgets.project_item_drag, 485
 spinetoolbox.widgets.properties_widget, 489
 spinetoolbox.widgets.report_plotting_failure, 490
 spinetoolbox.widgets.select_database_items, 490
 spinetoolbox.widgets.set_description_dialog, 492
 spinetoolbox.widgets.settings_widget, 492
 spinetoolbox.widgets.statusbars, 498
 spinetoolbox.widgets.time_pattern_editor, 498
 spinetoolbox.widgets.time_series_fixed_resolution_editor, 499
 spinetoolbox.widgets.time_series_variable_resolution_editor, 500
 spinetoolbox.widgets.toolbars, 501
 MonoSpaceFontTextBrowser (class in spinetoolbox.widgets.custom_qtextbrowser), 434
 mouseDoubleClickEvent() (spinetoolbox.spine_db_editor.graphics_items.ObjectItem method), 398
 mouseDoubleClickEvent() (spinetoolbox.widgets.project_item_drag.ProjectItemButton method), 487
 mouseDoubleClickEvent() (spinetoolbox.widgets.project_item_drag.ProjectItemSpecButton method), 487
 mouseMoveEvent() (spinetoolbox.spine_db_editor.graphics_items.CrossHairsItem method), 400
 mouseMoveEvent() (spinetoolbox.spine_db_editor.graphics_items.EntityItem method), 396
 mouseMoveEvent() (spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 337
 mouseMoveEvent() (spinetoolbox.spine_db_editor.widgets.tabular_view_header_widget.TabularHeaderView method), 384
 mouseMoveEvent() (spinetoolbox.widgets.about_widget.AboutWidget method), 403
 mouseMoveEvent() (spinetoolbox.widgets.custom_editors.CheckListEditor method), 416
 mouseMoveEvent() (spinetoolbox.widgets.custom_qcombobox.CustomQComboBox method), 420
 mouseMoveEvent() (spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene method), 421
 mouseMoveEvent() (spinetoolbox.widgets.multi_tab_window.TabBarPlus method), 468
 mouseMoveEvent() (spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget method), 478
 mouseMoveEvent() (spinetoolbox.widgets.project_item_drag.ProjectItemDragMixin method), 486
 mouseMoveEvent() (spinetoolbox.widgets.settings_widget.SettingsWidgetBase method), 493
 mousePressEvent() (spinetoolbox.link.JumpOrLink method), 538
 mousePressEvent() (spinetoolbox.widgets.project_item_icon.ConnectorButton method), 501
 mousePressEvent() (spinetoolbox.widgets.project_item_icon.ProjectItemIcon method), 579
 mousePressEvent() (spinetoolbox.spine_db_editor.graphics_items.ArcItem method), 399
 mousePressEvent() (spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 337
 mousePressEvent() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349
 mousePressEvent() (spinetoolbox.widgets.project_item_spec_button.ProjectItemSpecButton method), 487

	<i>box.spine_db_editor.widgets.tabular_view_header_widget.TabularViewHeaderWidget</i> (method), 384		<i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i> (method), 478
<code>mousePressEvent()</code>	(spinetool- <i>box.widgets.about_widget.AboutWidget</i> method), 403	<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.project_item_drag.ProjectItemDragMixin</i> method), 486
<code>mousePressEvent()</code>	(spinetool- <i>box.widgets.custom_editors.CheckListEditor</i> method), 416	<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.settings_widget.SettingsWidgetBase</i> method), 493
<code>mousePressEvent()</code>	(spinetool- <i>box.widgets.custom_editors.SearchBarEditor</i> method), 415	<code>move_tab()</code>	(spinetool- <i>box.widgets.multi_tab_window.MultiTabWindow</i> method), 466
<code>mousePressEvent()</code>	(spinetool- <i>box.widgets.custom_qgraphicsscene.DesignGraphicsScene</i> (method), 421	<code>moveBy()</code>	(spinetoolbox.link.LinkBase method), 536
<code>mousePressEvent()</code>	(spinetool- <i>box.widgets.custom_qgraphicsviews.CustomQGraphicsView</i> (method), 423	<code>moveBy()</code>	(spinetoolbox.spine_db_editor.graphics_items.ArcItem method), 399
<code>mousePressEvent()</code>	(spinetool- <i>box.widgets.custom_qgraphicsviews.CustomQGraphicsView</i> (method), 423	<code>moveBy()</code>	(spinetoolbox.spine_db_editor.graphics_items.EntityItem method), 395
<code>mousePressEvent()</code>	(spinetool- <i>box.widgets.multi_tab_window.TabBarPlus</i> method), 468	<code>moveColumns()</code>	(spinetool- <i>box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel</i> (method), 256
<code>mousePressEvent()</code>	(spinetool- <i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i> (method), 478	<code>moveEvent()</code>	(spinetool- <i>box.spine_db_editor.widgets.commit_viewer._AffectedItemsFromCommit</i> (method), 320
<code>mousePressEvent()</code>	(spinetool- <i>box.widgets.project_item_drag.ProjectItemButtonBase</i> (method), 486	<code>moveEvent()</code>	(spinetool- <i>box.spine_db_editor.widgets.url_toolbar._FilterArrayWidget</i> (method), 393
<code>mousePressEvent()</code>	(spinetool- <i>box.widgets.settings_widget.SettingsWidgetBase</i> (method), 493	<code>MoveIconCommand</code>	(class in spinetool- <i>box.project_commands</i>), 572
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.project_item_icon.ProjectItemIcon</i> method), 579	<code>msg</code>	(spinetoolbox.headless.HeadlessLogger attribute), 512
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.spine_db_editor.widgets.custom_qgraphicsviews.DesignGraphicsView</i> (method), 337	<code>msg</code>	(spinetoolbox.logger_interface.LoggerInterface attribute), 543
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.spine_db_editor.widgets.tabular_view_header_widget.TabularViewHeaderWidget</i> (method), 384	<code>msg</code>	(spinetoolbox.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGenerator attribute), 358
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.about_widget.AboutWidget</i> method), 403	<code>msg</code>	(spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor attribute), 378
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.custom_qgraphicsscene.DesignGraphicsScene</i> (method), 421	<code>msg</code>	(spinetoolbox.ui_main.ToolboxUI attribute), 627
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.custom_qgraphicsviews.CustomQGraphicsView</i> (method), 423	<code>msg</code>	(spinetoolbox.spine_db_editor.widgets.qwizard_process_page.QWizardProcessPage attribute), 441
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.multi_tab_window.MultiTabWindow</i> method), 467	<code>msg_error</code>	(spinetoolbox.headless.HeadlessLogger attribute), 512
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.multi_tab_window.TabBarPlus</i> method), 468	<code>msg_error</code>	(spinetoolbox.logger_interface.LoggerInterface attribute), 543
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.persistent_console_widget.PersistentConsoleWidget</i> (method), 478	<code>msg_error</code>	(spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel attribute), 262
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.project_item_drag.ProjectItemButtonBase</i> (method), 486	<code>msg_error</code>	(spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor attribute), 378
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.settings_widget.SettingsWidgetBase</i> (method), 493	<code>msg_error</code>	(spinetoolbox.ui_main.ToolboxUI attribute), 627
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.tabular_view_header_widget.TabularViewHeaderWidget</i> (method), 384	<code>msg_error</code>	(spinetoolbox.widgets.custom_qwidgets.QWizardProcessPage attribute), 441
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.tabular_view_header_widget.TabularViewHeaderWidget</i> (method), 384	<code>msg_proc</code>	(spinetoolbox.headless.HeadlessLogger attribute), 512
<code>mouseReleaseEvent()</code>	(spinetool- <i>box.widgets.tabular_view_header_widget.TabularViewHeaderWidget</i> (method), 384	<code>msg_proc</code>	(spinetoolbox.logger_interface.LoggerInterface attribute), 543

[msg_proc \(spinetoolbox.ui_main.ToolboxUI attribute\), 627](#)
[msg_proc \(spinetoolbox.widgets.custom_qwidgets.QWizardProcessPage attribute\), 441](#)
[msg_proc_error \(spinetoolbox.headless.HeadlessLogger attribute\), 512](#)
[msg_proc_error \(spinetoolbox.logger_interface.LoggerInterface attribute\), 543](#)
[msg_proc_error \(spinetoolbox.ui_main.ToolboxUI attribute\), 627](#)
[msg_proc_error \(spinetoolbox.widgets.custom_qwidgets.QWizardProcessPage attribute\), 441](#)
[msg_success \(spinetoolbox.headless.HeadlessLogger attribute\), 512](#)
[msg_success \(spinetoolbox.logger_interface.LoggerInterface attribute\), 543](#)
[msg_success \(spinetoolbox.ui_main.ToolboxUI attribute\), 627](#)
[msg_success \(spinetoolbox.widgets.custom_qwidgets.QWizardProcessPage attribute\), 441](#)
[msg_warning \(spinetoolbox.headless.HeadlessLogger attribute\), 512](#)
[msg_warning \(spinetoolbox.logger_interface.LoggerInterface attribute\), 543](#)
[msg_warning \(spinetoolbox.ui_main.ToolboxUI attribute\), 627](#)
[msg_warning \(spinetoolbox.widgets.custom_qwidgets.QWizardProcessPage attribute\), 441](#)
[MultiDBTreeItem \(class in spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item\), 266](#)
[MultiDBTreeModel \(class in spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_model\), 270](#)
[MultiSpineDBEditor \(class in spinetoolbox.spine_db_editor.widgets.multi_spine_db_editor\), 368](#)
[MultiTabSpecEditor \(class in spinetoolbox.widgets.multi_tab_spec_editor\), 462](#)
[MultiTabWindow \(class in spinetoolbox.widgets.multi_tab_window\), 464](#)

N

[n_items\(\) \(spinetoolbox.mvcmodels.project_item_model.ProjectItemModel method\), 202](#)
[name \(spinetoolbox.link.JumpLink property\), 539](#)
[name \(spinetoolbox.link.Link property\), 539](#)
[NAME \(spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModel attribute\), 262](#)
[Name \(spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftPivotTableModel property\), 283](#)
[name \(spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftPivotTableModel property\), 284](#)
[name \(spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftPivotTableModel property\), 282](#)
[name \(spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftPivotTableModel property\), 283](#)
[name \(spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftPivotTableModel property\), 283](#)
[name \(spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftPivotTableModel property\), 284](#)
[name \(spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftPivotTableModel property\), 284](#)
[name \(spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftPivotTableModel property\), 310](#)
[name\(\) \(spinetoolbox.project_item.specification_editor_window._SpecNameEditor method\), 231](#)
[name\(\) \(spinetoolbox.project_item_icon.ProjectItemIcon method\), 578](#)
[name\(\) \(spinetoolbox.widgets.jupyter_console_widget.JupyterConsoleWidget method\), 454](#)
[name\(\) \(spinetoolbox.widgets.multi_tab_window.MultiTabWindow method\), 465](#)
[name\(\) \(spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget method\), 477](#)
[name_changed \(spinetoolbox.project_item.specification_editor_window._SpecNameEditor attribute\), 230](#)
[new_db_editor\(\) \(spinetoolbox.ui_main.ToolboxUI method\), 633](#)
[new_kernel_name\(\) \(spinetoolbox.widgets.kernel_editor.KernelEditorBase method\), 455](#)
[new_line\(\) \(spinetoolbox.widgets.persistent_console_widget._CustomLineEdit method\), 476](#)
[new_line_indent \(spinetoolbox.widgets.persistent_console_widget._CustomLineEdit property\), 476](#)
[new_project\(\) \(spinetoolbox.ui_main.ToolboxUI method\), 629](#)
[new_tab_title \(spinetoolbox.widgets.multi_tab_spec_editor.MultiTabSpecEditor property\), 462](#)
[new_tab_title \(spinetoolbox.widgets.multi_tab_window.MultiTabWindow property\), 464](#)
[NewCommandLineArgItem \(class in spinetoolbox.mvcmodels.file_list_models\), 183](#)
[next_sibling\(\) \(spinetoolbox.mvcmodels.minimal_tree_model.TreeItem method\), 196](#)
[nextId\(\) \(spinetoolbox.widgets.add_up_spine_opt_wizard.AddUpSpineOptionsWizard method\), 539](#)

method), 408

nextId() (spinetoolbox.widgets.add_up_spine_opt_wizard.AddUpSpineOptWizard.project.SpineToolboxProject method), 407

nextId() (spinetoolbox.widgets.add_up_spine_opt_wizard.CriticalResourceChanges method), 407

nextId() (spinetoolbox.widgets.add_up_spine_opt_wizard.FailurePage method), 567

nextId() (spinetoolbox.widgets.add_up_spine_opt_wizard.IntroPage(spinetoolbox.project.SpineToolboxProject method), 406

nextId() (spinetoolbox.widgets.add_up_spine_opt_wizard.NotifyResourceReplacement method), 407

nextId() (spinetoolbox.widgets.add_up_spine_opt_wizard.SelectJuliaPage method), 567

nextId() (spinetoolbox.widgets.add_up_spine_opt_wizard.SuccessPage.spine_db_manager.SpineDBManager method), 407

nextId() (spinetoolbox.widgets.add_up_spine_opt_wizard.TotalFailurePage method), 408

nextId() (spinetoolbox.widgets.add_up_spine_opt_wizard.TroubleshootProblemsPage method), 407

nextId() (spinetoolbox.widgets.add_up_spine_opt_wizard.TroubleshootSolutionPage method), 407

nextId() (spinetoolbox.widgets.install_julia_wizard.FailurePage method), 451

nextId() (spinetoolbox.widgets.install_julia_wizard.InstallJuliaPage method), 451

nextId() (spinetoolbox.widgets.install_julia_wizard.IntroPage method), 451

nextId() (spinetoolbox.widgets.install_julia_wizard.SelectDirPage method), 451

nextId() (spinetoolbox.widgets.install_julia_wizard.SuccessPage method), 451

NiceButton (class in spinetoolbox.widgets.project_item_drag), 486

NO_CONFLICT (spinetoolbox.spine_db_editor.widgets.scenario_generator.ScenarioNameResolution attribute), 375

node_successors() (in module spinetoolbox.project), 570

non_empty_children (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.EmptyChildMixin property), 309

non_empty_children (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeNode property), 308

NONE (spinetoolbox.server.engine_client.ClientSecurityModel attribute), 232

Notification (class in spinetoolbox.widgets.notification), 469

notify_destination() (spinetoolbox.project_item.project_item.ProjectItem method), 225

notify_item_move() (spinetoolbox.project_item_icon.ProjectItemIcon method), 579

notify_resource_changes_to_predecessors() (spinetoolbox.project.SpineToolboxProject method), 567

notify_resource_changes_to_successors() (spinetoolbox.project.SpineToolboxProject method), 567

notify_resource_replacement_to_predecessors() (spinetoolbox.project.SpineToolboxProject method), 568

notify_resource_replacement_to_successors() (spinetoolbox.project.SpineToolboxProject method), 567

notify_session_committed() (spinetoolbox.spine_db_manager.SpineDBManager method), 601

O

object_class_name_list() (spinetoolbox.spine_db_editor.graphics_items.RelationshipItem property), 397

object_class_name_list() (spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetObjectClassNames method), 364

object_id_list() (spinetoolbox.spine_db_editor.graphics_items.RelationshipItem method), 397

object_name_list (spinetoolbox.spine_db_editor.graphics_items.RelationshipItem property), 397

object_name_list (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.RelationshipItem property), 253

object_name_list() (spinetoolbox.spine_db_editor.widgets.manage_items_dialogs.GetObjectsMethod method), 364

object_name_list_editor_requested (spinetoolbox.spine_db_editor.widgets.custom_delegates.ObjectNameListDelegate attribute), 328

object_selection_changed (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView attribute), 349

ObjectClassItem (class in spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item), 249

ObjectClassNameDelegate (class in spinetoolbox.spine_db_editor.widgets.custom_delegates), 327

ObjectGroupDialogBase (class in spinetoolbox.spine_db_editor.widgets.add_items_dialogs), 318

ObjectItem (class in spinetoolbox.spine_db_editor.graphics_items), 397

ObjectItem (class in spinetool-

`box.spine_db_editor.mvcmodels.entity_tree_item`), `ONLINE_DOCUMENTATION_URL` (in module `spinetoolbox.config`), 504

`ObjectLabelItem` (class in `spinetoolbox.spine_db_editor.graphics_items`), 401

`ObjectNameDelegate` (class in `spinetoolbox.spine_db_editor.widgets.custom_delegates`), 327

`ObjectNameListDelegate` (class in `spinetoolbox.spine_db_editor.widgets.custom_delegates`), 328

`ObjectNameListEditor` (class in `spinetoolbox.spine_db_editor.widgets.object_name_list_editor`), 370

`ObjectParameterDefinitionTableView` (class in `spinetoolbox.spine_db_editor.widgets.custom_qtableview`), 341

`ObjectParameterTableMixin` (class in `spinetoolbox.spine_db_editor.widgets.custom_qtableview`), 341

`ObjectParameterValueTableView` (class in `spinetoolbox.spine_db_editor.widgets.custom_qtableview`), 342

`ObjectRelationshipClassItem` (class in `spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item`), 251

`ObjectTreeModel` (class in `spinetoolbox.spine_db_editor.mvcmodels.entity_tree_model`), 254

`ObjectTreeRootItem` (class in `spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item`), 248

`ObjectTreeView` (class in `spinetoolbox.spine_db_editor.widgets.custom_qtreeview`), 350

`OK` (`spinetoolbox.headless.Status` attribute), 516

`OK` (`spinetoolbox.project.ItemNameStatus` attribute), 557

`okPressed` (`spinetoolbox.widgets.custom_qwidgets.FilterWidget` attribute), 437

`on_process_error()` (`spinetoolbox.execution_managers.QProcessExecutionManager` method), 506

`on_process_finished()` (`spinetoolbox.execution_managers.QProcessExecutionManager` method), 506

`on_ready_stderr()` (`spinetoolbox.execution_managers.QProcessExecutionManager` method), 506

`on_ready_stdout()` (`spinetoolbox.execution_managers.QProcessExecutionManager` method), 506

`on_state_changed()` (`spinetoolbox.execution_managers.QProcessExecutionManager` method), 506

`online_filters()` (`spinetoolbox.project_item.logging_connection.LoggingConnection` method), 218

`opacity` (`spinetoolbox.widgets.notification.Notification` attribute), 469

`open_anchor()` (`spinetoolbox.ui_main.ToolboxUI` method), 632

`open_containing_folder()` (`spinetoolbox.spine_db_editor.widgets.custom_qwidgets.OpenFileButton` method), 354

`open_db_editor()` (`spinetoolbox.spine_db_manager.SpineDBManager` method), 610

`open_db_file()` (`spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase` method), 379

`open_directory()` (`spinetoolbox.project_item.project_item.ProjectItem` method), 224

`open_file()` (`spinetoolbox.spine_db_editor.widgets.custom_qwidgets.OpenFileButton` method), 354

`open_file()` (`spinetoolbox.spine_db_editor.widgets.custom_qwidgets.OpenSQLiteFileButton` method), 355

`open_in_editor()` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.ParameterTableView` method), 340

`open_in_editor()` (`spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView` method), 344

`open_in_editor()` (`spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ParameterValueListWidget` method), 354

`open_project()` (in module `spinetoolbox.headless`), 515

`open_project()` (`spinetoolbox.ui_main.ToolboxUI` method), 629

`open_project()` (`spinetoolbox.widgets.open_project_widget.OpenProjectDialog` method), 472

`open_rsc_dir()` (`spinetoolbox.widgets.settings_widget.SettingsWidget` method), 496

`open_specification_file()` (`spinetoolbox.ui_main.ToolboxUI` method), 633

`open_url()` (in module `spinetoolbox.helpers`), 520

`open_value_editor()` (`spinetoolbox.widgets.array_editor.ArrayEditor` method), 409

`open_value_editor()` (`spinetoolbox.widgets.map_editor.MapEditor` method), 461

[OpenFileButton](#) (class in `spinetoolbox.spine_db_editor.widgets.custom_qwidgets`), 354
[OpenProjectDialog](#) (class in `spinetoolbox.widgets.open_project_widget`), 471
[OpenProjectDialogComboBox](#) (class in `spinetoolbox.widgets.custom_combobox`), 411
[OpenProjectDialogComboBoxContextMenu](#) (class in `spinetoolbox.widgets.custom_menus`), 418
[OpenSQLiteFileButton](#) (class in `spinetoolbox.spine_db_editor.widgets.custom_qwidgets`), 355
[original_db_map_ids](#) (`spinetoolbox.spine_db_editor.graphics_items.EntityItem` property), 395
[original_xy_data](#) (`spinetoolbox.widgets.plot_widget.PlotWidget` attribute), 483
[other_item\(\)](#) (`spinetoolbox.spine_db_editor.graphics_items.ArcItem` method), 399
[others\(\)](#) (`spinetoolbox.widgets.multi_tab_window.MultiTabWindow` method), 358
[outgoing_connection_links\(\)](#) (`spinetoolbox.project_item_icon.ProjectItemIcon` method), 578
[outgoing_connections\(\)](#) (`spinetoolbox.project.SpineToolboxProject` method), 568
[outgoing_links\(\)](#) (`spinetoolbox.project_item_icon.ConnectorButton` method), 580
[outline_color](#) (`spinetoolbox.link.LinkBase` property), 536
[override_console_and_execution_list\(\)](#) (`spinetoolbox.ui_main.ToolboxUI` method), 634
[OVERWRITE](#) (`spinetoolbox.spine_db_editor.widgets.scenario_generator.ScenarioGenerator` attribute), 375
[overwrite_check\(\)](#) (`spinetoolbox.ui_main.ToolboxUI` method), 631
[owner_names](#) (`spinetoolbox.widgets.jupyter_console_widget.JupyterConsoleWidget` property), 453
[owner_names](#) (`spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget` property), 477
P
[PackItem](#) (`spinetoolbox.mvcmodels.file_list_models.FileListModel` attribute), 182
[PaddingLabel](#) (class in `spinetoolbox.widgets.toolbars`), 503
[paint\(\)](#) (`spinetoolbox.helpers.CharIconEngine` method), 523
[paint\(\)](#) (`spinetoolbox.link.JumpOrLink` method), 538
[paint\(\)](#) (`spinetoolbox.project_item_icon.ProjectItemIcon` method), 580
[paint\(\)](#) (`spinetoolbox.spine_db_editor.graphics_items.EntityItem` method), 395
[paint\(\)](#) (`spinetoolbox.spine_db_editor.widgets.custom_delegates.ManageEntityItem` method), 332
[paint\(\)](#) (`spinetoolbox.spine_db_editor.widgets.custom_delegates.RelationItem` method), 323
[paint\(\)](#) (`spinetoolbox.spine_db_editor.widgets.custom_delegates.ScenarioItem` method), 323
[paint\(\)](#) (`spinetoolbox.spine_db_icon_manager.SceneIconEngine` method), 595
[paint\(\)](#) (`spinetoolbox.widgets.custom_delegates.CheckBoxDelegate` method), 412
[paint\(\)](#) (`spinetoolbox.widgets.custom_delegates.ComboBoxDelegate` method), 412
[paint\(\)](#) (`spinetoolbox.widgets.custom_editors._IconPainterDelegate` method), 416
[paintEvent\(\)](#) (`spinetoolbox.spine_db_editor.widgets.graph_layout_generator.ProgressBar` method), 410
[paintEvent\(\)](#) (`spinetoolbox.widgets.code_text_edit.LineNumberArea` method), 410
[paintEvent\(\)](#) (`spinetoolbox.widgets.custom_combobox.ElidedCombobox` method), 411
[paintEvent\(\)](#) (`spinetoolbox.widgets.custom_qwidgets._MenuToolBar` method), 439
[paintEvent\(\)](#) (`spinetoolbox.widgets.custom_qwidgets.MenuItemToolBarWidget` method), 439
[paintEvent\(\)](#) (`spinetoolbox.widgets.project_item_drag.ProjectItemSpecArray` method), 487
[paintEvent\(\)](#) (`spinetoolbox.widgets.project_item_drag.ShadeMixin` method), 487
[paintEvent\(\)](#) (`spinetoolbox.widgets.properties_widget.PropertiesWidgetBase` method), 489
[paintEvent\(\)](#) (`spinetoolbox.widgets.toolbars.MainToolBar` method), 503
[parameter_definition_id_key](#) (`spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel` property), 239
[parameter_definition_id_key](#) (`spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel` property), 298
[parameter_identifier\(\)](#) (in module `spinetoolbox.helpers`), 530

<code>parameter_value_editor_requested</code> (spinetool- box.spine_db_editor.widgets.custom_delegates.ParameterPi attribute), 324	<code>ParameterValueOrDefaultValueDelegate</code> (class in spinetool- box.spine_db_editor.widgets.custom_delegates), 325
<code>parameter_value_editor_requested</code> (spinetool- box.spine_db_editor.widgets.custom_delegates.ParameterValueDelegate), 331	<code>ParameterValueDelegateHeaderView</code> (class in spinetool- box.spine_db_editor.widgets.pivot_table_header_view), 373
<code>parameter_value_editor_requested</code> (spinetool- box.spine_db_editor.widgets.custom_delegates.ParameterValueDelegate), 325	<code>ParameterValueDelegateHeaderView</code> (class in spinetool- box.spine_db_editor.mvcmodels.pivot_table_models), 289
<code>ParameterDefaultValueDelegate</code> (class in spinetool- box.spine_db_editor.widgets.custom_delegates), 326	<code>ParameterValueTableView</code> (class in spinetool- box.spine_db_editor.widgets.custom_qtableview), 341
<code>ParameterDefinitionTableView</code> (class in spinetool- box.spine_db_editor.widgets.custom_qtableview), 341	<code>ParameterViewFilterMenu</code> (class in spinetool- box.spine_db_editor.widgets.custom_menus), 334
<code>ParameterDelegate</code> (class in spinetool- box.spine_db_editor.widgets.custom_delegates), 324	<code>ParameterViewMixin</code> (class in spinetool- box.spine_db_editor.widgets.parameter_view_mixin), 371
<code>ParameterNameDelegate</code> (class in spinetool- box.spine_db_editor.widgets.custom_delegates), 327	<code>parent</code> (spinetoolbox.project_item_icon.ConnectorButton property), 580
<code>ParameterNameDelegate</code> (class in spinetool- box.spine_db_editor.widgets.select_position_parameters_dialog), 377	<code>parent</code> (spinetoolbox.widgets.custom_qgraphicsviews.CustomQGraphicsView attribute), 423
<code>ParameterPivotTableDelegate</code> (class in spinetool- box.spine_db_editor.widgets.custom_delegates), 324	<code>parent</code> (spinetoolbox.widgets.datetime_editor.DatetimeEditor attribute), 443
<code>ParameterTableHeaderSection</code> (class in spinetool- box.plotting), 547	<code>parent</code> (spinetoolbox.widgets.duration_editor.DurationEditor attribute), 444
<code>ParameterTableView</code> (class in spinetool- box.spine_db_editor.widgets.custom_qtableview), 339	<code>parent</code> (spinetoolbox.widgets.map_editor.MapEditor at- tribute), 461
<code>ParameterValueDelegate</code> (class in spinetool- box.spine_db_editor.widgets.custom_delegates), 326	<code>parent()</code> (spinetoolbox.mvcmodels.file_list_models.FileListModel method), 183
<code>ParameterValueEditor</code> (class in spinetool- box.widgets.parameter_value_editor), 474	<code>parent()</code> (spinetoolbox.mvcmodels.minimal_tree_model.MinimalTreeModel method), 198
<code>ParameterValueEditorBase</code> (class in spinetool- box.widgets.parameter_value_editor_base), 475	<code>parent()</code> (spinetoolbox.mvcmodels.project_item_model.ProjectItemModel method), 200
<code>ParameterValueElementDelegate</code> (class in spinetool- box.spine_db_editor.widgets.custom_delegates), 324	<code>parent()</code> (spinetoolbox.mvcmodels.project_tree_item.BaseProjectTreeItem method), 206
<code>ParameterValueLineEditor</code> (class in spinetool- box.widgets.custom_editors), 414	<code>parent_item</code> (spinetool- box.mvcmodels.minimal_tree_model.TreeItem property), 196
<code>ParameterValueListDelegate</code> (class in spinetool- box.spine_db_editor.widgets.custom_delegates), 331	<code>parent_name()</code> (spinetool- box.project_item_icon.ConnectorButton method), 580
<code>ParameterValueListModel</code> (class in spinetool- box.spine_db_editor.mvcmodels.parameter_value_list_model), 278	<code>parse_item_dict()</code> (spinetool- box.project_item.project_item.ProjectItem static method), 223
<code>ParameterValueListTreeView</code> (class in spinetool- box.spine_db_editor.widgets.custom_qtreeview), 354	<code>parse_project_item_modules()</code> (spinetool- box.ui_main.ToolboxUI method), 628
	<code>parse_specification_file()</code> (in module spinetool- box.helpers), 528
	<code>parse_text()</code> (spinetool- box.widgets.persistent_console_widget.AnsiEscapeCodeHandler method), 480
	<code>PARSED_ROLE</code> (in module spinetool-

744 Index

- method*), 344
- `plot_data()` (in module *spinetoolbox.plotting*), 549
- `plot_db_mgr_items()` (in module *spinetoolbox.plotting*), 553
- `plot_in_window()` (*spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView* *method*), 340
- `plot_parameter_table_selection()` (in module *spinetoolbox.plotting*), 552
- `plot_pivot_table_selection()` (in module *spinetoolbox.plotting*), 553
- `plot_value_editor_table_selection()` (in module *spinetoolbox.plotting*), 552
- `plot_windows` (*spinetoolbox.widgets.plot_widget.PlotWidget* attribute), 483
- `plot_x_column` (*spinetoolbox.spine_db_editor.mvcmodels.pivot_table_model.PivotTableModel* property), 285
- `PlotCanvas` (class in *spinetoolbox.widgets.plot_canvas*), 482
- `PlottingError`, 547
- `PlotType` (class in *spinetoolbox.plotting*), 546
- `PlotWidget` (class in *spinetoolbox.widgets.plot_widget*), 483
- `plugin_path` (in module *spinetoolbox.main*), 544
- `PLUGIN_REGISTRY_URL` (in module *spinetoolbox.config*), 504
- `plugin_specs` (*spinetoolbox.plugin_manager.PluginManager* property), 555
- `plugin_toolbars` (*spinetoolbox.plugin_manager.PluginManager* property), 555
- `PluginManager` (class in *spinetoolbox.plugin_manager*), 555
- `plugins_dirs()` (in module *spinetoolbox.helpers*), 529
- `PLUGINS_PATH` (in module *spinetoolbox.config*), 504
- `PluginToolBar` (class in *spinetoolbox.widgets.toolbars*), 501
- `PluginWorkFailed`, 556
- `plus_clicked` (*spinetoolbox.widgets.multi_tab_window.TabBarPlus* attribute), 468
- `populate_context_menu()` (*spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityTableView* *method*), 336
- `populate_context_menu()` (*spinetoolbox.spine_db_editor.widgets.custom_qtableview.ParameterTableModel* *method*), 340
- `populate_context_menu()` (*spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView* *method*), 343
- `populate_context_menu()` (*spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView* *method*), 344
- `populate_context_menu()` (*spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView* *method*), 344
- `populate_context_menu()` (*spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView* *method*), 345
- `populate_context_menu()` (*spinetoolbox.spine_db_editor.widgets.custom_qtableview.PivotTableView* *method*), 345
- `populate_context_menu()` (*spinetoolbox.spine_db_editor.widgets.custom_qtreeview.AlternativeTreeView* *method*), 352
- `populate_context_menu()` (*spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ItemTreeView* *method*), 351
- `populate_context_menu()` (*spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ParameterValueListModel* *method*), 354
- `populate_context_menu()` (*spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ScenarioTreeView* *method*), 353
- `populate_list()` (*spinetoolbox.widgets.plugin_manager_widgets.InstallPluginDialog* *method*), 485
- `populate_list()` (*spinetoolbox.widgets.plugin_manager_widgets.ManagePluginsDialog* *method*), 485
- `populate_pivot_action_group()` (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* *method*), 385
- `populate_table_view()` (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddReadyRelationDialog* *method*), 315
- `position()` (*spinetoolbox.fetch_parent.FetchParent* *method*), 507
- `predecessor_names()` (*spinetoolbox.project.SpineToolboxProject* *method*), 568
- `preferred_row_height()` (in module *spinetoolbox.helpers*), 531
- `prepare_plot_in_window_menu()` (in module *spinetoolbox.widgets.plot_widget*), 484
- `prepare_remote_execution()` (*spinetoolbox.project.SpineToolboxProject* *method*), 570
- `previous_sibling()` (*spinetoolbox.mvcmodels.minimal_tree_model.TreeItem* *method*), 196
- `private_name` (*spinetoolbox.widgets.qwidgets.QWizardProcessPage._ExecutionModel* attribute), 440
- `process_started()` (*spinetoolbox.widgets.qwidgets.QWizardProcessPage._ExecutionModel* attribute), 440

`box.execution_managers.QProcessExecutionManager` (class in `spinetoolbox.execution_managers`), 506

`program()` (`spinetoolbox.execution_managers.QProcessExecutionManager` method), 505

`ProgressBarWidget` (class in `spinetoolbox.spine_db_editor.widgets.graph_layout_generator`), 358

`progressed` (`spinetoolbox.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGenerator` attribute), 358

`project()` (`spinetoolbox.ui_main.ToolboxUI` method), 628

`project_about_to_be_torn_down` (`spinetoolbox.project.SpineToolboxProject` attribute), 558

`project_dir` (`spinetoolbox.headless.ModifiableProject` property), 513

`project_execution_about_to_start` (`spinetoolbox.project.SpineToolboxProject` attribute), 558

`project_execution_finished` (`spinetoolbox.project.SpineToolboxProject` attribute), 558

`PROJECT_FILENAME` (in module `spinetoolbox.config`), 504

`project_item` (`spinetoolbox.mvcmodels.project_tree_item.LeafProjectTreeItem` property), 207

`project_item()` (`spinetoolbox.project_item_icon.ConnectorButton` method), 580

`project_item_context_menu()` (`spinetoolbox.ui_main.ToolboxUI` method), 639

`project_item_from_clipboard()` (`spinetoolbox.ui_main.ToolboxUI` method), 637

`project_item_icon()` (`spinetoolbox.ui_main.ToolboxUI` method), 638

`project_item_icons()` (`spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene` method), 422

`project_item_properties_ui()` (`spinetoolbox.ui_main.ToolboxUI` method), 638

`project_item_to_clipboard()` (`spinetoolbox.ui_main.ToolboxUI` method), 637

`PROJECT_LOCAL_DATA_DIR_NAME` (in module `spinetoolbox.config`), 504

`PROJECT_LOCAL_DATA_FILENAME` (in module `spinetoolbox.config`), 504

`PROJECT_ZIP_FILENAME` (in module `spinetoolbox.config`), 504

`ProjectDirectoryIconProvider` (class in `spinetoolbox.helpers`), 524

`ProjectItem` (class in `spinetoolbox.project_item.project_item`), 220

`ProjectItemButton` (class in `spinetoolbox.widgets.project_item_drag`), 486

`ProjectItemButtonBase` (class in `spinetoolbox.widgets.project_item_drag`), 486

`ProjectItemDragMixin` (class in `spinetoolbox.widgets.project_item_drag`), 486

`ProjectItemFactory` (class in `spinetoolbox.project_item.project_item_factory`), 226

`ProjectItemIcon` (class in `spinetoolbox.project_item.project_item_factory`), 226

`ProjectItemModel` (class in `spinetoolbox.mvcmodels.project_item_model`), 199

`ProjectItemSpecArray` (class in `spinetoolbox.widgets.project_item_drag`), 487

`ProjectItemSpecButton` (class in `spinetoolbox.widgets.project_item_drag`), 487

`ProjectItemSpecificationModel` (class in `spinetoolbox.mvcmodels.project_item_specification_models`), 203

`ProjectSettings` (class in `spinetoolbox.project_settings`), 583

`ProjectUpgrader` (class in `spinetoolbox.project_upgrader`), 584

`prompt` (`spinetoolbox.widgets.persistent_console_widget.PersistentConsole` property), 477

`prompt_exit_without_saving()` (`spinetoolbox.project_item.specification_editor_window.SpecificationEditor` method), 230

`prompt_save_location()` (`spinetoolbox.ui_main.ToolboxUI` method), 632

`prompt_to_save_changes()` (in module `spinetoolbox.project_item.specification_editor_window`), 231

`PropertiesWidgetBase` (class in `spinetoolbox.widgets.properties_widget`), 489

`PropertyQLineEdit` (class in `spinetoolbox.widgets.custom_qlineedit`), 426

`PropertyQSpinBox` (class in `spinetoolbox.widgets.custom_qwidgets`), 441

`prune_selected_items()` (`spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView` method), 337

`public_name` (`spinetoolbox.widgets.custom_qwidgets.QWizardProcessPage._ExecutionManager` attribute), 440

`purge_items()` (`spinetoolbox.spine_db_manager.SpineDBManager` method), 609

`PurgeSettingsDialog` (class in `spinetoolbox.widgets.custom_qwidgets`), 442

`push()` (`spinetoolbox.spine_db_commands.AgedUndoStack` method), 590

`push_alternative_ids()` (`spinetoolbox.spine_db_parcel.SpineDBParcel` method), 611

- `push_feature_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 612
- `push_object_class_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 611
- `push_object_group_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 611
- `push_object_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 611
- `push_parameter_definition_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 611
- `push_parameter_value_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 611
- `push_parameter_value_list_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 611
- `push_relationship_class_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 611
- `push_relationship_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 611
- `push_scenario_alternative_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 612
- `push_scenario_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 611
- `push_tool_feature_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 612
- `push_tool_feature_method_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 612
- `push_tool_ids()` (*spinetoolbox.spine_db_parcel.SpineDBParcel* method), 612
- `put()` (*spinetoolbox.qlthread_pool_executor.QtBasedQueue* method), 589
- `pySide6_version_check()` (in module *spinetoolbox.helpers*), 521
- ## Q
- `QProcessExecutionManager` (class in *spinetoolbox.execution_managers*), 505
- `qsettings` (*spinetoolbox.widgets.settings_widget.SettingsWidgetBase* property), 493
- `qsettings()` (*spinetoolbox.ui_main.ToolboxUI* method), 628
- `QtBasedFuture` (class in *spinetoolbox.qlthread_pool_executor*), 589
- `QtBasedQueue` (class in *spinetoolbox.qlthread_pool_executor*), 589
- `QtBasedThread` (class in *spinetoolbox.qlthread_pool_executor*), 589
- `QtBasedThreadPoolExecutor` (class in *spinetoolbox.qlthread_pool_executor*), 590
- `query()` (*spinetoolbox.spine_db_manager.SpineDBManager* method), 600
- `query()` (*spinetoolbox.spine_db_worker.SpineDBWorker* method), 614
- `QuietLogger` (class in *spinetoolbox.helpers*), 527
- `QWizardProcessPage` (class in *spinetoolbox.widgets.custom_qwidgets*), 440
- `QWizardProcessPage._ExecutionManager` (class in *spinetoolbox.widgets.custom_qwidgets*), 440
- ## R
- `raise_if_incompatible_x()` (in module *spinetoolbox.plotting*), 548
- `raise_if_not_common_x_labels()` (in module *spinetoolbox.plotting*), 548
- `RankDelegate` (class in *spinetoolbox.widgets.custom_delegates*), 413
- `RankIcon` (class in *spinetoolbox.project_item_icon*), 582
- `raw_text()` (*spinetoolbox.widgets.persistent_console_widget._CustomLineEdit* method), 476
- `rcv_next()` (*spinetoolbox.server.engine_client.EngineClient* method), 232
- `read_settings()` (*spinetoolbox.widgets.settings_widget.SettingsWidget* method), 496
- `read_settings()` (*spinetoolbox.widgets.settings_widget.SpineDBEditorSettingsMixin* method), 494
- `reattach()` (*spinetoolbox.widgets.multi_tab_window.MultiTabWindow* method), 467
- `rebuild_graph()` (*spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin* method), 360
- `receive_error_msg()` (*spinetoolbox.project_item.logging_connection.LoggingConnection* method), 217
- `receive_error_msg()` (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase* method), 381
- `receive_error_msg()` (*spinetoolbox.spine_db_manager.SpineDBManager* method), 597

`receive_resources_from_source()` (*spinetoolbox.project_item.logging_connection.HeadlessConnection* method), 216
`receive_resources_from_source()` (*spinetoolbox.project_item.logging_connection.LoggingConnection* method), 218
`receive_session_committed()` (*spinetoolbox.project_item.logging_connection.LoggingConnection* method), 217
`receive_session_committed()` (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor* method), 380
`receive_session_committed()` (*spinetoolbox.spine_db_manager.SpineDBManager* method), 597
`receive_session_refreshed()` (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor* method), 380
`receive_session_refreshed()` (*spinetoolbox.spine_db_manager.SpineDBManager* method), 597
`receive_session_rolled_back()` (*spinetoolbox.project_item.logging_connection.LoggingConnection* method), 217
`receive_session_rolled_back()` (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor* method), 383
`receive_session_rolled_back()` (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor* method), 380
`receive_session_rolled_back()` (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor* method), 380
`receive_session_rolled_back()` (*spinetoolbox.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin* method), 390
`receive_session_rolled_back()` (*spinetoolbox.spine_db_manager.SpineDBManager* method), 597
`receive_text_changed()` (*spinetoolbox.widgets.commit_dialog.CommitDialog* method), 411
`RecentProjectsPopupMenu` (class in *spinetoolbox.widgets.custom_menus*), 419
`rect()` (*spinetoolbox.project_item_icon.ConnectorButton* method), 580
`rect()` (*spinetoolbox.project_item_icon.ProjectItemIcon* method), 578
`recursive_overwrite()` (in module *spinetoolbox.helpers*), 522
`redo()` (*spinetoolbox.project_commands.AddConnectionCommand* method), 574
`redo()` (*spinetoolbox.project_commands.AddJumpCommand* method), 574
`redo()` (*spinetoolbox.project_commands.AddProjectItemsCommand* method), 572
`redo()` (*spinetoolbox.project_commands.AddSpecificationCommand* method), 572
`redo()` (*spinetoolbox.project_commands.MoveIconCommand* method), 572
`redo()` (*spinetoolbox.project_commands.RemoveAllProjectItemsCommand* method), 573
`redo()` (*spinetoolbox.project_commands.RemoveConnectionsCommand* method), 574
`redo()` (*spinetoolbox.project_commands.RemoveJumpsCommand* method), 574
`redo()` (*spinetoolbox.project_commands.RemoveProjectItemsCommand* method), 573
`redo()` (*spinetoolbox.project_commands.RemoveSpecificationCommand* method), 577
`redo()` (*spinetoolbox.project_commands.RenameProjectItemCommand* method), 573
`redo()` (*spinetoolbox.project_commands.ReplaceSpecificationCommand* method), 576
`redo()` (*spinetoolbox.project_commands.SaveSpecificationAsCommand* method), 577
`redo()` (*spinetoolbox.project_commands.SetConnectionDefaultFilterOnlineCommand* method), 576
`redo()` (*spinetoolbox.project_commands.SetConnectionOptionsCommand* method), 576
`redo()` (*spinetoolbox.project_commands.SetFiltersOnlineCommand* method), 575
`redo()` (*spinetoolbox.project_commands.SetItemSpecificationCommand* method), 572
`redo()` (*spinetoolbox.project_commands.SetJumpConditionCommand* method), 575
`redo()` (*spinetoolbox.project_commands.SetProjectDescriptionCommand* method), 572
`redo()` (*spinetoolbox.project_commands.UpdateJumpCmdLineArgsCommand* method), 575
`redo()` (*spinetoolbox.project_item.specification_editor_window.ChangeSpecificationEditorWindow* method), 229
`redo()` (*spinetoolbox.spine_db_commands.AddItemCommand* method), 593
`redo()` (*spinetoolbox.spine_db_commands.AgedUndoCommand* method), 591
`redo()` (*spinetoolbox.spine_db_commands.RemoveItemsCommand* method), 594
`redo()` (*spinetoolbox.spine_db_commands.SpineDBMacro* method), 591
`redo()` (*spinetoolbox.spine_db_commands.UpdateItemsCommand* method), 593
`redo_age` (*spinetoolbox.spine_db_commands.AgedUndoStack* property), 590
`redo_indexes()` (in module *spinetoolbox.plotting*), 548
`redo_fit()` (*spinetoolbox.widgets.custom_editors.SearchBarEditor* method), 415
`redo_refresh()` (*spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel* method), 178
`redo_refresh_active_elements()` (*spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel* method), 178

[box.ui_main.ToolboxUI method](#)), 631
[refresh_copy_paste_actions\(\)](#) ([spinetool-box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase](#) method), 379
[refresh_edit_action_states\(\)](#) ([spinetool-box.ui_main.ToolboxUI method](#)), 636
[refresh_icon\(\)](#) ([spinetool-box.spine_db_editor.graphics_items.CrossHairsItem](#) method), 399
[refresh_icon\(\)](#) ([spinetool-box.spine_db_editor.graphics_items.CrossHairsRelationshipItem](#) method), 400
[refresh_icon\(\)](#) ([spinetool-box.spine_db_editor.graphics_items.EntityItem](#) method), 395
[refresh_resource_filter_model\(\)](#) ([spinetool-box.project_item.logging_connection.LoggingConnection](#) method), 218
[refresh_session\(\)](#) ([spinetool-box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase](#) method), 380
[refresh_session\(\)](#) ([spinetool-box.spine_db_manager.SpineDBManager](#) method), 601
[refresh_toolbars\(\)](#) ([spinetool-box.ui_main.ToolboxUI method](#)), 629
[refresh_views\(\)](#) ([spinetool-box.spine_db_editor.widgets.tabular_view_mixin.TabularViewMixin](#) method), 385
[refreshed](#) ([spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel](#) attribute), 177
[register_anchor_callback\(\)](#) ([spinetool-box.ui_main.ToolboxUI method](#)), 632
[register_listener\(\)](#) ([spinetool-box.spine_db_manager.SpineDBManager](#) method), 600
[relationship_class_renderer\(\)](#) ([spinetool-box.spine_db_icon_manager.SpineDBIconManager](#) method), 595
[relationship_selection_changed](#) ([spinetool-box.spine_db_editor.widgets.custom_qtreeview.EntityTreeView](#) attribute), 348
[RelationshipClassItem](#) (class in [spinetool-box.spine_db_editor.mvcmodels.entity_tree_item](#)), 250
[RelationshipClassNameDelegate](#) (class in [spinetool-box.spine_db_editor.widgets.custom_delegates](#)), 327
[RelationshipItem](#) (class in [spinetool-box.spine_db_editor.graphics_items](#)), 397
[RelationshipItem](#) (class in [spinetool-box.spine_db_editor.mvcmodels.entity_tree_item](#)), 253
[RelationshipParameterDefinitionTableView](#) (class in [spinetool-box.spine_db_editor.widgets.custom_qtableview](#)), 341
[RelationshipParameterTableMixin](#) (class in [spinetool-box.spine_db_editor.widgets.custom_qtableview](#)), 341
[RelationshipParameterValueTableView](#) (class in [spinetool-box.spine_db_editor.widgets.custom_qtableview](#)), 342
[RelationshipPivotTableDelegate](#) (class in [spinetool-box.spine_db_editor.widgets.custom_delegates](#)), 322
[RelationshipPivotTableModel](#) (class in [spinetool-box.spine_db_editor.mvcmodels.pivot_table_models](#)), 291
[RelationshipTreeModel](#) (class in [spinetool-box.spine_db_editor.mvcmodels.entity_tree_models](#)), 254
[RelationshipTreeRootItem](#) (class in [spinetool-box.spine_db_editor.mvcmodels.entity_tree_item](#)), 249
[RelationshipTreeView](#) (class in [spinetool-box.spine_db_editor.widgets.custom_qtreeview](#)), 350
[release_level\(\)](#) ([spinetool-box.widgets.jupyter_console_widget.JupyterConsoleWidget](#) method), 385
[reload_frozen_table\(\)](#) ([spinetool-box.plugin_manager.PluginManager](#) method), 556
[reload_plugins_with_local_data\(\)](#) ([spinetool-box.plugin_manager.PluginManager](#) method), 556
[remaining_time\(\)](#) ([spinetool-box.widgets.notification.Notification](#) method), 470
[RemoteSpineEngineManager](#) (class in [spinetool-box.spine_engine_manager](#)), 622
[remove_all_items\(\)](#) ([spinetool-box.ui_main.ToolboxUI method](#)), 632
[remove_alternatives\(\)](#) ([spinetool-box.spine_db_editor.widgets.custom_qtableview.PivotTableView](#) method), 343
[remove_child\(\)](#) ([spinetool-box.mvcmodels.project_tree_item.BaseProjectTreeItem](#) method), 206
[remove_children\(\)](#) ([spinetool-box.mvcmodels.minimal_tree_model.TreeItem](#) method), 197

<code>remove_children()</code>	(spinetool- box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem method), 269	box.spine_db_manager.SpineDBManager remove_items() (spinetool- box.spine_db_worker.SpineDBWorker method), box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem method), 268
<code>remove_children_by_id()</code>	(spinetool- box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem method), 268	remove_items_from_filter_list() (spinetool- box.widgets.custom_menus.FilterMenuBase method), 420
<code>remove_column()</code>	(spinetool- box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel method), 256	remove_items_in_cache() (spinetool- box.spine_db_manager.SpineDBManager method), 317
<code>remove_column()</code>	(spinetool- box.spine_db_editor.widgets.add_items_dialogs.AddRelationshipDialog method), 317	remove_jump() (spinetool- box.project.SpineToolboxProject method), 565
<code>remove_connection()</code>	(spinetool- box.project.SpineToolboxProject method), 564	remove_leaves() (spinetool- box.mvcmodels.project_item_model.ProjectItemModel method), 203
<code>remove_db_map_ids()</code>	(spinetool- box.spine_db_editor.graphics_items.EntityItem method), 396	remove_links() (spinetool- box.widgets.custom_qgraphicsviews.DesignQGraphicsView method), 425
<code>remove_db_map_listener()</code>	(spinetool- box.spine_db_manager.SpineDBManager method), 600	remove_members() (spinetool- box.spine_db_editor.widgets.add_items_dialogs.ObjectGroupDialog method), 319
<code>remove_directory_from_recents()</code>	(spinetool- box.widgets.open_project_widget.OpenProjectDialog static method), 473	remove_metadata() (spinetool- box.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel method), 261
<code>remove_entity_tree_items()</code>	(spinetool- box.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin method), 391	remove_notification() (spinetool- box.project_item.project_item.ProjectItem method), 185
<code>remove_filter()</code>	(spinetool- box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel method), 185	remove_notification() (spinetool- box.project_item_icon.ExclamationIcon method), 582
<code>remove_from_model()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_model.PivotModel method), 280	remove_objects() (spinetool- box.spine_db_editor.widgets.custom_qtableview.PivotTableView. method), 286
<code>remove_from_model()</code>	(spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModelBase method), 286	remove_parameters() (spinetool- box.spine_db_editor.widgets.custom_qtableview.PivotTableView. method), 425
<code>remove_icon()</code>	(spinetool- box.widgets.custom_qgraphicsviews.DesignQGraphicsView method), 425	remove_path_from_recent_projects() (spinetool- box.ui_main.ToolboxUI method), 636
<code>remove_item()</code>	(spinetoolbox.fetch_parent.FetchParent method), 508	remove_project_from_server() (spinetool- box.server.engine_client.EngineClient method), 234
<code>remove_item()</code>	(spinetool- box.mvcmodels.project_item_model.ProjectItemModel method), 202	remove_relationships() (spinetool- box.spine_db_editor.widgets.custom_qtableview.PivotTableView. method), 344
<code>remove_item_by_name()</code>	(spinetool- box.project.SpineToolboxProject method), 566	remove_scenarios() (spinetool- box.spine_db_editor.widgets.custom_qtableview.PivotTableView. method), 259
<code>remove_item_metadata()</code>	(spinetool- box.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel method), 259	remove_selected() (spinetool- box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 185
<code>remove_items()</code>	(spinetool- box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel method), 185	remove_selected() (spinetool- box.spine_db_editor.widgets.custom_qtableview.ParameterTableView. method), 185
<code>remove_items()</code>	(spinetool- box.spine_db_editor.widgets.custom_qtableview.ParameterTableView. method), 185	

method), 340

remove_selected() (spinetool- RemoveItemsCommand (class in spinetool-
box.spine_db_editor.widgets.custom_qtreeview.AlternativeTreeView.spine_db_commands), 593
method), 352 RemoveJumpsCommand (class in spinetool-
remove_selected() (spinetool- box.project_commands), 574
box.spine_db_editor.widgets.custom_qtreeview.EnhancedProjectItemsCommand (class in spinetool-
method), 349 box.project_commands), 573
remove_selected() (spinetool- removeRow() (spinetool-
box.spine_db_editor.widgets.custom_qtreeview.ItemTreeView box.mvcmodels.project_item_specification_models.ProjectItemSp
method), 351 method), 204
remove_selected() (spinetool- removeRows() (spinetool-
box.spine_db_editor.widgets.custom_qtreeview.ParameterValueListModel box.mvcmodels.array_model.ArrayModel
method), 354 method), 176
remove_selected() (spinetool- removeRows() (spinetool-
box.spine_db_editor.widgets.custom_qtreeview.ScenarioTreeView box.mvcmodels.compound_table_model.CompoundTableModel
method), 353 method), 179
remove_selected() (spinetool- removeRows() (spinetool-
box.spine_db_editor.widgets.custom_qtreeview.ToolFeatureTreeView box.mvcmodels.empty_row_model.EmptyRowModel
method), 351 method), 181
remove_selected_links() (spinetool- removeRows() (spinetool-
box.widgets.custom_qgraphicsviews.DesignQGraphicsView box.mvcmodels.map_model.MapModel
method), 426 method), 191
remove_selected_rows() (spinetool- removeRows() (spinetool-
box.spine_db_editor.widgets.add_items_dialogs.AddItemDialog box.mvcmodels.minimal_table_model.MinimalTableModel
method), 315 method), 195
remove_specification() (spinetool- removeRows() (spinetool-
box.mvcmodels.project_item_specification_models.ProjectItemSpecificationModel box.mvcmodels.time_pattern_model.TimePatternModel
method), 203 method), 210
remove_specification() (spinetool- removeRows() (spinetool-
box.project.SpineToolboxProject method), box.mvcmodels.time_series_model_fixed_resolution.TimeSeriesM
561 method), 212
remove_specification() (spinetool- removeRows() (spinetool-
box.ui_main.ToolboxUI method), 633 box.mvcmodels.time_series_model_variable_resolution.TimeSeries
remove_values() (spinetool- method), 214
box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel (spinetool-
method), 255 box.spine_db_editor.mvcmodels.metadata_table_model_base.Met
remove_values() (spinetool- method), 264
box.spine_db_editor.widgets.custom_qtableview.ProjectSpecificationPartitionCommand (class in spinetool-
method), 344 box.project_commands), 576
RemoveAllProjectItemsCommand (class in spinetool- rename() (spinetoolbox.project_item.project_item.ProjectItem
box.project_commands), 572 method), 224
removeColumns() (spinetool- rename_dir() (in module spinetoolbox.helpers), 520
box.mvcmodels.map_model.MapModel rename_item() (spinetool-
method), 191 box.project.SpineToolboxProject method),
removeColumns() (spinetool- 563
box.mvcmodels.minimal_table_model.MinimalTable RenameProjectItemCommand (class in spinetool-
method), 195 box.project_commands), 573
RemoveConnectionsCommand (class in spinetool- repair_specification() (spinetool-
box.project_commands), 574 box.project_item.project_item_factory.ProjectItemFactory
RemoveEntitiesDelegate (class in spinetool- static method), 228
box.spine_db_editor.widgets.custom_delegates), repair_specification() (spinetool-
333 box.ui_main.ToolboxUI method), 632
RemoveEntitiesDialog (class in spinetool- replace_arg() (spinetool-
box.spine_db_editor.widgets.edit_or_remove_items_dialogs) box.mvcmodels.file_list_models.CommandLineArgsModel

method), 183

replace_resources_from_downstream() (spine-toolbox.project_item.project_item.ProjectItem method), 223

replace_resources_from_source() (spinetoolbox.project_item.logging_connection.HeadlessConnection method), 216

replace_resources_from_source() (spinetoolbox.project_item.logging_connection.LoggingConnection method), 219

replace_resources_from_upstream() (spinetoolbox.project_item.project_item.ProjectItem method), 222

replace_specification() (spinetoolbox.mvcmodels.project_item_specification_models.ProjectItemSpecificationModel method), 203

replace_specification() (spinetoolbox.project.SpineToolboxProject method), 561

replace_specification() (spinetoolbox.ui_main.ToolboxUI method), 632

ReplaceSpecificationCommand (class in spine-toolbox.project_commands), 576

report_plotting_failure() (in module spine-toolbox.widgets.report_plotting_failure), 490

request_restart_kernel_manager() (spinetoolbox.widgets.jupyter_console_widget.JupyterConsoleWidget method), 454

request_shutdown_kernel_manager() (spinetoolbox.widgets.jupyter_console_widget.JupyterConsoleWidget method), 454

request_start_kernel() (spinetoolbox.widgets.jupyter_console_widget.JupyterConsoleWidget method), 453

REQUIRED_SPINE_OPT_VERSION (in module spine-toolbox.config), 504

reset() (spinetoolbox.mvcmodels.array_model.ArrayModel method), 176

reset() (spinetoolbox.mvcmodels.indexed_value_table_model.IndexedValueTableModel method), 187

reset() (spinetoolbox.mvcmodels.map_model.MapModel method), 191

reset() (spinetoolbox.mvcmodels.time_series_model_fixed_resolution_model.FixedResolutionTimeSeriesModel method), 212

reset() (spinetoolbox.mvcmodels.time_series_model_variable_resolution_model.VariableResolutionTimeSeriesModel method), 214

reset() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.AlternativeTreeView method), 352

reset() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349

reset() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.SquareTreeView method), 353

reset() (spinetoolbox.widgets.persistent_console_widget._CustomLineEdit method), 476

reset_executions_button_text() (spinetoolbox.widgets.custom_qtextbrowser.CustomQTextBrowser method), 433

reset_executions_button_text() (spinetoolbox.widgets.statusbars.MainStatusBar method), 498

reset_fetch_parents() (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.ParameterTableModel method), 290

reset_fetch_parents() (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 285

reset_fetching() (spinetoolbox.fetch_parent.FetchParent method), 507

reset_list_widgets() (spinetoolbox.spine_db_editor.widgets.add_items_dialogs.ObjectGroupDialog method), 319

reset_model() (spinetoolbox.mvcmodels.empty_row_model.EmptyRowModel method), 181

reset_model() (spinetoolbox.mvcmodels.file_list_models.JumpCommandLineArgsModel method), 184

reset_model() (spinetoolbox.mvcmodels.filter_execution_model.FilterExecutionModel method), 186

reset_model() (spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel method), 195

reset_model() (spinetoolbox.spine_db_editor.mvcmodels.pivot_model.PivotModel method), 279

reset_model() (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 286

reset_model() (spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.HalfSolomonTableModel method), 298

reset_model() (spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddOrManageRelationshipsDialog method), 317

reset_model() (spinetoolbox.spine_db_editor.widgets.add_items_dialogs.AddRelationshipsDialog method), 318

reset_model() (spinetoolbox.spine_db_editor.widgets.add_items_dialogs.ManageRelationshipsDialog method), 319

reset_position() (spinetoolbox.spine_db_editor.graphics_items.ObjectLabelItem method), 401

reset_square() (spinetoolbox.spine_db_worker.SpineDBWorker method), 445

RESET_REGISTRY (spinetool-

box.widgets.add_up_spine_opt_wizard._PageId (attribute), 406
 reset_relationship_class_combo_box() (spine-toolbox.spine_db_editor.widgets.add_items_dialogs.ResetRelationshipClassDialog method), 318
 reset_selection() (spinetool-box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel method), 184
 reset_state() (spinetool-box.widgets.custom_qwidgets.FilterWidget method), 437
 reset_zoom() (spinetool-box.widgets.custom_qgraphicsviews.CustomQGraphicsView method), 423
 reset_zoom() (spinetool-box.widgets.custom_qgraphicsviews.DesignQGraphicsView method), 425
 ResetRegistryPage (class in spinetool-box.widgets.add_up_spine_opt_wizard), 407
 ResizableTreeView (class in spinetool-box.spine_db_editor.widgets.custom_qtreeview), 348
 resize_window_to_columns() (spinetool-box.spine_db_editor.widgets.add_items_dialogs.ResizeWindowToColumnsDialog method), 318
 resize_window_to_columns() (spinetool-box.spine_db_editor.widgets.manage_items_dialogs.ResizeWindowToColumnsDialog method), 364
 resizeEvent() (spinetool-box.spine_db_editor.widgets.custom_qtableview.PivotTableView method), 346
 resizeEvent() (spinetool-box.widgets.code_text_edit.CodeTextEdit method), 410
 resizeEvent() (spinetool-box.widgets.custom_qgraphicsviews.CustomQGraphicsView method), 424
 resizeEvent() (spinetool-box.widgets.custom_qwidgets.ElidedTextMixin method), 436
 resizeEvent() (spinetool-box.widgets.multi_tab_window.TabBarPlus method), 468
 resizeEvent() (spinetool-box.widgets.persistent_console_widget.PersistentConsoleWidget method), 478
 ResizingViewMixin (class in spinetool-box.widgets.custom_qwidgets), 442
 resource() (spinetool-box.mvcmodels.file_list_models.FileListModel method), 182
 ResourceFilterModel (class in spinetool-box.mvcmodels.resource_filter_model), 208
 resources_for_direct_predecessors() (spine-toolbox.project_item.project_item.ProjectItem method), 222
 resources_for_indirect_predecessors() (spinetool-box.project_item.project_item.ProjectItem method), 222
 RestartKernel() (spinetool-box.spine_engine_manager.LocalSpineEngineManager method), 620
 restart_kernel() (spinetool-box.spine_engine_manager.RemoteSpineEngineManager method), 622
 restart_kernel() (spinetool-box.spine_engine_manager.SpineEngineManagerBase method), 618
 restart_persistent() (spinetool-box.spine_engine_manager.LocalSpineEngineManager method), 621
 restart_persistent() (spinetool-box.spine_engine_manager.RemoteSpineEngineManager method), 623
 restart_persistent() (spinetool-box.spine_engine_manager.SpineEngineManagerBase method), 619
 RestoreAllSpineDialog() (spinetool-box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 337
 RestoreAndSaveDialog() (spinetool-box.ui_main.ToolboxUI method), 640
 restore_dialog_dimensions() (spinetool-box.widgets.kernel_editor.KernelEditorBase method), 458
 restore_dock_widgets() (spinetool-box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor method), 382
 restore_dock_widgets() (spinetool-box.ui_main.ToolboxUI method), 633
 restore_override_cursor() (spinetool-box.ui_main.ToolboxUI method), 630
 restore_project() (spinetool-box.ui_main.ToolboxUI method), 629
 restore_project_items() (spinetool-box.project.SpineToolboxProject method), 565
 restore_pruned_items() (spinetool-box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 337
 restore_saved_julia_kernel() (spinetool-box.widgets.settings_widget.SettingsWidget method), 497
 restore_saved_python_kernel() (spinetool-box.widgets.settings_widget.SettingsWidget method), 497
 restore_selections() (spinetool-

<code>box.project_item.project_item.ProjectItem</code> method), 221	<code>rollback_session()</code> <code>box.spine_db_manager.SpineDBManager</code> method), 601
<code>restore_ui()</code> (in module <code>spinetoolbox.helpers</code>), 531	<code>rollback_session()</code> (spinetool- <code>box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor</code> method), 381
<code>restore_ui()</code> (spinetoolbox.ui_main.ToolboxUI method), 631	<code>root()</code> (spinetoolbox.mvcmodels.project_item_model.ProjectItemModel method), 199
<code>restore_ui()</code> (spinetool- <code>box.widgets.multi_tab_window.MultiTabWindow</code> method), 468	<code>root_index</code> (spinetool- <code>box.spine_db_editor.mvcmodels.multi_db_tree_model.MultiDBTreeModel</code> property), 270
<code>result()</code> (spinetoolbox.qthread_pool_executor.QtBasedFuture method), 589	<code>root_item</code> (spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_model property), 270
<code>retranslateUi()</code> (spinetool- <code>box.spine_db_editor.ui.scenario_generator.Ui_Form</code> method), 313	<code>root_item_type</code> (spinetool- <code>box.spine_db_editor.mvcmodels.entity_tree_models.ObjectTreeModel</code> property), 254
<code>retranslateUi()</code> (spinetool- <code>box.spine_db_editor.ui.select_databases.Ui_Form</code> method), 313	<code>root_item_type</code> (spinetool- <code>box.spine_db_editor.mvcmodels.entity_tree_models.RelationshipTreeModel</code> property), 255
<code>retranslateUi()</code> (spinetool- <code>box.spine_db_editor.ui.spine_db_editor_window.Ui_MainWindow</code> method), 314	<code>root_item_type</code> (spinetool- <code>box.spine_db_editor.mvcmodels.multi_db_tree_model.MultiDBTreeModel</code> property), 270
<code>retrieve_project()</code> (spinetool- <code>box.server.engine_client.EngineClient</code> method), 233	<code>RootItem</code> (class in spinetool- <code>box.spine_db_editor.mvcmodels.tree_item_utility</code>), 310
<code>retrieve_project()</code> (spinetool- <code>box.ui_main.ToolboxUI</code> method), 635	<code>RootProjectTreeItem</code> (class in spinetool- <code>box.mvcmodels.project_tree_item</code>), 206
<code>return_code</code> (in module <code>spinetoolbox.__main__</code>), 503	<code>rotate_anticlockwise()</code> (spinetool- <code>box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView</code> method), 338
<code>revalidate_workflow()</code> (spinetool- <code>box.project_item.project_item.ProjectItem</code> method), 223	<code>rotate_clockwise()</code> (spinetool- <code>box.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView</code> method), 338
<code>revitalize()</code> (spinetool- <code>box.spine_db_editor.mvcmodels.entity_tree_item.ObjectClassItem</code> method), 250	<code>row()</code> (spinetoolbox.mvcmodels.project_tree_item.BaseProjectTreeItem method), 206
<code>revitalize()</code> (spinetool- <code>box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem</code> method), 269	<code>row()</code> (spinetoolbox.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel method), 256
<code>RIGHT</code> (spinetoolbox.widgets.plot_canvas.LegendPosition attribute), 482	<code>row_data()</code> (spinetool- <code>box.mvcmodels.minimal_table_model.MinimalTableModel</code> method), 194
<code>rollback()</code> (spinetool- <code>box.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel</code> method), 259	<code>row_data()</code> (spinetoolbox.spine_db_editor.mvcmodels.pivot_model.PivotModel method), 281
<code>rollback()</code> (spinetool- <code>box.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel</code> method), 260	<code>rowCount()</code> (spinetool- <code>box.spine_db_editor.mvcmodels.compound_table_model.CompoundTableModel</code> method), 177
<code>rollback()</code> (spinetool- <code>box.spine_db_editor.widgets.item_metadata_editor.ItemMetadataEditor</code> method), 363	<code>rowCount()</code> (spinetool- <code>box.mvcmodels.file_list_models.FileListModel</code> method), 182
<code>rollback()</code> (spinetool- <code>box.spine_db_editor.widgets.metadata_editor.MetadataEditor</code> method), 368	<code>rowCount()</code> (spinetool- <code>box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel</code> method), 185
<code>rollback_session()</code> (spinetool- <code>box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor</code> method), 380	

[rowCount\(\)](#) (spinetoolbox.mvcmodels.filter_execution_model.FilterExecutionModel method), 359
[rowCount\(\)](#) (spinetoolbox.mvcmodels.indexed_value_table_model.IndexedValueTableModel method), 186
[rowCount\(\)](#) (spinetoolbox.mvcmodels.indexed_value_table_model.IndexedValueTableModel method), 620
[rowCount\(\)](#) (spinetoolbox.mvcmodels.indexed_value_table_model.IndexedValueTableModel method), 187
[rowCount\(\)](#) (spinetoolbox.mvcmodels.map_model.MapModel method), 622
[rowCount\(\)](#) (spinetoolbox.mvcmodels.map_model.MapModel method), 191
[rowCount\(\)](#) (spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel method), 618
[rowCount\(\)](#) (spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel method), 193
[rowCount\(\)](#) (spinetoolbox.mvcmodels.minimal_tree_model.MinimalTreeModel method), 198
[rowCount\(\)](#) (spinetoolbox.mvcmodels.project_item_model.ProjectItemModel method), 559
[rowCount\(\)](#) (spinetoolbox.mvcmodels.project_item_model.ProjectItemModel method), 200
[rowCount\(\)](#) (spinetoolbox.mvcmodels.project_item_specification_models.ProjectItemSpecificationModel method), 204
[rowCount\(\)](#) (spinetoolbox.mvcmodels.project_item_specification_models.ProjectItemSpecificationModel method), 204
[rowCount\(\)](#) (spinetoolbox.mvcmodels.frozen_table_model.FrozenTableModel method), 256
[rowCount\(\)](#) (spinetoolbox.mvcmodels.frozen_table_model.FrozenTableModel method), 256
[rowCount\(\)](#) (spinetoolbox.mvcmodels.metadata_table_model_base.MetadataTableModelBase method), 263
[rowCount\(\)](#) (spinetoolbox.mvcmodels.pivot_table_models.PivotTableModelBase method), 287
[rowCount\(\)](#) (spinetoolbox.mvcmodels.pivot_table_models.PivotTableModelBase method), 287
[rows](#) (spinetoolbox.spine_db_editor.mvcmodels.pivot_model.PivotModel property), 279
[rows](#) (spinetoolbox.spine_db_editor.mvcmodels.pivot_model.PivotModel property), 279
[rows_to_row_count_tuples\(\)](#) (in module spinetoolbox.helpers), 522
[rowsInserted\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349
[rowsInserted\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349
[rowsInserted\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ItemTreeView method), 351
[rowsInserted\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ItemTreeView method), 351
[rowsInserted\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView method), 350
[rowsInserted\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ObjectTreeView method), 350
[rowsInserted\(\)](#) (spinetoolbox.widgets.custom_qwidgets.ResizingViewMixin method), 442
[rowsInserted\(\)](#) (spinetoolbox.widgets.custom_qwidgets.ResizingViewMixin method), 442
[rowsRemoved\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349
[rowsRemoved\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeView method), 349
[run\(\)](#) (spinetoolbox.kernel_fetcher.KernelFetcher method), 534
[run\(\)](#) (spinetoolbox.qthread_pool_executor.QtBasedThread method), 590
[run\(\)](#) (spinetoolbox.qthread_pool_executor.QtBasedThread method), 590
[run\(\)](#) (spinetoolbox.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGeneratorRunnable method), 381
[run\(\)](#) (spinetoolbox.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGeneratorRunnable method), 381
[run\(\)](#) (spinetoolbox.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGeneratorRunnable method), 381
[run\(\)](#) (spinetoolbox.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGeneratorRunnable method), 381
[run_engine\(\)](#) (spinetoolbox.spine_engine_manager.LocalSpineEngineManager method), 620
[run_engine\(\)](#) (spinetoolbox.spine_engine_manager.RemoteSpineEngineManager method), 622
[run_engine\(\)](#) (spinetoolbox.spine_engine_manager.SpineEngineManagerBase method), 618
[run_execution_animation\(\)](#) (spinetoolbox.link.JumpOrLink method), 538
[same_path\(\)](#) (in module spinetoolbox.helpers), 533
[save\(\)](#) (spinetoolbox.project.SpineToolboxProject method), 559
[save_and_close\(\)](#) (spinetoolbox.widgets.settings_widget.SettingsWidgetBase method), 496
[save_downloaded_file\(\)](#) (spinetoolbox.server.engine_client.EngineClient method), 496
[save_positions\(\)](#) (spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView method), 357
[save_project\(\)](#) (spinetoolbox.ui_main.ToolboxUI method), 630
[save_project_base\(\)](#) (spinetoolbox.ui_main.ToolboxUI method), 630
[save_selections\(\)](#) (spinetoolbox.project_item.project_item.ProjectItem method), 221
[save_settings\(\)](#) (spinetoolbox.widgets.settings_widget.SettingsWidget method), 496
[save_settings\(\)](#) (spinetoolbox.widgets.settings_widget.SettingsWidgetBase method), 493
[save_settings\(\)](#) (spinetoolbox.widgets.settings_widget.SettingsWidgetBase method), 493
[save_settings\(\)](#) (spinetoolbox.widgets.settings_widget.SpineDBEditorSettingsMixin method), 494
[save_specification_file\(\)](#) (spinetoolbox.project.SpineToolboxProject method), 562
[save_state\(\)](#) (spinetoolbox.widgets.custom_qwidgets.FilterWidget method), 437
[save_state\(\)](#) (spinetoolbox.widgets.custom_qwidgets.FilterWidget method), 437
[save_ui\(\)](#) (in module spinetoolbox.helpers), 531
[save_window_state\(\)](#) (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase method), 381
[save_window_state\(\)](#) (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase method), 381
[save_window_state\(\)](#) (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase method), 381
[save_window_state\(\)](#) (spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase method), 381

- method*), 468
- SaveSpecificationAsCommand (class in *spinetoolbox.project_commands*), 577
- SCATTER (*spinetoolbox.plotting.PlotType* attribute), 547
- SCATTER_LINE (*spinetoolbox.plotting.PlotType* attribute), 547
- SCENARIO_DATA (in module *spinetoolbox.spine_db_editor.mvcmodels.mime_types*), 266
- scenario_selection_changed (*spinetoolbox.spine_db_editor.widgets.custom_qtreeview.ScenarioTreeView* attribute), 353
- ScenarioAlternativeItem (class in *spinetoolbox.spine_db_editor.mvcmodels.scenario_item*), 295
- ScenarioAlternativePivotHeaderView (class in *spinetoolbox.spine_db_editor.widgets.pivot_table_header_view*), 374
- ScenarioAlternativePivotTableModel (class in *spinetoolbox.spine_db_editor.mvcmodels.pivot_table_model*), 292
- ScenarioAlternativeTableDelegate (class in *spinetoolbox.spine_db_editor.widgets.custom_delegates*), 323
- ScenarioDBItem (class in *spinetoolbox.spine_db_editor.mvcmodels.scenario_item*), 294
- ScenarioDelegate (class in *spinetoolbox.spine_db_editor.widgets.custom_delegates*), 330
- ScenarioGenerator (class in *spinetoolbox.spine_db_editor.widgets.scenario_generator*), 375
- ScenarioItem (class in *spinetoolbox.spine_db_editor.mvcmodels.scenario_item*), 294
- ScenarioModel (class in *spinetoolbox.spine_db_editor.mvcmodels.scenario_model*), 296
- ScenarioTreeView (class in *spinetoolbox.spine_db_editor.widgets.custom_qtreeview*), 353
- scene (*spinetoolbox.spine_db_icon_manager._SceneSvgRenderer* attribute), 595
- SceneIconEngine (class in *spinetoolbox.spine_db_icon_manager*), 595
- scrollContentsBy() (*spinetoolbox.widgets.persistent_console_widget.PersistentConsoleWidget* method), 478
- scrolling_to_bottom() (in module *spinetoolbox.helpers*), 532
- search_filter_expression() (*spinetoolbox.mvcmodels.filter_checkbox_list_model.DataToValueFilterCheckboxListModel* method), 186
- search_filter_expression() (*spinetoolbox.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel* method), 185
- SearchBarDelegate (class in *spinetoolbox.spine_db_editor.widgets.object_name_list_editor*), 370
- SearchBarEditor (class in *spinetoolbox.spine_db_editor.widgets.custom_editors*), 414
- select_all_executions() (*spinetoolbox.widgets.custom_qtextbrowser.CustomQTextBrowser* method), 433
- select_certificate_directory() (in module *spinetoolbox.helpers*), 526
- select_conda_executable() (in module *spinetoolbox.helpers*), 526
- SELECT_DIRS (*spinetoolbox.widgets.install_julia_wizard._PageId* attribute), 450
- select_execution() (*spinetoolbox.widgets.custom_qtextbrowser.CustomQTextBrowser* method), 433
- select_gams_executable() (in module *spinetoolbox.helpers*), 526
- select_item() (*spinetoolbox.project_item_icon.ProjectItemIcon* method), 580
- SELECT_JULIA (*spinetoolbox.widgets.add_up_spine_opt_wizard._PageId* attribute), 406
- select_julia_executable() (in module *spinetoolbox.helpers*), 526
- select_julia_project() (in module *spinetoolbox.helpers*), 526
- select_link_drawer() (*spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene* method), 422
- select_position_parameters() (*spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView* method), 337
- select_python_interpreter() (in module *spinetoolbox.helpers*), 526
- SelectDatabaseItems (class in *spinetoolbox.widgets.select_database_items*), 491
- SelectDatabaseItemsDialog (class in *spinetoolbox.widgets.custom_qwidgets*), 442
- SelectDirsPage (class in *spinetoolbox.widgets.install_julia_wizard*), 451
- selected_alternative_ids (*spinetoolbox.spine_db_editor.widgets.custom_qtreeview.AlternativeTreeView* property), 352
- selected_alternative_ids (*spinetoolbox.spine_db_editor.widgets.custom_qtreeview.AlternativeTreeView* property), 352

`box.spine_db_editor.widgets.custom_qtreeview.ScenarioTreeModel` (in module `box.spine_db_editor.widgets.custom_qtreeview`), 353

`SELECTED_COLOR` (in module `spinetoolbox.spine_db_editor.mvcmodels.colors`), 237

`selected_row_changed` (in module `spinetoolbox.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel`), 255

`selection()` (in module `spinetoolbox.widgets.open_project_widget.OpenProjectDialog`), 472

`selection_made` (in module `spinetoolbox.spine_db_editor.widgets.select_position_parameters_dialog.SelectPositionParametersDialog`), 377

`SelectJuliaPage` (class in `spinetoolbox.widgets.add_up_spine_opt_wizard`), 406

`SelectPositionParametersDialog` (class in `spinetoolbox.spine_db_editor.widgets.select_position_parameters_dialog`), 377

`send_get_persistent_completions()` (in module `spinetoolbox.server.engine_client.EngineClient`), 234

`send_get_persistent_history_item()` (in module `spinetoolbox.server.engine_client.EngineClient`), 234

`send_interrupt_persistent()` (in module `spinetoolbox.server.engine_client.EngineClient`), 234

`send_is_complete()` (in module `spinetoolbox.server.engine_client.EngineClient`), 234

`send_issue_persistent_command()` (in module `spinetoolbox.server.engine_client.EngineClient`), 234

`send_kill_persistent()` (in module `spinetoolbox.server.engine_client.EngineClient`), 234

`send_request_to_persistent()` (in module `spinetoolbox.server.engine_client.EngineClient`), 234

`send_request_to_persistent_generator()` (in module `spinetoolbox.server.engine_client.EngineClient`), 234

`send_restart_persistent()` (in module `spinetoolbox.server.engine_client.EngineClient`), 234

`separate_metadata_and_item_metadata()` (in module `spinetoolbox.helpers`), 532

`separator` (in module `spinetoolbox.plotting.ParameterTableHeaderSection`), 548

`serial` (in module `spinetoolbox.version.VersionInfo`), 641

`session_committed` (in module `spinetoolbox.spine_db_manager.SpineDBManager`), 596

`session_refreshed` (in module `spinetoolbox.spine_db_manager.SpineDBManager`), 596

`session_rolled_back` (in module `spinetoolbox.spine_db_manager.SpineDBManager`), 596

`set_action()` (in module `spinetoolbox.widgets.custom_menus.CustomContextMenu`), 418

`set_array_type()` (in module `spinetoolbox.spine_db_editor.widgets.select_position_parameters_dialog.SelectPositionParametersDialog`), 177

`set_auto_check_filters_state()` (in module `spinetoolbox.widgets.link_properties_widget.LinkPropertiesWidget`), 460

`set_auto_expand_objects()` (in module `spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityQGraphicsView`), 336

`set_auto_expand_objects()` (in module `spinetoolbox.widgets.settings_widget.SpineDBEditorSettingsMixin`), 494

`set_auto_filter()` (in module `spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_models.CompoundParameterModel`), 240

`set_auto_filter()` (in module `spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel`), 299

`set_ban_icon()` (in module `spinetoolbox.spine_db_editor.graphics_items.CrossHairsItem`), 399

`set_base_offset()` (in module `spinetoolbox.widgets.custom_editors.SearchBarEditor`), 415

`set_bg_choice()` (in module `spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene`), 422

`set_bg_color()` (in module `spinetoolbox.widgets.custom_qgraphicsscene.DesignGraphicsScene`), 422

`set_box()` (in module `spinetoolbox.mvcmodels.map_model.MapModel`), 191

`set_busy()` (in module `spinetoolbox.fetch_parent.FetchParent`), 508

`set_check_icon()` (in module `spinetoolbox.spine_db_editor.graphics_items.CrossHairsItem`), 399

`set_color()` (in module `spinetoolbox.widgets.project_item_drag.ProjectItemSpecArray`), 488

`set_color()` (in module `spinetoolbox.widgets.toolbars.MainToolBar`), 502

<code>set_color()</code>	(<code>spinetoolbox.widgets.toolbars.PluginToolBar</code> method), 502	<code>set_data()</code>	(<code>spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem</code> method), 267
<code>set_color()</code>	(<code>spinetoolbox.widgets.toolbars.ToolBar</code> method), 501	<code>set_data()</code>	(<code>spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftDataModel</code> method), 284
<code>set_color_and_icon()</code>	(<code>spinetoolbox.widgets.properties_widget.PropertiesWidgetBase</code> method), 489	<code>set_data()</code>	(<code>spinetoolbox.spine_db_editor.mvcmodels.scenario_item.ScenarioAlternativeItem</code> method), 295
<code>set_colored()</code>	(<code>spinetoolbox.helpers.ColoredIconEngine</code> method), 523	<code>set_data()</code>	(<code>spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureRequirementItem</code> method), 305
<code>set_colored()</code>	(<code>spinetoolbox.helpers.ColoredIconEngine</code> method), 524	<code>set_data()</code>	(<code>spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.LeafItemUtility</code> method), 311
<code>set_colored_icons()</code>	(<code>spinetoolbox.widgets.project_item_drag.ProjectItemButtonBase</code> method), 486	<code>set_data()</code>	(<code>spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.StandardTreeItemUtility</code> method), 308
<code>set_colored_icons()</code>	(<code>spinetoolbox.widgets.project_item_drag.ProjectItemSpecArray</code> method), 488	<code>set_data()</code>	(<code>spinetoolbox.widgets.custom_editors.CheckListEditor</code> method), 416
<code>set_colored_icons()</code>	(<code>spinetoolbox.widgets.toolbars.MainToolBar</code> method), 502	<code>set_data()</code>	(<code>spinetoolbox.widgets.custom_editors.CustomLineEditor</code> method), 413
<code>set_condition()</code>	(<code>spinetoolbox.widgets.jump_properties_widget.JumpPropertiesWidget</code> method), 452	<code>set_data()</code>	(<code>spinetoolbox.widgets.custom_editors.IconColorEditor</code> method), 417
<code>set_connection_file()</code>	(<code>spinetoolbox.widgets.jupyter_console_widget.JupyterConsoleWidget</code> method), 454	<code>set_data()</code>	(<code>spinetoolbox.widgets.custom_editors.ParameterValueLineEdit</code> method), 414
<code>set_connection_options()</code>	(<code>spinetoolbox.project_item.logging_connection.LoggingConnection</code> method), 219	<code>set_data()</code>	(<code>spinetoolbox.widgets.custom_qgraphicsviews.EntityClassGraphicsView</code> method), 414
<code>set_cross_hairs_items()</code>	(<code>spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityClassGraphicsView</code> method), 337	<code>set_db_maps()</code>	(<code>spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model_base.MetadataTableModelBase</code> method), 263
<code>set_current_tab()</code>	(<code>spinetoolbox.widgets.multi_tab_window.MultiTabWindow</code> method), 467	<code>set_debug_actions()</code>	(<code>spinetoolbox.ui_main.ToolboxUI</code> method), 633
<code>set_current_urls()</code>	(<code>spinetoolbox.spine_db_editor.widgets.url_toolbar.UrlToolBar</code> method), 393	<code>set_default_parameter_data()</code>	(<code>spinetoolbox.spine_db_editor.widgets.parameter_view_mixin.ParameterViewMixin</code> method), 371
<code>set_data()</code>	(<code>spinetoolbox.mvcmodels.minimal_tree_model.TreeItem</code> method), 197	<code>set_default_row()</code>	(<code>spinetoolbox.mvcmodels.empty_row_model.EmptyRowModel</code> method), 181
<code>set_data()</code>	(<code>spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.EntityClassItem</code> method), 249	<code>set_description()</code>	(<code>spinetoolbox.metaobject.MetaObject</code> method), 545
<code>set_data()</code>	(<code>spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.EntityClassItem</code> method), 252	<code>set_description()</code>	(<code>spinetoolbox.project.SpineToolboxProject</code> method), 559
<code>set_data()</code>	(<code>spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.EntityClassItem</code> method), 248	<code>set_relationship_max_dimension()</code>	(<code>spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityClassGraphicsView</code> method), 336
<code>set_data()</code>	(<code>spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.EntityClassItem</code> method), 248	<code>set_relationship_max_dimension()</code>	(<code>spinetoolbox.spine_db_editor.widgets.custom_qgraphicsviews.EntityClassGraphicsView</code> method), 336

<code>box.spine_engine_worker.SpineEngineWorker</code>	<code>set_frozen()</code>	<code>(spinetool-</code>
<code>method)</code> , 625	<code>box.spine_db_editor.mvcmodels.pivot_model.PivotModel</code>	<code>method)</code> , 280
<code>set_entity_ids()</code>	<code>(spinetool-</code>	<code>set_frozen_value()</code>
<code>box.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel</code>	<code>method)</code> , 258	<code>box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel</code>
<code>set_error_mode()</code>	<code>(spinetoolbox.ui_main.ToolboxUI</code>	<code>method)</code> , 286
<code>static method)</code> , 628	<code>set_frozen_value()</code>	<code>(spinetool-</code>
<code>set_exception()</code>	<code>(spinetool-</code>	<code>box.spine_db_editor.mvcmodels.pivot_model.PivotModel</code>
<code>box.qthread_pool_executor.QtBasedFuture</code>	<code>method)</code> , 589	<code>method)</code> , 280
<code>set_external_copy_and_paste_actions()</code>	<code>(spine-</code>	<code>set_frozen_value()</code>
<code>toolbox.widgets.custom_qtableview.CopyPasteTableView</code>	<code>method)</code> , 428	<code>box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel</code>
<code>set_fetched()</code>	<code>(spinetoolbox.fetch_parent.FetchParent</code>	<code>method)</code> , 286
<code>method)</code> , 508	<code>method)</code> , 508	<code>set_headers()</code>
<code>set_filter()</code>	<code>(spinetool-</code>	<code>box.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel</code>
<code>box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel</code>	<code>method)</code> , 185	<code>method)</code> , 255
<code>set_filter()</code>	<code>(spinetool-</code>	<code>set_horizontal_header_labels()</code>
<code>box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel</code>	<code>method)</code> , 293	<code>(spinetool-</code>
<code>set_filter_accepted_values()</code>	<code>(spinetool-</code>	<code>set_hover_brush()</code>
<code>box.spine_db_editor.widgets.custom_menus.ParameterView</code>	<code>method)</code> , 334	<code>box.spine_db_editor.widgets.connector_button.ConnectorButton</code>
<code>set_filter_alternative_ids()</code>	<code>(spinetool-</code>	<code>method)</code> , 581
<code>box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel</code>	<code>method)</code> , 243	<code>set_icon()</code>
<code>set_filter_alternative_ids()</code>	<code>(spinetool-</code>	<code>(spinetool-</code>
<code>box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel</code>	<code>method)</code> , 301	<code>box.spine_db_editor.widgets.connector_button.ConnectorButton</code>
<code>set_filter_class_ids()</code>	<code>(spinetool-</code>	<code>method)</code> , 221
<code>box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel</code>	<code>method)</code> , 240	<code>set_icon()</code>
<code>set_filter_default_online_status()</code>	<code>(spinetool-</code>	<code>(spinetool-</code>
<code>box.project_item.logging_connection.LoggingConnection</code>	<code>method)</code> , 218	<code>box.spine_db_editor.widgets.connector_button.ConnectorButton</code>
<code>set_filter_enabled()</code>	<code>(spinetool-</code>	<code>method)</code> , 399
<code>box.project_item.logging_connection.HeadlessConnection</code>	<code>method)</code> , 216	<code>set_icon_and_properties_ui()</code>
<code>set_filter_entity_ids()</code>	<code>(spinetool-</code>	<code>(spinetool-</code>
<code>box.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel</code>	<code>method)</code> , 242	<code>box.spine_db_editor.widgets.connector_button.ConnectorButton</code>
<code>set_filter_entity_ids()</code>	<code>(spinetool-</code>	<code>method)</code> , 638
<code>box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel</code>	<code>method)</code> , 301	<code>set_ignore_year()</code>
<code>set_filter_list()</code>	<code>(spinetool-</code>	<code>(spinetool-</code>
<code>box.widgets.custom_qwidgets.FilterWidget</code>	<code>method)</code> , 437	<code>box.mvcmodels.time_series_model_fixed_resolution.TimeSeriesModel</code>
<code>set_filter_rejected_values()</code>	<code>(spinetool-</code>	<code>box.mvcmodels.time_series_model_variable_resolution.TimeSeriesModel</code>
<code>box.spine_db_editor.widgets.custom_menus.ParameterView</code>	<code>method)</code> , 334	<code>method)</code> , 215
<code>set_friend_connectors_enabled()</code>	<code>(spinetool-</code>	<code>set_item()</code>
<code>box.project_item_icon.ConnectorButton</code>	<code>method)</code> , 581	<code>(spinetool-</code>
		<code>box.widgets.properties_widget.PropertiesWidgetBase</code>
		<code>method)</code> , 489
		<code>set_item_log_selected()</code>
		<code>(spinetool-</code>
		<code>box.widgets.custom_qtextbrowser.CustomQTextBrowser</code>
		<code>method)</code> , 422
		<code>set_julia_exe()</code>
		<code>(spinetool-</code>
		<code>box.widgets.install_julia_wizard.InstallJuliaWizard</code>
		<code>method)</code> , 459
		<code>set_kernel_name()</code>
		<code>(spinetool-</code>
		<code>box.widgets.kernel_editor.MiniPythonKernelEditor</code>
		<code>method)</code> , 459
		<code>set_keyboard_shortcuts()</code>
		<code>(spinetool-</code>
		<code>box.widgets.open_project_widget.OpenProjectDialog</code>
		<code>method)</code> , 471
		<code>set_killed()</code>
		<code>(spinetool-</code>
		<code>box.widgets.persistent_console_widget.PersistentConsoleWidget</code>
		<code>method)</code> , 479
		<code>set_layout_generator()</code>
		<code>(spinetool-</code>

<code>box.spine_db_editor.widgets.graph_layout_generators.PivotBarWidget</code> (method), 358	<code>set_pivot()</code> (spinetool- box.spine_db_editor.mvcmodels.pivot_model.PivotModel method), 280
<code>set_lexer_name()</code> (spinetool- box.widgets.code_text_edit.CodeTextEdit method), 410	<code>set_pivot()</code> (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableM method), 286
<code>set_link()</code> (spinetool- box.widgets.jump_properties_widget.JumpPropertiesWidget method), 452	<code>set_pivot_x_column()</code> (spinetool- box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableM method), 286
<code>set_link()</code> (spinetool- box.widgets.link_properties_widget.LinkPropertiesWidget method), 460	<code>set_plus_icon()</code> (spinetool- box.spine_db_editor.graphics_items.CrossHairsItem method), 399
<code>set_list()</code> (spinetool- box.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel method), 185	<code>set_plus_without_bumping()</code> (spinetool- box.project_item_icon.ProjectItemIcon method), 580
<code>set_merge_dbs()</code> (spinetool- box.spine_db_editor.widgets.custom_qgraphicsviews.MergeDatabasesDialog method), 336	<code>set_project_actions_enabled()</code> (spinetool- box.widgets.toolbars.MainToolBar method), 502
<code>set_merge_dbs()</code> (spinetool- box.widgets.settings_widget.SpineDBEditorSettingsWidget method), 494	<code>set_project_description()</code> (spinetool- box.widgets.toolbars.ToolBar method), 501
<code>set_model_data()</code> (spinetool- box.spine_db_editor.widgets.manage_items_dialogs.ManageItemsDialogToolboxUI method), 364	<code>set_properties_ui()</code> (spinetool- box.project_item.project_item.ProjectItem method), 231
<code>set_models()</code> (spinetool- box.spine_db_editor.widgets.custom_qtableview.ItemMetadataTableV method), 347	<code>set_rank()</code> (spinetool- box.mvcmodels.file_list_models.CommandLineArgItem method), 183
<code>set_name()</code> (spinetoolbox.metaobject.MetaObject method), 544	<code>set_rank()</code> (spinetool- box.project_item.project_item.ProjectItem method), 222
<code>set_name_attributes()</code> (spinetool- box.project_item_icon.ProjectItemIcon method), 578	<code>set_rank()</code> (spinetoolbox.project_item_icon.RankIcon method), 582
<code>set_normal_brush()</code> (spinetool- box.project_item_icon.ConnectorButton method), 581	<code>set_raw_text()</code> (spinetool- box.widgets.persistent_console_widget._CustomLineEdit method), 476
<code>set_normal_icon()</code> (spinetool- box.spine_db_editor.graphics_items.CrossHairsItem method), 399	<code>set_repeat()</code> (spinetool- box.mvcmodels.time_series_model_fixed_resolution.TimeSeriesM method), 213
<code>set_obsolete()</code> (spinetool- box.fetch_parent.FetchParent method), 508	<code>set_repeat()</code> (spinetool- box.mvcmodels.time_series_model_variable_resolution.TimeSerie method), 215
<code>set_online()</code> (spinetool- box.mvcmodels.resource_filter_model.ResourceFilterModel method), 209	<code>set_resolution()</code> (spinetool- box.mvcmodels.time_series_model_fixed_resolution.TimeSeriesM method), 213
<code>set_online()</code> (spinetool- box.project_item.logging_connection.LoggingConnection method), 218	<code>set_result()</code> (spinetool- box.qthread_pool_executor.QtBasedFuture method), 589
<code>set_opacity()</code> (spinetool- box.widgets.notification.Notification method), 470	<code>set_rows_to_default()</code> (spinetool- box.mvcmodels.empty_row_model.EmptyRowModel method), 181
<code>set_orientation()</code> (spinetool- box.widgets.project_item_drag.NiceButton method), 486	<code>set_scenario_alternatives()</code> (spinetool- box.spine_db_editor.mvcmodels.item_metadata_table_model.SpineDBManager method), 609
<code>set_parameter_value_ids()</code> (spinetool- box.spine_db_editor.mvcmodels.item_metadata_table_model.SpineDBManager method), 258	

[set_selected\(\)](#) (spinetoolbox.widgets.plain_parameter_value_editor.PlainParameterValueEditor method), 185
[set_selected\(\)](#) (spinetoolbox.mvcmodels.filter_checkbox_list_model.SimpleFilterCheckboxListModel method), 185
[set_selected\(\)](#) (spinetoolbox.spine_db_editor.mvcmodels.frozen_table_model.FrozenTableModel method), 256
[set_selected_path\(\)](#) (spinetoolbox.widgets.open_project_widget.OpenProjectDialog method), 472
[set_show_previews\(\)](#) (spinetoolbox.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGeneratorRunnable method), 359
[set_size_according_to_parent\(\)](#) (spinetoolbox.widgets.plot_widget._PlotDataWidget method), 484
[set_specification\(\)](#) (spinetoolbox.project_item.project_item.ProjectItem method), 221
[set_start\(\)](#) (spinetoolbox.mvcmodels.time_series_model_fixed_resolution_model.FixedResolutionModel method), 213
[set_start_time\(\)](#) (spinetoolbox.server.engine_client.EngineClient method), 232
[set_style\(\)](#) (spinetoolbox.helpers.CustomSyntaxHighlighter method), 531
[set_taskbar_icon\(\)](#) (in module spinetoolbox.helpers), 521
[set_toolbar_colored_icons\(\)](#) (spinetoolbox.widgets.settings_widget.SettingsWidget method), 496
[set_toolbox\(\)](#) (spinetoolbox.widgets.custom_qtextbrowser.CustomQTextBrowser method), 433
[set_ui\(\)](#) (spinetoolbox.widgets.custom_qgraphicsviews.DesignQGraphicsView method), 425
[set_up\(\)](#) (spinetoolbox.mvcmodels.minimal_tree_model.TreeItem method), 197
[set_up\(\)](#) (spinetoolbox.project_item.project_item.ProjectItem method), 225
[set_value\(\)](#) (spinetoolbox.widgets.array_editor.ArrayEditor method), 408
[set_value\(\)](#) (spinetoolbox.widgets.datetime_editor.DatetimeEditor method), 443
[set_value\(\)](#) (spinetoolbox.widgets.duration_editor.DurationEditor method), 444
[set_value\(\)](#) (spinetoolbox.widgets.map_editor.MapEditor method), 461
[set_value\(\)](#) (spinetoolbox.widgets.plain_parameter_value_editor.PlainParameterValueEditor method), 185
[set_value\(\)](#) (spinetoolbox.widgets.time_pattern_editor.TimePatternEditor method), 499
[set_value\(\)](#) (spinetoolbox.widgets.time_series_fixed_resolution_editor.TimeSeriesFixedResolutionEditor method), 500
[set_value\(\)](#) (spinetoolbox.widgets.time_series_variable_resolution_editor.TimeSeriesVariableResolutionEditor method), 500
[set_visible\(\)](#) (spinetoolbox.widgets.project_item_drag.ProjectItemSpecArray method), 489
[set_work_directory\(\)](#) (spinetoolbox.ui_main.ToolboxUI method), 628
[set_work_directory\(\)](#) (spinetoolbox.widgets.settings_widget.SettingsWidget method), 496
[set_file_size\(\)](#) (spinetoolbox.mvcmodels.time_series_model_fixed_resolution_model.FixedResolutionModel method), 213
[set_undo_stack\(\)](#) (spinetoolbox.spine_db_commands.AgedUndoStack method), 590
[SetConnectionDefaultFilterOnlineStatus](#) (class in spinetoolbox.project_commands), 575
[SetConnectionOptionsCommand](#) (class in spinetoolbox.project_commands), 576
[setData\(\)](#) (spinetoolbox.mvcmodels.array_model.ArrayModel method), 177
[setData\(\)](#) (spinetoolbox.mvcmodels.file_list_models.CommandLineArgItem method), 183
[setData\(\)](#) (spinetoolbox.mvcmodels.file_list_models.NewCommandLineArgItem method), 183
[setData\(\)](#) (spinetoolbox.mvcmodels.map_model.MapModel method), 191
[setData\(\)](#) (spinetoolbox.mvcmodels.minimal_table_model.MinimalTableModel method), 194
[setData\(\)](#) (spinetoolbox.mvcmodels.minimal_tree_model.MinimalTreeModel method), 199
[setData\(\)](#) (spinetoolbox.mvcmodels.resource_filter_model.ResourceFilter method), 209
[setData\(\)](#) (spinetoolbox.mvcmodels.time_pattern_model.TimePatternModel method), 211
[setData\(\)](#) (spinetoolbox.mvcmodels.time_series_model_fixed_resolution_model.FixedResolutionModel method), 212
[setData\(\)](#) (spinetoolbox.mvcmodels.time_series_model_variable_resolution_model.VariableResolutionModel method), 214
[setData\(\)](#) (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel method), 263
[setData\(\)](#) (spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel method), 289
[SetDescriptionDialog](#) (class in spinetoolbox.widgets.set_description_dialog), 492
[setDocument\(\)](#) (spinetoolbox.widgets.code_text_edit.CodeTextEdit method), 492

<code>method</code>), 410	<code>setItemSpecificationCommand</code> (class in <code>spinetool-box.project_commands</code>), 571
<code>setEditorData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.AlternativeDelegate</code> method), 330	<code>setItemSpecificationCommand</code> (class in <code>spinetool-box.project_commands</code>), 574
<code>setEditorData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ParameterDelegate</code> method), 325	<code>setMinimum()</code> (<code>spinetool-box.widgets.custom_qwidgets.HorizontalSpinBox</code> method), 441
<code>setEditorData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ParameterPivotTableDelegate</code> method), 324	<code>setModel()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_qtableview.PivotTableView</code> method), 346
<code>setEditorData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ParameterValueDelegate</code> method), 331	<code>setModel()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_qtreeview.EntityTreeView</code> method), 349
<code>setEditorData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.RelationshipPivotTableDelegate</code> method), 323	<code>setModel()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_qtableview.AutoFilterCopyPasteTableView</code> method), 429
<code>setEditorData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ScenarioAlternativeDelegate</code> method), 323	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.AlternativeDelegate</code> method), 330
<code>setEditorData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ScenarioDelegate</code> method), 330	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ManageItemsDelegate</code> method), 331
<code>setEditorData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ToolFeatureDelegate</code> method), 329	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ParameterDelegate</code> method), 325
<code>setEditorData()</code> (<code>spinetool-box.spine_db_editor.widgets.select_position_parameters_dialog.ParameterEditorWidget</code> method), 377	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ParameterPivotTableDelegate</code> method), 324
<code>setEditorData()</code> (<code>spinetool-box.widgets.custom_delegates.ComboBoxDelegate</code> method), 412	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ParameterValueEditor</code> method), 324
<code>SetFiltersOnlineCommand</code> (class in <code>spinetool-box.project_commands</code>), 575	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ParameterValueListEditor</code> method), 331
<code>setFormatScope()</code> (<code>spinetool-box.widgets.persistent_console_widget.AnsiEscapesController</code> method), 480	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ParameterValueOnTableEditor</code> method), 326
<code>setHeaderData()</code> (<code>spinetool-box.mvcmodels.array_model.ArrayModel</code> method), 177	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.RelationshipPivotTableDelegate</code> method), 323
<code>setHeaderData()</code> (<code>spinetool-box.mvcmodels.indexed_value_table_model.IndexedValueTableModel</code> method), 188	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ScenarioAlternativeDelegate</code> method), 323
<code>setHeaderData()</code> (<code>spinetool-box.mvcmodels.map_model.MapModel</code> method), 191	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ScenarioDelegate</code> method), 330
<code>setHeaderData()</code> (<code>spinetool-box.mvcmodels.minimal_table_model.MinimalTableModel</code> method), 194	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_delegates.ToolFeatureDelegate</code> method), 329
<code>setHorizontalHeader()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_qtableview.PivotTableEditor</code> method), 346	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.object_name_list_editor.SearchBarEditor</code> method), 370
<code>setIndexWidget()</code> (<code>spinetool-box.spine_db_editor.widgets.custom_qtableview.PivotTableEditor</code> method), 346	<code>setModelData()</code> (<code>spinetool-box.spine_db_editor.widgets.select_position_parameters_dialog.PivotTableEditor</code> method), 370

box.ui_main.ToolboxUI method), 635

`show_add_relationship_classes_form()` (*spine-
toolbox.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin*
method), 391

`show_add_relationships_form()` (*spinetool-
box.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin*
method), 391

`show_all_hidden_items()` (*spinetool-
box.spine_db_editor.widgets.custom_qgraphicsviews.EntityGraphicsView*
method), 336

`show_auto_filter_menu()` (*spinetool-
box.widgets.custom_qtableview.AutoFilterCopyPasteTableView*
method), 429

`show_color_dialog()` (*spinetool-
box.widgets.settings_widget.SettingsWidget*
method), 496

`show_context_menu()` (*spinetool-
box.spine_db_editor.widgets.custom_qtableview.PivotTableView*
method), 343

`show_context_menu()` (*spinetool-
box.spine_db_editor.widgets.custom_qtableview.PivotTableView*
method), 344

`show_context_menu()` (*spinetool-
box.widgets.open_project_widget.OpenProjectDialog*
method), 473

`show_edit_object_classes_form()` (*spinetool-
box.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin*
method), 391

`show_edit_objects_form()` (*spinetool-
box.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin*
method), 391

`show_edit_relationship_classes_form()` (*spine-
toolbox.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin*
method), 391

`show_edit_relationships_form()` (*spinetool-
box.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin*
method), 391

`show_error()` (*spinetool-
box.project_item.specification_editor_window.SpecificationEditorWidget*
method), 229

`show_getting_started_guide()` (*spinetool-
box.ui_main.ToolboxUI method*), 635

`show_hidden_items()` (*spinetool-
box.spine_db_editor.widgets.custom_qgraphicsviews.EntityGraphicsView*
method), 337

`show_icon_color_editor()` (*spinetool-
box.spine_db_editor.widgets.manage_items_dialogs.ShowIconColorEditorWidget*
method), 365

`show_install_plugin_dialog()` (*spinetool-
box.plugin_manager.PluginManager method*), 556

`show_item_context_menu()` (*spinetool-
box.ui_main.ToolboxUI method*), 635

`show_julia_kernel_context_menu_on_combobox()` (*spinetoolbox.widgets.settings_widget.SettingsWidget*
method), 495

`show_julia_kernel_context_menu_on_combobox_list()` (*spinetoolbox.widgets.settings_widget.SettingsWidget*
method), 495

`show_manage_members_form()` (*spinetool-
box.ui_main.ToolboxUI method*), 636

`show_manage_plugins_dialog()` (*spinetool-
box.plugin_manager.PluginManager method*), 556

`show_manage_relationships_form()` (*spinetool-
box.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin*
method), 391

`show_mass_export_items_dialog()` (*spinetool-
box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase*
method), 380

`show_mass_remove_items_form()` (*spinetool-
box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase*
method), 380

`show_object_name_list_editor()` (*spinetool-
box.spine_db_editor.widgets.parameter_view_mixin.ParameterView*
method), 371

`show_parameter_value_editor()` (*spinetool-
box.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase*
method), 381

`show_plot_data()` (*spinetool-
box.widgets.plot_widget.PlotWidget method*), 484

`show_plus_button_context_menu()` (*spinetool-
box.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDBEditor*
method), 369

`show_plus_button_context_menu()` (*spinetool-
box.widgets.multi_tab_spec_editor.MultiTabSpecEditor*
method), 463

`show_plus_button_context_menu()` (*spinetool-
box.widgets.multi_tab_spec_editor.MultiTabSpecEditor*
method), 464

`show_project_or_item_context_menu()` (*spinetool-
box.ui_main.ToolboxUI method*), 635

`show_python_kernel_context_menu_on_combobox()` (*spinetoolbox.widgets.settings_widget.SettingsWidget*
method), 495

`show_python_kernel_context_menu_on_combobox_list()` (*spinetoolbox.widgets.settings_widget.SettingsWidget*
method), 495

`show_recent_projects_menu()` (*spinetool-
box.ui_main.ToolboxUI method*), 630

`show_remove_entity_tree_items_form()` (*spine-
toolbox.spine_db_editor.widgets.tree_view_mixin.TreeViewMixin*
method), 391

`show_settings()` (*spinetoolbox.ui_main.ToolboxUI*

method), 635

show_specification_context_menu() (spinetoolbox.ui_main.ToolboxUI method), 633

show_specification_form() (spinetoolbox.ui_main.ToolboxUI method), 635

show_user_guide() (spinetoolbox.spine_db_editor.widgets.multi_spine_db_editor.MultiSpineDBEditor method), 369

show_user_guide() (spinetoolbox.ui_main.ToolboxUI method), 635

showEvent() (spinetoolbox.widgets.project_item_drag.ProjectItemSpecArea method), 488

ShowIconColorEditorMixin (class in spinetoolbox.spine_db_editor.widgets.manage_items_dialogs), 365

shows_item() (spinetoolbox.fetch_parent.FetchParent method), 508

shows_item() (spinetoolbox.fetch_parent.FlexibleFetchParent method), 511

shows_item() (spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_definition_model.CompoundParameterDefinitionModel method), 239

shutdown() (spinetoolbox.qthread_pool_executor.QtBasedThreadPoolExecutor method), 590

shutdown_kernel() (spinetoolbox.spine_engine_manager.LocalSpineEngineManager method), 620

shutdown_kernel() (spinetoolbox.spine_engine_manager.RemoteSpineEngineManager method), 622

shutdown_kernel() (spinetoolbox.spine_engine_manager.SpineEngineManagerBase method), 618

shutdown_kernel_client() (spinetoolbox.widgets.jupyter_console_widget.JupyterConsoleWidget method), 454

signal_waiter() (in module spinetoolbox.helpers), 530

SignalWaiter (class in spinetoolbox.helpers), 530

SimpleFilterCheckboxListModel (class in spinetoolbox.mvcmodels.filter_checkbox_list_model), 184

single_models (spinetoolbox.mvcmodels.compound_table_model.CompoundWithEmptyTableModel property), 180

SingleObjectParameterDefinitionModel (class in spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models), 301

SingleObjectParameterDefinitionMixin (class in spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models), 300

SingleObjectParameterValueModel (class in spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models), 301

SingleParameterDefinitionMixin (class in spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models), 300

SingleParameterModel (class in spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models), 298

SingleParameterValueMixin (class in spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models), 300

SingleRelationshipParameterDefinitionModel (class in spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models), 301

SingleRelationshipParameterMixin (class in spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models), 300

SingleRelationshipParameterValueModel (class in spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models), 301

sizeHint() (spinetoolbox.spine_db_editor.widgets.commit_viewer._AffectedItemsFromCommit method), 321

sizeHint() (spinetoolbox.spine_db_editor.widgets.url_toolbar._DBListWidget method), 393

sizeHint() (spinetoolbox.spine_db_editor.widgets.url_toolbar._FilterArrayWidget method), 393

sizeHint() (spinetoolbox.spine_db_editor.widgets.url_toolbar._FilterWidget method), 393

sizeHint() (spinetoolbox.spine_db_editor.widgets.url_toolbar._UrlFilterDialog method), 393

sizeHint() (spinetoolbox.widgets.code_text_edit.LineNumberArea method), 410

sizeHint() (spinetoolbox.widgets.menu_widget._MenuToolBar method), 439

sleep() (spinetoolbox.link.ConnectionLinkDrawer method), 540

sleep() (spinetoolbox.link.LinkDrawerBase method), 540

solve_connection_file() (in module spinetoolbox.helpers), 533

sort() (spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel method), 533

<i>method</i>), 265	<i>specifications()</i> (<i>spinetool-</i>
SortChildrenMixin (class in <i>spinetool-</i>	<i>box.mvcmodels.project_item_specification_models.FilteredSpecifi</i>
<i>box.spine_db_editor.mvcmodels.tree_item_utility</i>),	<i>method</i>), 205
309	<i>specifications()</i> (<i>spinetool-</i>
source_model (<i>spinetool-</i>	<i>box.project.SpineToolboxProject</i>
<i>box.spine_db_editor.widgets.custom_qtableview.PivotTable</i>	<i>method</i>),
<i>property</i>), 346	SpineDBCommand (class in <i>spinetool-</i>
SourcesTreeView (class in <i>spinetool-</i>	<i>box.spine_db_commands</i>), 592
<i>box.widgets.custom_qtreeview</i>), 435	SpineDBEditor (class in <i>spinetool-</i>
spec_name (<i>spinetoolbox.widgets.project_item_drag.ProjectItemSpec</i>	<i>box.spine_db_editor.widgets.spine_db_editor</i>),
<i>property</i>), 487	382
spec_toolbar() (<i>spinetool-</i>	SpineDBEditorBase (class in <i>spinetool-</i>
<i>box.project_item.specification_editor_window.SpecificationEditorWindow</i>	<i>box.spine_db_editor.widgets.spine_db_editor</i>),
<i>method</i>), 229	378
specification() (<i>spinetool-</i>	SpineDBEditorSettingsMixin (class in <i>spinetool-</i>
<i>box.mvcmodels.project_item_specification_models.FilteredSpecificationModels</i>	<i>box.widgets.settings_widget</i>), 493
<i>method</i>), 205	SpineDBEditorSettingsWidget (class in <i>spinetool-</i>
specification() (<i>spinetool-</i>	<i>box.widgets.settings_widget</i>), 494
<i>box.mvcmodels.project_item_specification_models.SpineDBIconManagerModel</i>	SpineDBIconManager (class in <i>spinetool-</i>
<i>method</i>), 205	<i>box.spine_db_icon_manager</i>), 595
specification() (<i>spinetool-</i>	SpineDBMacro (class in <i>spinetool-</i>
<i>box.project_item.project_item.ProjectItem</i>	<i>box.spine_db_commands</i>), 591
<i>method</i>), 221	SpineDBManager (class in <i>spinetool-</i>
specification_about_to_be_removed (<i>spinetool-</i>	<i>box.spine_db_manager</i>), 596
<i>box.project.SpineToolboxProject</i> attribute),	SpineDBParcel (class in <i>spinetool-</i>
558	<i>box.spine_db_parcel</i>), 611
specification_added (<i>spinetool-</i>	SpineDBWorker (class in <i>spinetool-</i>
<i>box.project.SpineToolboxProject</i> attribute),	<i>box.spine_db_worker</i>), 613
558	SpineEngineManagerBase (class in <i>spinetool-</i>
specification_from_dict() (in module <i>spinetool-</i>	<i>box.spine_engine_manager</i>), 618
<i>box.helpers</i>), 529	SpineEngineWorker (class in <i>spinetool-</i>
specification_index() (<i>spinetool-</i>	<i>box.spine_engine_worker</i>), 625
<i>box.mvcmodels.project_item_specification_models.SpineToolboxSpecificationModel</i>	<i>box.spine_engine_worker</i>), 625
<i>method</i>), 205	SpineToolboxSpecificationModel
SPECIFICATION_LOCAL_DATA_FILENAME (in module	module, 175
<i>spinetoolbox.config</i>), 504	spinetoolbox.__main__
specification_name_to_id() (<i>spinetool-</i>	module, 503
<i>box.project.SpineToolboxProject</i> method),	spinetoolbox._version
561	module, 504
specification_replaced (<i>spinetool-</i>	spinetoolbox.config
<i>box.mvcmodels.project_item_specification_models.SpineToolboxSpecificationModel</i>	module, 504
attribute), 203	SpineToolboxSpecificationModel
specification_replaced (<i>spinetool-</i>	module, 505
<i>box.project.SpineToolboxProject</i> attribute),	spinetoolbox.fetch_parent
558	module, 507
specification_row() (<i>spinetool-</i>	spinetoolbox.headless
<i>box.mvcmodels.project_item_specification_models.SpineToolboxSpecificationModel</i>	module, 511
<i>method</i>), 205	SpineToolboxSpecificationModel
specification_saved (<i>spinetool-</i>	module, 516
<i>box.project.SpineToolboxProject</i> attribute),	spinetoolbox.kernel_fetcher
559	module, 534
SpecificationEditorWindowBase (class in <i>spinetool-</i>	spinetoolbox.link
<i>box.project_item.specification_editor_window</i>),	module, 535
229	spinetoolbox.load_project_items
	module, 541

spinetoolbox.log_mixin	spinetoolbox.project_item
module, 541	module, 215
spinetoolbox.logger_interface	spinetoolbox.project_item.logging_connection
module, 542	module, 215
spinetoolbox.main	spinetoolbox.project_item.project_item
module, 543	module, 219
spinetoolbox.metaobject	spinetoolbox.project_item.project_item_factory
module, 544	module, 226
spinetoolbox.mvcmodels	spinetoolbox.project_item.specification_editor_window
module, 175	module, 228
spinetoolbox.mvcmodels.array_model	spinetoolbox.project_item_icon
module, 175	module, 577
spinetoolbox.mvcmodels.compound_table_model	spinetoolbox.project_settings
module, 177	module, 583
spinetoolbox.mvcmodels.empty_row_model	spinetoolbox.project_upgrader
module, 181	module, 583
spinetoolbox.mvcmodels.file_list_models	spinetoolbox.qthread_pool_executor
module, 182	module, 589
spinetoolbox.mvcmodels.filter_checkbox_list_model	spinetoolbox.server
module, 184	module, 231
spinetoolbox.mvcmodels.filter_execution_model	spinetoolbox.server.engine_client
module, 186	module, 231
spinetoolbox.mvcmodels.indexed_value_table_model	spinetoolbox.spine_db_commands
module, 187	module, 590
spinetoolbox.mvcmodels.map_model	spinetoolbox.spine_db_editor
module, 188	module, 235
spinetoolbox.mvcmodels.minimal_table_model	spinetoolbox.spine_db_editor.graphics_items
module, 193	module, 394
spinetoolbox.mvcmodels.minimal_tree_model	spinetoolbox.spine_db_editor.main
module, 196	module, 401
spinetoolbox.mvcmodels.project_item_model	spinetoolbox.spine_db_editor.mvcmodels
module, 199	module, 235
spinetoolbox.mvcmodels.project_item_specification_model	spinetoolbox.spine_db_editor.mvcmodels.alternative_item
module, 203	module, 235
spinetoolbox.mvcmodels.project_tree_item	spinetoolbox.spine_db_editor.mvcmodels.alternative_model
module, 205	module, 236
spinetoolbox.mvcmodels.resource_filter_model	spinetoolbox.spine_db_editor.mvcmodels.colors
module, 208	module, 237
spinetoolbox.mvcmodels.shared	spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_model
module, 210	module, 237
spinetoolbox.mvcmodels.time_pattern_model	spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_model
module, 210	module, 244
spinetoolbox.mvcmodels.time_series_model_fixed_resolution	spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item
module, 211	module, 248
spinetoolbox.mvcmodels.time_series_model_variable_resolution	spinetoolbox.spine_db_editor.mvcmodels.entity_tree_models
module, 213	module, 254
spinetoolbox.plotting	spinetoolbox.spine_db_editor.mvcmodels.frozen_table_model
module, 545	module, 255
spinetoolbox.plugin_manager	spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model
module, 555	module, 257
spinetoolbox.project	spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model
module, 557	module, 260
spinetoolbox.project_commands	spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model
module, 571	module, 261

spinetoolbox.spine_db_editor.mvcmodels.mime_type_spine module, 265
 spinetoolbox.spine_db_editor.mvcmodels.multi_db_spine module, 266
 spinetoolbox.spine_db_editor.mvcmodels.multi_db_spine module, 270
 spinetoolbox.spine_db_editor.mvcmodels.parameters_spine module, 271
 spinetoolbox.spine_db_editor.mvcmodels.parameters_spine module, 277
 spinetoolbox.spine_db_editor.mvcmodels.parameters_spine module, 278
 spinetoolbox.spine_db_editor.mvcmodels.pivot_model_spine module, 279
 spinetoolbox.spine_db_editor.mvcmodels.pivot_table_spine module, 281
 spinetoolbox.spine_db_editor.mvcmodels.scenarios_spine module, 293
 spinetoolbox.spine_db_editor.mvcmodels.scenarios_spine module, 296
 spinetoolbox.spine_db_editor.mvcmodels.single_spine module, 297
 spinetoolbox.spine_db_editor.mvcmodels.tool_favorites_spine module, 302
 spinetoolbox.spine_db_editor.mvcmodels.tool_favorites_spine module, 306
 spinetoolbox.spine_db_editor.mvcmodels.tree_items_spine module, 307
 spinetoolbox.spine_db_editor.mvcmodels.tree_model_spine module, 311
 spinetoolbox.spine_db_editor.mvcmodels.utils_spine module, 312
 spinetoolbox.spine_db_editor.scenario_generator_spine module, 402
 spinetoolbox.spine_db_editor.ui_spine module, 313
 spinetoolbox.spine_db_editor.ui.scenario_generator_spine module, 313
 spinetoolbox.spine_db_editor.ui.select_databases_spine module, 313
 spinetoolbox.spine_db_editor.ui.spine_db_editor_spine module, 313
 spinetoolbox.spine_db_editor.widgets_spine module, 314
 spinetoolbox.spine_db_editor.widgets.add_items_spine module, 314
 spinetoolbox.spine_db_editor.widgets.commit_view_spine module, 320
 spinetoolbox.spine_db_editor.widgets.custom_despines module, 321
 spinetoolbox.spine_db_editor.widgets.custom_mespin module, 333
 spinetoolbox.spine_db_editor.widgets.custom_qspine module, 335
 spinetoolbox.spine_db_editor.widgets.custom_qtableview module, 338
 spinetoolbox.spine_db_editor.widgets.custom_qtreeview module, 348
 spinetoolbox.spine_db_editor.widgets.custom_qwidgets module, 354
 spinetoolbox.spine_db_editor.widgets.edit_or_remove_items module, 355
 spinetoolbox.spine_db_editor.widgets.graph_layout_generator module, 358
 spinetoolbox.spine_db_editor.widgets.graph_view_mixin module, 359
 spinetoolbox.spine_db_editor.widgets.item_metadata_editor module, 362
 spinetoolbox.spine_db_editor.widgets.manage_items_dialogs module, 363
 spinetoolbox.spine_db_editor.widgets.mass_select_items_dialog module, 365
 spinetoolbox.spine_db_editor.widgets.metadata_editor module, 367
 spinetoolbox.spine_db_editor.widgets.multi_spine_db_editor module, 368
 spinetoolbox.spine_db_editor.widgets.object_name_list_editor module, 370
 spinetoolbox.spine_db_editor.widgets.parameter_view_mixin module, 371
 spinetoolbox.spine_db_editor.widgets.pivot_table_header_view module, 373
 spinetoolbox.spine_db_editor.widgets.scenario_generator module, 375
 spinetoolbox.spine_db_editor.widgets.select_position_parameters module, 377
 spinetoolbox.spine_db_editor.widgets.spine_db_editor module, 378
 spinetoolbox.spine_db_editor.widgets.tabular_view_header_view module, 383
 spinetoolbox.spine_db_editor.widgets.tabular_view_mixin module, 384
 spinetoolbox.spine_db_editor.widgets.tree_view_mixin module, 390
 spinetoolbox.spine_db_editor.widgets.url_toolbar module, 392
 spinetoolbox.spine_db_icon_manager module, 594
 spinetoolbox.spine_db_manager module, 596
 spinetoolbox.spine_db_parcel module, 611
 spinetoolbox.spine_db_worker module, 613
 spinetoolbox.spine_engine_manager module, 618
 spinetoolbox.spine_engine_worker module, 624

spinetoolbox.ui_main	spinetoolbox.widgets.jupyter_console_widget
module, 627	module, 453
spinetoolbox.version	spinetoolbox.widgets.kernel_editor
module, 641	module, 455
spinetoolbox.widgets	spinetoolbox.widgets.link_properties_widget
module, 402	module, 460
spinetoolbox.widgets.about_widget	spinetoolbox.widgets.map_editor
module, 402	module, 461
spinetoolbox.widgets.add_project_item_widget	spinetoolbox.widgets.map_value_editor
module, 404	module, 462
spinetoolbox.widgets.add_up_spine_opt_wizard	spinetoolbox.widgets.multi_tab_spec_editor
module, 405	module, 462
spinetoolbox.widgets.array_editor	spinetoolbox.widgets.multi_tab_window
module, 408	module, 463
spinetoolbox.widgets.array_value_editor	spinetoolbox.widgets.notification
module, 409	module, 469
spinetoolbox.widgets.code_text_edit	spinetoolbox.widgets.open_project_widget
module, 409	module, 471
spinetoolbox.widgets.commit_dialog	spinetoolbox.widgets.parameter_value_editor
module, 410	module, 474
spinetoolbox.widgets.custom_combobox	spinetoolbox.widgets.parameter_value_editor_base
module, 411	module, 474
spinetoolbox.widgets.custom_delegates	spinetoolbox.widgets.persistent_console_widget
module, 412	module, 476
spinetoolbox.widgets.custom_editors	spinetoolbox.widgets.plain_parameter_value_editor
module, 413	module, 481
spinetoolbox.widgets.custom_menus	spinetoolbox.widgets.plot_canvas
module, 417	module, 481
spinetoolbox.widgets.custom_qcombobox	spinetoolbox.widgets.plot_widget
module, 420	module, 482
spinetoolbox.widgets.custom_qgraphicsscene	spinetoolbox.widgets.plugin_manager_widgets
module, 421	module, 484
spinetoolbox.widgets.custom_qgraphicsviews	spinetoolbox.widgets.project_item_drag
module, 423	module, 485
spinetoolbox.widgets.custom_qlineedit	spinetoolbox.widgets.properties_widget
module, 426	module, 489
spinetoolbox.widgets.custom_qtableview	spinetoolbox.widgets.report_plotting_failure
module, 427	module, 490
spinetoolbox.widgets.custom_qtextbrowser	spinetoolbox.widgets.select_database_items
module, 432	module, 490
spinetoolbox.widgets.custom_qtreeview	spinetoolbox.widgets.set_description_dialog
module, 434	module, 492
spinetoolbox.widgets.custom_qwidgets	spinetoolbox.widgets.settings_widget
module, 435	module, 492
spinetoolbox.widgets.datetime_editor	spinetoolbox.widgets.statusbars
module, 443	module, 498
spinetoolbox.widgets.duration_editor	spinetoolbox.widgets.time_pattern_editor
module, 444	module, 498
spinetoolbox.widgets.indexed_value_table_contents_editor	spinetoolbox.widgets.time_series_fixed_resolution_editor
module, 444	module, 499
spinetoolbox.widgets.install_julia_wizard	spinetoolbox.widgets.time_series_variable_resolution_editor
module, 449	module, 500
spinetoolbox.widgets.jump_properties_widget	spinetoolbox.widgets.toolbars
module, 451	module, 501

- SpineToolboxCommand (class in *spinetoolbox.project_commands*), 571
- SpineToolboxProject (class in *spinetoolbox.project*), 557
- split (in module *spinetoolbox.version*), 641
- splitter_widgets() (*spinetoolbox.spine_db_editor.widgets.add_items_dialogs.MassSelectItemsDialog* method), 318
- sqlite_file_exported (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase* attribute), 378
- src_center (*spinetoolbox.link.LinkBase* property), 536
- src_rect (*spinetoolbox.link.LinkBase* property), 536
- src_rect (*spinetoolbox.link.LinkDrawerBase* property), 540
- STACKED_BAR (*spinetoolbox.plotting.PlotType* attribute), 547
- STACKED_LINE (*spinetoolbox.plotting.PlotType* attribute), 547
- StandardDBItem (class in *spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility*), 310
- StandardTreeItem (class in *spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility*), 308
- start() (*spinetoolbox.plugin_manager.PluginWorker* method), 557
- start() (*spinetoolbox.spine_engine_worker.SpineEngineWorker* method), 626
- start_detached_jupyter_console() (*spinetoolbox.ui_main.ToolboxUI* method), 639
- start_drag() (*spinetoolbox.widgets.multi_tab_window.MultiTabWindow* method), 467
- start_dragging() (*spinetoolbox.widgets.multi_tab_window.TabBarPlus* method), 468
- start_execution() (*spinetoolbox.execution_managers.ExecutionManager* method), 505
- start_execution() (*spinetoolbox.execution_managers.QProcessExecutionManager* method), 506
- start_execution() (*spinetoolbox.server.engine_client.EngineClient* method), 233
- start_execution() (*spinetoolbox.ui_main.ToolboxUI* method), 640
- start_execution() (*spinetoolbox.widgets.custom_qtextbrowser.CustomQTextBrowser* method), 433
- start_fetching_julia_kernels() (*spinetoolbox.widgets.settings_widget.SettingsWidget* method), 497
- start_fetching_python_kernels() (*spinetoolbox.widgets.settings_widget.SettingsWidget* method), 497
- start_i_julia_install_process() (*spinetoolbox.widgets.kernel_editor.KernelEditorBase* method), 457
- start_i_julia_install_python_kernel_process() (*spinetoolbox.widgets.kernel_editor.KernelEditorBase* method), 458
- start_i_julia_rebuild_process() (*spinetoolbox.widgets.kernel_editor.KernelEditorBase* method), 457
- start_kernelspec_install_process() (*spinetoolbox.widgets.kernel_editor.KernelEditorBase* method), 456
- start_package_install_process() (*spinetoolbox.widgets.kernel_editor.KernelEditorBase* method), 456
- start_relationship() (*spinetoolbox.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin* method), 361
- start_self_destruction() (*spinetoolbox.widgets.notification.Notification* method), 470
- state_storing_requested (*spinetoolbox.spine_db_editor.widgets.mass_select_items_dialogs.MassSelectItemsDialog* attribute), 366
- Status (class in *spinetoolbox.headless*), 516
- STATUSBAR_SS (in module *spinetoolbox.config*), 504
- STONEHOUSE (*spinetoolbox.server.engine_client.ClientSecurityModel* attribute), 232
- stop() (*spinetoolbox.project.SpineToolboxProject* method), 567
- stop() (*spinetoolbox.spine_db_editor.widgets.graph_layout_generator.GraphLayoutGenerator* method), 358
- stop_engine() (*spinetoolbox.spine_engine_manager.LocalSpineEngineManager* method), 620
- stop_engine() (*spinetoolbox.spine_engine_manager.RemoteSpineEngineManager* method), 622
- stop_engine() (*spinetoolbox.spine_engine_manager.SpineEngineManagerBase* method), 618
- stop_engine() (*spinetoolbox.spine_engine_worker.SpineEngineWorker* method), 626
- stop_execution() (*spinetoolbox.execution_managers.ExecutionManager* method), 505
- stop_execution() (*spinetoolbox.execution_managers.QProcessExecutionManager* method), 506

[stop_execution\(\)](#) (*spinetoolbox.server.engine_client.EngineClient* method), 233
[stop_fetcher](#) (*spinetoolbox.kernel_fetcher.KernelFetcher* attribute), 534
[stop_fetching_julia_kernels\(\)](#) (*spinetoolbox.widgets.settings_widget.SettingsWidget* method), 497
[stop_fetching_kernels\(\)](#) (*spinetoolbox.ui_main.ToolboxUI* method), 630
[stop_fetching_python_kernels\(\)](#) (*spinetoolbox.widgets.settings_widget.SettingsWidget* method), 497
[stop_invalidating_filter\(\)](#) (*spinetoolbox.spine_db_editor.mvcmodels.compound_parameter_model.CompoundParameterModel* method), 240
[stop_thread\(\)](#) (*spinetoolbox.kernel_fetcher.KernelFetcher* method), 534
[sub_model_at_row\(\)](#) (*spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel* method), 178
[sub_model_row\(\)](#) (*spinetoolbox.mvcmodels.compound_table_model.CompoundTableModel* method), 178
[submit\(\)](#) (*spinetoolbox.qthread_pool_executor.QtBasedThreadPoolExecutor* method), 590
[succeeded](#) (*spinetoolbox.plugin_manager._PluginWorker* attribute), 556
[SUCCESS](#) (*spinetoolbox.widgets.add_up_spine_opt_wizard._PageId* attribute), 406
[SUCCESS](#) (*spinetoolbox.widgets.install_julia_wizard._PageId* attribute), 450
[successfully_undone](#) (*spinetoolbox.project_commands.SpineToolboxCommand* attribute), 571
[successor_names\(\)](#) (*spinetoolbox.project.SpineToolboxProject* method), 568
[SuccessPage](#) (class in *spinetoolbox.widgets.add_up_spine_opt_wizard*), 407
[SuccessPage](#) (class in *spinetoolbox.widgets.install_julia_wizard*), 451
[suggest_db_map_codename\(\)](#) (*spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftDatabaseHeaderItem* method), 285
[supported_img_formats\(\)](#) (in module *spinetoolbox.helpers*), 521
[supportedDropActions\(\)](#) (*spinetoolbox.spine_db_editor.mvcmodels.scenario_model.ScenarioModel* method), 296
[supportedDropActions\(\)](#) (*spinetoolbox.spine_db_editor.mvcmodels.tool_feature_model.ToolFeatureModel* method), 307
[supports_specification\(\)](#) (*spinetoolbox.ui_main.ToolboxUI* method), 635
[supports_specifications\(\)](#) (*spinetoolbox.ui_main.ToolboxUI* method), 630
[system_lc_numeric\(\)](#) (in module *spinetoolbox.widgets.custom_qtableview*), 432

T

[TabBarPlus](#) (class in *spinetoolbox.widgets.multi_tab_window*), 468
[tabify_and_raise\(\)](#) (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditor* method), 382
[tabLayoutChange\(\)](#) (*spinetoolbox.widgets.multi_tab_window.TabBarPlus* method), 468
[TabularViewFilterMenu](#) (class in *spinetoolbox.spine_db_editor.widgets.custom_menus*), 335
[TabularViewHeaderWidget](#) (class in *spinetoolbox.spine_db_editor.widgets.tabular_view_header_widget*), 383
[TabularViewMixin](#) (class in *spinetoolbox.spine_db_editor.widgets.tabular_view_mixin*), 384
[take_db_map\(\)](#) (*spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem* method), 267
[take_link\(\)](#) (*spinetoolbox.widgets.custom_qgraphicsviews.DesignQGraphicsView* method), 426
[take_suggested_db_map\(\)](#) (*spinetoolbox.spine_db_editor.mvcmodels.pivot_table_models.TopLeftDatabaseHeaderItem* method), 285
[tear_down\(\)](#) (*spinetoolbox.mvcmodels.minimal_tree_model.TreeItem* method), 197
[tear_down\(\)](#) (*spinetoolbox.project.SpineToolboxProject* method), 570
[tear_down\(\)](#) (*spinetoolbox.project_item.logging_connection.LoggingConnection* method), 219
[tear_down\(\)](#) (*spinetoolbox.project_item.project_item.ProjectItem* method), 225
[tear_down\(\)](#) (*spinetoolbox.project_item.specification_editor_window.SpecificationEditorWindow* method), 230
[tear_down\(\)](#) (*spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.ObjectClassItem* method), 250

tear_down() (spinetool- box.mvcmodels.time_series_model_fixed_resolution),
 box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem
 method), 269 TimeSeriesModelVariableResolution
 tear_down() (spinetool- (class in spinetool-
 box.spine_db_editor.mvcmodels.tree_item_utility.FetchMoreMixins), 213
 method), 309
 tear_down() (spinetool- TimeSeriesVariableResolutionEditor
 box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor(class in spinetool-
 method), 383 box.widgets.time_series_variable_resolution_editor),
 tear_down() (spinetool- 500
 box.spine_db_editor.widgets.spine_db_editor.SpineDBWidgetBaseAction (class in spinetool-
 method), 381 box.widgets.custom_qwidgets), 440
 tear_down() (spinetool- to_dict() (spinetoolbox.project_item.logging_connection.LoggingConnection
 box.widgets.notification.ChangeNotifier method), 219
 method), 471 to_dict() (spinetoolbox.project_settings.ProjectSettings
 method), 583
 tear_down_recursively() (spinetool- toggle_properties_tabbar_visibility() (spine-
 box.mvcmodels.minimal_tree_model.TreeItem toolbox.ui_main.ToolboxUI method), 634
 method), 197
 teardown_process() (spinetool- toggle_selected() (spinetool-
 box.execution_managers.QProcessExecutionManager box.widgets.custom_editors.CheckListEditor
 method), 506 method), 416
 text() (spinetoolbox.widgets.custom_qwidgets.ElidedTextWidget), 436 toggle_visibility() (spinetool-
 method), 436 box.widgets.project_item_drag.ProjectItemSpecArray
 text_edited (spinetool- method), 489
 box.widgets.custom_editors._CustomLineEditDelegate tool_bar (spinetoolbox.widgets.custom_qwidgets.ToolBarWidget
 attribute), 414 attribute), 438
 TEXTBROWSER_SS (in module spinetoolbox.config), 504 tool_bar (spinetoolbox.widgets.custom_qwidgets.ToolBarWidgetAction
 thread() (spinetoolbox.spine_engine_worker.SpineEngineWorker attribute), 438
 method), 626 tool_bar (spinetoolbox.widgets.custom_qwidgets.ToolBarWidgetBase
 attribute), 438
 TIME_PATTERN (spinetool- tool_feature_item (spinetool-
 box.widgets.parameter_value_editor_base.ValueType box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureMe
 attribute), 475 property), 306
 TIME_SERIES_FIXED_RESOLUTION (spinetool- tool_tip (spinetoolbox.spine_db_editor.mvcmodels.alternative_item.Alter
 box.widgets.parameter_value_editor_base.ValueType box.widgets.parameter_value_editor_base.ValueType
 attribute), 475 property), 236
 TIME_SERIES_VARIABLE_RESOLUTION (spinetool- tool_tip (spinetoolbox.spine_db_editor.mvcmodels.scenario_item.Scenari
 box.widgets.parameter_value_editor_base.ValueType property), 295
 attribute), 475 tool_tip (spinetoolbox.spine_db_editor.mvcmodels.scenario_item.Scenari
 property), 294
 TimeoutError, 589
 TimePatternEditor (class in spinetool- tool_tip (spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.Fear
 box.widgets.time_pattern_editor), 498 property), 303
 TimePatternModel (class in spinetool- tool_tip (spinetoolbox.spine_db_editor.mvcmodels.tool_feature_item.Tool
 box.mvcmodels.time_pattern_model), 210 property), 304
 timerEvent() (spinetool- tool_tip (spinetoolbox.spine_db_editor.mvcmodels.tree_item_utility.Standa
 box.widgets.multi_tab_window.MultiTabWindow property), 308
 method), 467 tool_tip_data_from_parsed() (spinetool-
 TimeSeriesFixedResolutionEditor box.spine_db_manager.SpineDBManager
 (class in spinetool- static method), 603
 box.widgets.time_series_fixed_resolution_editor), 499
 TimeSeriesFixedResolutionTableView (class in spinetoolbox.widgets.custom_qtableview), 429
 TimeSeriesModelFixedResolution
 (class in spinetool-
 box.widgets.custom_qwidgets), 438
 TimeSeriesModelVariableResolution
 (class in spinetool-
 box.widgets.custom_qwidgets), 438
 TimeSeriesVariableResolutionEditor
 (class in spinetool-
 box.widgets.time_series_variable_resolution_editor), 500
 Toolbox (class in spinetoolbox.widgets.toolbars), 501
 ToolboxWidget (class in spinetool-
 box.widgets.custom_qwidgets), 438
 ToolboxWidgetAction (class in spinetool-
 box.widgets.custom_qwidgets), 438
 ToolboxWidgetBase (class in spinetool-

[box.widgets.custom_qwidgets\), 438](#)
[toolbox \(spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase](#)
[property\), 378](#)
[toolbox \(spinetoolbox.widgets.add_project_item_widget.AddProjectItemWidget](#)
[attribute\), 404](#)
[toolbox\(\) \(spinetoolbox.project.SpineToolboxProject](#)
[method\), 559](#)
[ToolboxUI \(class in spinetoolbox.ui_main\), 627](#)
[ToolFeatureDelegate \(class in spinetool-](#)
[box.spine_db_editor.widgets.custom_delegates\), 328](#)
[ToolFeatureLeafItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.tool_feature_item\), 304](#)
[ToolFeatureMethodLeafItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.tool_feature_item\), 306](#)
[ToolFeatureMethodRootItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.tool_feature_item\), 305](#)
[ToolFeatureModel \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.tool_feature_model\), 306](#)
[ToolFeatureRequiredItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.tool_feature_item\), 305](#)
[ToolFeatureRootItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.tool_feature_item\), 304](#)
[ToolFeatureTreeView \(class in spinetool-](#)
[box.spine_db_editor.widgets.custom_qtreeview\), 351](#)
[ToolLeafItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.tool_feature_item\), 304](#)
[ToolRootItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.tool_feature_item\), 303](#)
[top_left_id\(\) \(spinetool-](#)
[box.spine_db_editor.mvcmodels.pivot_table_model](#)
[method\), 288](#)
[top_left_indexes\(\) \(spinetool-](#)
[box.spine_db_editor.mvcmodels.pivot_table_model](#)
[method\), 287](#)
[TopLeftAlternativeHeaderItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.pivot_table_model\), 283](#)
[TopLeftDatabaseHeaderItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.pivot_table_models\), 284](#)
[TopLeftHeaderItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.pivot_table_model\), 281](#)
[TopLeftObjectHeaderItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.pivot_table_models\), 283](#)
[TopLeftParameterHeaderItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.pivot_table_models\), 283](#)
[TopLeftParameterIndexHeaderItem](#)
[\(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.pivot_table_models\), 283](#)
[TopLeftScenarioHeaderItem \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.pivot_table_models\), 284](#)
[TOTAL_FAILURE \(spinetool-](#)
[box.widgets.add_up_spine_opt_wizard._PageId](#)
[attribute\), 406](#)
[TotalFailurePage \(class in spinetool-](#)
[box.widgets.add_up_spine_opt_wizard\), 408](#)
[traitslets_logger \(in module spinetool-](#)
[box.widgets.jupyter_console_widget\), 453](#)
[TransparentIconEngine \(class in spinetool-](#)
[box.helpers\), 523](#)
[tree_built \(spinetool-](#)
[box.mvcmodels.resource_filter_model.ResourceFilterModel](#)
[attribute\), 208](#)
[TreeItem \(class in spinetool-](#)
[box.mvcmodels.minimal_tree_model\), 196](#)
[TreeModelBase \(class in spinetool-](#)
[box.spine_db_editor.mvcmodels.tree_model_base\), 311](#)
[TreeNode \(class in spinetoolbox.plotting\), 547](#)
[TreeViewMixin \(class in spinetool-](#)
[box.spine_db_editor.widgets.tree_view_mixin\), 390](#)
[trigger\(\) \(spinetoolbox.helpers.SignalWaiter method\), 530](#)
[trim_columns\(\) \(spinetool-](#)
[box.mvcmodels.map_model.MapModel](#)
[method\), 192](#)
[TROUBLESHOOT PROBLEMS \(spinetool-](#)
[box.widgets.add_up_spine_opt_wizard._PageId](#)
[attribute\), 406](#)
[TROUBLESHOOT SOLUTION \(spinetool-](#)
[box.widgets.add_up_spine_opt_wizard._PageId](#)
[attribute\), 406](#)
[TroubleshootProblemsPage \(class in spinetool-](#)
[box.widgets.add_up_spine_opt_wizard\), 407](#)
[TroubleshootSolutionPage \(class in spinetool-](#)
[box.widgets.add_up_spine_opt_wizard\), 407](#)
[try_number_from_string\(\) \(in module spinetool-](#)
[box.helpers\), 525](#)
[tuple_itemgetter\(\) \(in module spinetoolbox.helpers\), 522](#)
[turn_node_to_xy_data\(\) \(in module spinetool-](#)

box.plotting), 548

twinned_axes() (*spinetoolbox.widgets.plot_canvas.PlotCanvas* method), 482

two_column_as_csv() (in module *spinetoolbox.spine_db_editor.mvcmodels.utils*), 312

U

Ui_Form (class in *spinetoolbox.spine_db_editor.ui.scenario_generator*), 313

Ui_Form (class in *spinetoolbox.spine_db_editor.ui.select_databases*), 313

Ui_MainWindow (class in *spinetoolbox.spine_db_editor.ui.spine_db_editor_window*), 313

undo() (*spinetoolbox.project_commands.AddConnectionCommand* method), 574

undo() (*spinetoolbox.project_commands.AddJumpCommand* method), 574

undo() (*spinetoolbox.project_commands.AddProjectItemsCommand* method), 572

undo() (*spinetoolbox.project_commands.AddSpecificationCommand* method), 576

undo() (*spinetoolbox.project_commands.MoveIconCommand* method), 572

undo() (*spinetoolbox.project_commands.RemoveAllProjectItemsCommand* method), 573

undo() (*spinetoolbox.project_commands.RemoveConnectionsCommand* method), 574

undo() (*spinetoolbox.project_commands.RemoveJumpsCommand* method), 574

undo() (*spinetoolbox.project_commands.RemoveProjectItemsCommand* method), 573

undo() (*spinetoolbox.project_commands.RemoveSpecificationCommand* method), 577

undo() (*spinetoolbox.project_commands.RenameProjectItemCommand* method), 573

undo() (*spinetoolbox.project_commands.ReplaceSpecificationCommand* method), 576

undo() (*spinetoolbox.project_commands.SaveSpecificationCommand* method), 577

undo() (*spinetoolbox.project_commands.SetConnectionDefaultFilterOnlineStatus* method), 576

undo() (*spinetoolbox.project_commands.SetConnectionOptionsCommand* method), 576

undo() (*spinetoolbox.project_commands.SetFiltersOnlineCommand* method), 575

undo() (*spinetoolbox.project_commands.SetItemSpecificationCommand* method), 572

undo() (*spinetoolbox.project_commands.SetJumpConditionCommand* method), 575

undo() (*spinetoolbox.project_commands.SetProjectDescriptionCommand* method), 572

undo() (*spinetoolbox.project_commands.UpdateJumpCmdLineArgsCommand* method), 575

undo() (*spinetoolbox.project_item.specification_editor_window.ChangeSpec* method), 229

undo() (*spinetoolbox.spine_db_commands.AddItemCommand* method), 593

undo() (*spinetoolbox.spine_db_commands.AgedUndoCommand* method), 591

undo() (*spinetoolbox.spine_db_commands.RemoveItemsCommand* method), 594

undo() (*spinetoolbox.spine_db_commands.SpineDBMacro* method), 591

undo() (*spinetoolbox.spine_db_commands.UpdateItemsCommand* method), 593

undo_age (*spinetoolbox.spine_db_commands.AgedUndoStack* property), 590

undo_critical_commands() (*spinetoolbox.ui_main.ToolboxUI* method), 631

undo_specification() (*spinetoolbox.project_item.project_item.ProjectItem* method), 221

UndoRedoMixin (class in *spinetoolbox.widgets.custom_qwidgets*), 436

unique_alternatives() (in module *spinetoolbox.spine_db_editor.scenario_generation*), 402

unique_name() (in module *spinetoolbox.helpers*), 528

unregister_listener() (*spinetoolbox.spine_db_manager.SpineDBManager* method), 600

unset_item() (*spinetoolbox.widgets.properties_widget.PropertiesWidgetBase* method), 489

unset_link() (*spinetoolbox.widgets.jump_properties_widget.JumpPropertiesWidget* method), 452

unset_link() (*spinetoolbox.widgets.link_properties_widget.LinkPropertiesWidget* method), 460

update() (*spinetoolbox.mvcmodels.file_list_models.FileListModel* method), 183

update() (*spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.EntityTreeItem* method), 252

update() (*spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem* method), 269

update() (*spinetoolbox.widgets.project_item_drag._ChoppedIcon* method), 487

update() (*spinetoolbox.widgets.project_item_drag._ChoppedIconEngine* method), 487

update() (*spinetoolbox.widgets.project_item_drag.ProjectItemSpecArray* method), 488

update_actions_availability() (*spinetool-*

<code>box.spine_db_editor.mvcmodels.parameter_value_list_item.ValueList</code>	<code>box.spine_db_editor.graphics_items.ArcItem</code>
<code>method), 278</code>	<code>method), 399</code>
<code>update_item_in_db()</code>	<code>(spinetool- update_links_geometry() (spinetool-</code>
<code>box.spine_db_editor.mvcmodels.scenario_item.ScenarioItem</code>	<code>box.project_item_icon.ProjectItemIcon</code>
<code>method), 295</code>	<code>method), 579</code>
<code>update_item_in_db()</code>	<code>(spinetool- update_links_geometry() (spinetool-</code>
<code>box.spine_db_editor.mvcmodels.scenario_item.ScenarioItem</code>	<code>box.widgets.settings_widget.SettingsWidget</code>
<code>method), 294</code>	<code>method), 496</code>
<code>update_item_in_db()</code>	<code>(spinetool- update_list_values() (spinetool-</code>
<code>box.spine_db_editor.mvcmodels.tool_feature_item.FeatureList</code>	<code>box.spine_db_manager.SpineDBManager</code>
<code>method), 303</code>	<code>method), 608</code>
<code>update_item_in_db()</code>	<code>(spinetool- update_metadata() (spinetool-</code>
<code>box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureList</code>	<code>box.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel</code>
<code>method), 305</code>	<code>method), 261</code>
<code>update_item_in_db()</code>	<code>(spinetool- update_metadata() (spinetool-</code>
<code>box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureList</code>	<code>box.spine_db_manager.SpineDBManager</code>
<code>method), 306</code>	<code>method), 608</code>
<code>update_item_in_db()</code>	<code>(spinetool- update_model() (spinetool-</code>
<code>box.spine_db_editor.mvcmodels.tool_feature_item.ToolFeatureList</code>	<code>box.spine_db_editor.mvcmodels.pivot_model.PivotModel</code>
<code>method), 304</code>	<code>method), 279</code>
<code>update_item_in_db()</code>	<code>(spinetool- update_model() (spinetool-</code>
<code>box.spine_db_editor.mvcmodels.tree_item_utility.LeafItem</code>	<code>box.spine_db_editor.mvcmodels.pivot_table_models.PivotTableModel</code>
<code>method), 310</code>	<code>method), 286</code>
<code>update_item_metadata()</code>	<code>(spinetool- update_name() (spinetool-</code>
<code>box.spine_db_editor.mvcmodels.item_metadata_table_model.ItemMetadataTableModel</code>	<code>box.spine_db_editor.mvcmodels.graphics_items.ObjectItem</code>
<code>method), 259</code>	<code>method), 398</code>
<code>update_items()</code>	<code>(spinetool- update_name_item() (spinetool-</code>
<code>box.spine_db_manager.SpineDBManager</code>	<code>box.project_item_icon.ProjectItemIcon</code>
<code>method), 605</code>	<code>method), 578</code>
<code>update_items()</code>	<code>(spinetool- update_name_label() (spinetool-</code>
<code>box.spine_db_worker.SpineDBWorker</code>	<code>box.project_item.project_item.ProjectItem</code>
<code>method), 616</code>	<code>method), 225</code>
<code>update_items_in_cache()</code>	<code>(spinetool- update_object_classes() (spinetool-</code>
<code>box.spine_db_manager.SpineDBManager</code>	<code>box.spine_db_manager.SpineDBManager</code>
<code>method), 598</code>	<code>method), 607</code>
<code>update_items_in_db()</code>	<code>(spinetool- update_objects() (spinetool-</code>
<code>box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel</code>	<code>box.spine_db_manager.SpineDBManager</code>
<code>method), 300</code>	<code>method), 607</code>
<code>update_items_in_db()</code>	<code>(spinetool- update_opacity() (spinetool-</code>
<code>box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel</code>	<code>box.spine_db_editor.mvcmodels.notification.Notification</code>
<code>method), 299</code>	<code>method), 470</code>
<code>update_items_in_db()</code>	<code>(spinetool- update_parameter_definitions() (spinetool-</code>
<code>box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel</code>	<code>box.spine_db_editor.mvcmodels.spine_db_manager.SpineDBManager</code>
<code>method), 301</code>	<code>method), 608</code>
<code>update_items_in_db()</code>	<code>(spinetool- update_parameter_value_lists() (spinetool-</code>
<code>box.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel</code>	<code>box.spine_db_editor.mvcmodels.spine_db_manager.SpineDBManager</code>
<code>method), 302</code>	<code>method), 608</code>
<code>update_items_path()</code>	<code>(spinetool- update_parameter_value_metadata() (spinetool-</code>
<code>box.widgets.settings_widget.SettingsWidget</code>	<code>box.spine_db_manager.SpineDBManager</code>
<code>method), 496</code>	<code>method), 609</code>
<code>update_jump()</code>	<code>(spinetool- update_parameter_values() (spinetool-</code>
<code>box.project.SpineToolboxProject</code>	<code>box.spine_db_manager.SpineDBManager</code>
<code>method), 565</code>	<code>method), 608</code>
<code>update_line()</code>	<code>(spinetool- update_path() (spinetool-</code>

- box.project_item_icon.ConnectorButton*
method), 580
- `update_path()` (*spinetool-*
box.project_item_icon.ProjectItemIcon
method), 578
- `update_path()` (*spinetool-*
box.project_item_icon.RankIcon
method), 582
- `update_properties_ui()` (*spinetool-*
box.ui_main.ToolboxUI method), 631
- `update_recent_projects()` (*spinetool-*
box.ui_main.ToolboxUI method), 637
- `update_recents()` (*spinetool-*
box.widgets.open_project_widget.OpenProjectDialog
static method), 472
- `update_relationship_classes()` (*spinetool-*
box.spine_db_manager.SpineDBManager
method), 607
- `update_relationships()` (*spinetool-*
box.spine_db_manager.SpineDBManager
method), 607
- `update_scenarios()` (*spinetool-*
box.spine_db_manager.SpineDBManager
method), 607
- `update_scene_bg()` (*spinetool-*
box.widgets.settings_widget.SettingsWidget
method), 496
- `update_tool_feature_methods()` (*spinetool-*
box.spine_db_manager.SpineDBManager
method), 608
- `update_tool_features()` (*spinetool-*
box.spine_db_manager.SpineDBManager
method), 608
- `update_tools()` (*spinetool-*
box.spine_db_manager.SpineDBManager
method), 608
- `update_ui()` (*spinetool-*
box.widgets.settings_widget.SettingsWidget
method), 496
- `update_ui()` (*spinetool-*
box.widgets.settings_widget.SettingsWidgetBase
method), 493
- `update_ui()` (*spinetool-*
box.widgets.settings_widget.SpineDBEditorSettingsWidget
method), 494
- `update_ui_and_close()` (*spinetool-*
box.widgets.settings_widget.SettingsWidgetBase
method), 493
- `update_undo_redo_actions()` (*spinetool-*
box.spine_db_editor.widgets.spine_db_editor.SpineDBEditor
method), 379
- `update_window_modified()` (*spinetool-*
box.ui_main.ToolboxUI method), 628
- `update_window_title()` (*spinetool-*
box.ui_main.ToolboxUI method), 628
- `updateEditorGeometry()` (*spinetool-*
box.spine_db_editor.widgets.custom_delegates.AlternativeDelegates
method), 330
- `updateEditorGeometry()` (*spinetool-*
box.spine_db_editor.widgets.custom_delegates.ManageItemsDelegates
method), 331
- `updateEditorGeometry()` (*spinetool-*
box.spine_db_editor.widgets.custom_delegates.ParameterDelegates
method), 325
- `updateEditorGeometry()` (*spinetool-*
box.spine_db_editor.widgets.custom_delegates.PivotTableDelegates
method), 322
- `updateEditorGeometry()` (*spinetool-*
box.spine_db_editor.widgets.custom_delegates.ScenarioDelegates
method), 330
- `updateEditorGeometry()` (*spinetool-*
box.spine_db_editor.widgets.custom_delegates.ToolFeatureDelegates
method), 329
- `updateEditorGeometry()` (*spinetool-*
box.spine_db_editor.widgets.object_name_list_editor.SearchBarDialog
method), 370
- `updateEditorGeometry()` (*spinetool-*
box.spine_db_editor.widgets.select_position_parameters_dialog.SelectPositionParametersDialog
method), 377
- `updateEditorGeometry()` (*spinetool-*
box.widgets.custom_delegates.ComboBoxDelegate
method), 412
- `UpdateItemsCommand` (class in *spinetool-*
box.spine_db_commands), 593
- `UpdateJumpCmdLineArgsCommand` (class in *spinetool-*
box.project_commands), 575
- `upgrade()` (*spinetoolbox.project_upgrader.ProjectUpgrader*
method), 584
- `upgrade_to_latest()` (*spinetool-*
box.project_upgrader.ProjectUpgrader
method), 584
- `upgrade_v10_to_v11()` (*spinetool-*
box.project_upgrader.ProjectUpgrader static
method), 587
- `upgrade_v1_to_v2()` (*spinetool-*
box.project_item.project_item.ProjectItem
static method), 225
- `upgrade_v1_to_v2()` (*spinetool-*
box.project_upgrader.ProjectUpgrader static
method), 584
- `upgrade_v2_to_v3()` (*spinetool-*
box.project_item.project_item.ProjectItem
static method), 225
- `upgrade_v2_to_v3()` (*spinetool-*
box.project_upgrader.ProjectUpgrader
method), 585
- `upgrade_v3_to_v4()` (*spinetool-*
box.project_upgrader.ProjectUpgrader static
method), 585

- method*), 585
 - `upgrade_v4_to_v5()` (*spinetoolbox.project_upgrader.ProjectUpgrader static method*), 585
 - `upgrade_v5_to_v6()` (*spinetoolbox.project_upgrader.ProjectUpgrader static method*), 585
 - `upgrade_v6_to_v7()` (*spinetoolbox.project_upgrader.ProjectUpgrader static method*), 586
 - `upgrade_v7_to_v8()` (*spinetoolbox.project_upgrader.ProjectUpgrader static method*), 586
 - `upgrade_v8_to_v9()` (*spinetoolbox.project_upgrader.ProjectUpgrader static method*), 586
 - `upgrade_v9_to_v10()` (*spinetoolbox.project_upgrader.ProjectUpgrader static method*), 587
 - `upload_project()` (*spinetoolbox.server.engine_client.EngineClient method*), 232
 - `upstream_resources_updated()` (*spinetoolbox.project_item.project_item.ProjectItem method*), 222
 - `UrlToolBar` (class in *spinetoolbox.spine_db_editor.widgets.url_toolbar*), 392
 - `use_as_window()` (*spinetoolbox.widgets.plot_widget.PlotWidget method*), 484
- ## V
- `V_MARGIN` (*spinetoolbox.widgets.custom_qwidgets.TitleWidget attribute*), 440
 - `vacuum()` (*spinetoolbox.spine_db_editor.widgets.spine_db_editor.SpineDBEditorBase method*), 379
 - `validate()` (*spinetoolbox.widgets.open_project_widget.DirValidator method*), 473
 - `validate_project_item_name()` (*spinetoolbox.project.SpineToolboxProject method*), 563
 - `validator_state_changed()` (*spinetoolbox.widgets.open_project_widget.OpenProjectDialog method*), 471
 - `value` (*spinetoolbox.mvcmodels.indexed_value_table_model.IndexedValueTableModel property*), 187
 - `VALUE` (*spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model.ItemType attribute*), 258
 - `VALUE` (*spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel attribute*), 262
 - `value()` (*spinetoolbox.mvcmodels.map_model.MapModel method*), 192
 - `value()` (*spinetoolbox.widgets.array_editor.ArrayEditor method*), 408
 - `value()` (*spinetoolbox.widgets.custom_qwidgets.HorizontalSpinBox method*), 441
 - `value()` (*spinetoolbox.widgets.datetime_editor.DatetimeEditor method*), 443
 - `value()` (*spinetoolbox.widgets.duration_editor.DurationEditor method*), 444
 - `value()` (*spinetoolbox.widgets.map_editor.MapEditor method*), 461
 - `value()` (*spinetoolbox.widgets.plain_parameter_value_editor.PlainParameterEditor method*), 481
 - `value()` (*spinetoolbox.widgets.time_pattern_editor.TimePatternEditor method*), 499
 - `value()` (*spinetoolbox.widgets.time_series_fixed_resolution_editor.TimeSeriesFixedResolutionEditor method*), 500
 - `value()` (*spinetoolbox.widgets.time_series_variable_resolution_editor.TimeSeriesVariableResolutionEditor method*), 500
 - `value_column_header` (*spinetoolbox.spine_db_editor.widgets.custom_qtableview.ParameterDefinitionTableWidget attribute*), 341
 - `value_column_header` (*spinetoolbox.spine_db_editor.widgets.custom_qtableview.ParameterTableWidget attribute*), 339
 - `value_column_header` (*spinetoolbox.spine_db_editor.widgets.custom_qtableview.ParameterValueTableWidget attribute*), 341
 - `value_editor_requested` (*spinetoolbox.spine_db_editor.widgets.custom_delegates.ParameterValueEditorDelegate attribute*), 324
 - `value_field` (*spinetoolbox.spine_db_editor.mvcmodels.empty_parameter_models.EmptyParameterModel property*), 245
 - `value_field` (*spinetoolbox.spine_db_editor.mvcmodels.single_parameter_models.SingleParameterModel property*), 248
 - `valueChanged` (*spinetoolbox.widgets.custom_qwidgets.HorizontalSpinBox attribute*), 441
 - `ValueItem` (class in *spinetoolbox.spine_db_editor.mvcmodels.parameter_value_list_item*), 278
 - `ValueListDelegate` (class in *spinetoolbox.spine_db_editor.widgets.custom_delegates*), 327
 - `values` (*spinetoolbox.mvcmodels.time_series_model_fixed_resolution.TimeSeriesFixedResolutionModel property*), 187
 - `values` (*spinetoolbox.mvcmodels.time_series_model_variable_resolution.TimeSeriesVariableResolutionModel property*), 187
 - `ValueType` (class in *spinetoolbox.spine_db_editor.mvcmodels.item_metadata_table_model.ItemType*), 258
 - `ValueType` (class in *spinetoolbox.spine_db_editor.mvcmodels.metadata_table_model.MetadataTableModel*), 262
 - `VersionInfo` (class in *spinetoolbox.version*), 641
 - `VERTEX_EXTENT` (*spinetoolbox.widgets.array_editor.ArrayEditor attribute*), 408

box.spine_db_editor.widgets.graph_view_mixin.GraphViewMixin (class in spinetoolbox.spine_db_editor.widgets.graph_view_mixin), 420
 attribute), 359
 verticalScrollbarValueChanged() (spinetoolbox.spine_db_editor.widgets.custom_qtreeview.EntityTreeViewMixin method), 388
 method), 349
 visit_all() (spinetoolbox.spine_db_editor.widgets.custom_qwidgets), 440
 box.mvcmodels.minimal_tree_model.MinimalTreeModel (class in spinetoolbox.mvcmodels.minimal_tree_model), 198
 visual_key (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.MemberObjectItem attribute), 253
 box.spine_db_editor.mvcmodels.entity_tree_item.RelationshipClassItem (class in spinetoolbox.mvcmodels.entity_tree_item), 251
 visual_key (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.RelationshipClassItem attribute), 251
 box.spine_db_editor.mvcmodels.entity_tree_item.RelationshipItem (class in spinetoolbox.mvcmodels.entity_tree_item), 253
 visual_key (spinetoolbox.spine_db_editor.mvcmodels.entity_tree_item.RelationshipItem attribute), 253
 box.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem (class in spinetoolbox.mvcmodels.multi_db_tree_item), 267
 attribute), 267
 XYData (class in spinetoolbox.plotting), 547
 Y (class in spinetoolbox.plotting), 547
 y (spinetoolbox.plotting.XYData attribute), 547
 y (spinetoolbox.project_item.project_item.ProjectItem attribute), 220
 y (spinetoolbox.widgets.add_project_item_widget.AddProjectItemWidget attribute), 404
 y_label (spinetoolbox.plotting.XYData attribute), 547
 yield_formats() (spinetoolbox.helpers.CustomSyntaxHighlighter method), 531
 Z (class in spinetoolbox.widgets.custom_qgraphicsviews), 423
 zoom_factor (spinetoolbox.widgets.custom_qgraphicsviews.CustomQGraphicsView property), 423
 zoom_in() (spinetoolbox.widgets.custom_qgraphicsviews.CustomQGraphicsView method), 424
 zoom_out() (spinetoolbox.widgets.custom_qgraphicsviews.CustomQGraphicsView method), 424
 will_have_children_change() (spinetoolbox.fetch_parent.FetchParent method), 508
 will_have_children_change() (spinetoolbox.fetch_parent.FlexibleFetchParent method), 511
 will_have_children_change() (spinetoolbox.spine_db_editor.mvcmodels.multi_db_tree_item.MultiDBTreeItem method), 268
 wipe_out() (spinetoolbox.link._SvgIcon method), 537
 wipe_out() (spinetoolbox.link._TextIcon method), 537
 wipe_out() (spinetoolbox.link.JumpOrLink method), 538
 wipe_out() (spinetoolbox.link.LinkBase method), 537
 wipe_out() (spinetoolbox.widgets.custom_menus.FilterMenuBase method), 537