
ViUR Vi

Release 3.0

Tilman Oestereich, Andreas H. Kelch

May 20, 2022

1 About	3
Python Module Index	79
Index	81



Administrativ viusal interface for the ViUR Information System

ABOUT

soon...

1.1 Getting started

1.1.1 Systemrequirements

soon...

1.1.2 Setup and Installation

Linux or WSL

soon...

Mac OS

soon...

1.2 Reference Guide

1.2.1 Overview

soon...

1.2.2 Navigation

soon...

1.2.3 Handlers

soon...

1.2.4 Actionbars

soon...

1.3 Tutorials

1.3.1 Create a ActionBar Plugin

soon...

1.3.2 Create a Handler Plugin

soon...

1.3.3 Create a Navigation Plugin

soon...

1.4 API Reference

This page contains auto-generated API reference documentation¹.

1.4.1 vi

Subpackages

`vi.actions`

Submodules

`vi.actions.context`

Module Contents

Classes

ContextAction

¹ Created with sphinx-autoapi


```

class vi.actions.context.ContextAction(module, handler, actionName, *args, **kwargs)
    Bases: flare.button.Button

    onAttach(self)

    onDetach(self)

    onSelectionChanged(self, table, selection, *args, **kwargs)

    onClick(self, sender=None)

    openModule(self, data, title=None)

    static isSuitableFor(module, handler, actionName)

```

`vi.actions.edit`

Module Contents

Classes

[*SaveContinue*](#)

[*SaveSingleton*](#)

[*ExecuteSingleton*](#)

[*SaveClose*](#)

[*Refresh*](#)

[*CancelClose*](#)

```

class vi.actions.edit.SaveContinue(*args, **kwargs)
    Bases: flare.button.Button

    static isSuitableFor(module, handler, actionName)

    onClick(self, sender=None)

    resetLoadingState(self)

```

```

class vi.actions.edit.SaveSingleton(*args, **kwargs)
    Bases: flare.button.Button

    static isSuitableFor(module, handler, actionName)

    onClick(self, sender=None)

    resetLoadingState(self)

```

```

class vi.actions.edit.ExecuteSingleton(*args, **kwargs)
    Bases: flare.button.Button

```

```
    static isSuitableFor(module, handler, actionName)

    onClick(self, sender=None)

    resetLoadingState(self)

class vi.actions.edit.SaveClose(*args, **kwargs)
    Bases: flare.button.Button

    static isSuitableFor(module, handler, actionName)

    onClick(self, sender=None)

    resetLoadingState(self)

class vi.actions.edit.Refresh(*args, **kwargs)
    Bases: flare.button.Button

    static isSuitableFor(module, handler, actionName)

    onClick(self, sender=None)

    performReload(self, sender=None)

    resetLoadingState(self)

class vi.actions.edit.CancelClose(*args, **kwargs)
    Bases: flare.button.Button

    static isSuitableFor(module, handler, actionName)

    onClick(self, sender=None)

    resetLoadingState(self)
```

vi.actions.file

Module Contents

Classes

<i>FileSelectUploader</i>	Small wrapper around <input type="file">.
<i>AddNodeAction</i>	Adds a new directory to a tree.simple application.
<i>AddLeafAction</i>	Allows uploading of files using the file dialog.
<i>EditAction</i>	Provides editing in a tree.simple application.
<i>DownloadAction</i>	Allows downloading files from the server.

```
class vi.actions.file.FileSelectUploader(*args, **kwargs)
    Bases: flare.html5.Input

    Small wrapper around <input type="file">. Creates the element; executes the click (=opens the file dialog); runs
    the callback if a file has been selected and removes itself from its parent.

    onChange(self, event)
```

```

class vi.actions.file.AddNodeAction(*args, **kwargs)
    Bases: flare.button.Button
    Adds a new directory to a tree.simple application.
    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    createDir(self, dialog, dirName)
    onMkDir(self, req)
    resetLoadingState(self)

class vi.actions.file.AddLeafAction(*args, **kwargs)
    Bases: flare.button.Button
    Allows uploading of files using the file dialog.
    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    resetLoadingState(self)

class vi.actions.file.EditAction(*args, **kwargs)
    Bases: flare.button.Button
    Provides editing in a tree.simple application. If a directory is selected, it opens a dialog for renaming that direc-
    tory, otherwise the full editWidget is used.
    onAttach(self)
    onDetach(self)
    onSelectionActivated(self, table, selection)
    onSelectionChanged(self, table, selection, *args, **kwargs)
    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    editDir(self, dialog, dirName)
    resetLoadingState(self)

class vi.actions.file.DownloadAction(*args, **kwargs)
    Bases: flare.button.Button
    Allows downloading files from the server.
    onAttach(self)
    onDetach(self)
    onSelectionChanged(self, table, selection, *args, **kwargs)
    static isSuitableFor(module, handler, actionName)

```

```
onClick(self, sender=None)

disableViUnloadingWarning(self, *args, **kwargs)

enableViUnloadingWarning(self, *args, **kwargs)

doDownload(self, fileData)

resetLoadingState(self)
```

vi.actions.hierarchy

Module Contents

Classes

<i>AddAction</i>	Adds a new node in a hierarchy application.
<i>EditAction</i>	Edits a node in a hierarchy application.
<i>CloneAction</i>	Allows cloning an entry (including its subentries) in a hierarchy application.
<i>DeleteAction</i>	Deletes a node from a hierarchy application.
<i>ReloadAction</i>	Allows adding an entry in a list-module.
<i>SelectRootNode</i>	Selector for hierarchy root nodes.
<i>ListViewAction</i>	Allows adding an entry in a list-module.

```
class vi.actions.hierarchy.AddAction(*args, **kwargs)
```

```
    Bases: flare.button.Button
```

```
    Adds a new node in a hierarchy application.
```

```
    static isSuitableFor(module, handler, actionName)
```

```
    onClick(self, sender=None)
```

```
    resetLoadingState(self)
```

```
class vi.actions.hierarchy.EditAction(*args, **kwargs)
```

```
    Bases: flare.button.Button
```

```
    Edits a node in a hierarchy application.
```

```
    onAttach(self)
```

```
    onDetach(self)
```

```
    onSelectionChanged(self, table, selection, *args, **kwargs)
```

```
    onSelectionActivated(self, table, selection)
```

```
    static isSuitableFor(module, handler, actionName)
```

```
    onClick(self, sender=None)
```

```
    openEditor(self, key)
```

resetLoadingState(*self*)

class vi.actions.hierarchy.CloneAction(*args, **kwargs)

Bases: flare.button.Button

Allows cloning an entry (including its subentries) in a hierarchy application.

onAttach(*self*)

onDetach(*self*)

onSelectionChanged(*self*, *table*, *selection*, *args, **kwargs)

static isSuitableFor(*module*, *handler*, *actionName*)

onClick(*self*, *sender*=None)

openEditor(*self*, *key*)

resetLoadingState(*self*)

class vi.actions.hierarchy.DeleteAction(*args, **kwargs)

Bases: flare.button.Button

Deletes a node from a hierarchy application.

onAttach(*self*)

onDetach(*self*)

onSelectionChanged(*self*, *table*, *selection*, *args, **kwargs)

static isSuitableFor(*module*, *handler*, *actionName*)

onClick(*self*, *sender*=None)

doDelete(*self*, *dialog*)

allDeletedSuccess(*self*, *success*)

resetLoadingState(*self*)

class vi.actions.hierarchy.ReloadAction(*args, **kwargs)

Bases: flare.button.Button

Allows adding an entry in a list-module.

static isSuitableFor(*module*, *handler*, *actionName*)

onClick(*self*, *sender*=None)

resetLoadingState(*self*)

class vi.actions.hierarchy.SelectRootNode(*module*, *handler*, *actionName*, *args, **kwargs)

Bases: flare.html5.Select

Selector for hierarchy root nodes.

onAttach(*self*)

onDetach(*self*)

update(*self*)

onRootNodeChanged(*self*, *newNode*, **args*, ***kwargs*)

onRootNodesAvailable(*self*, *req*)

onChange(*self*, *event*)

static isSuitableFor(*module*, *handler*, *actionName*)

class **vi.actions.hierarchy.ListViewAction**(**args*, ***kwargs*)

Bases: `flare.button.Button`

Allows adding an entry in a list-module.

static isSuitableFor(*module*, *handler*, *actionName*)

onClick(*self*, *sender=None*)

resetLoadingState(*self*)

vi.actions.list

Module Contents

Classes

<i>AddAction</i>	Allows adding an entry in a list-module.
<i>EditAction</i>	Allows editing an entry in a list-module.
<i>CloneAction</i>	Allows cloning an entry in a list-module.
<i>DeleteAction</i>	Allows deleting an entry in a list-module.
<i>ListPreviewAction</i>	
<i>ListPreviewInlineAction</i>	
<i>CloseAction</i>	
<i>SelectAction</i>	
<i>SelectFieldsPopup</i>	
<i>SelectFieldsAction</i>	
<i>ReloadAction</i>	Allows Reloading
<i>TableNextPage</i>	
<i>TablePrevPage</i>	
<i>TableItems</i>	
<i>SetPageRowAmountAction</i>	Load a bunch of pages
<i>LoadNextBatchAction</i>	Load a bunch of pages
<i>LoadAllAction</i>	Allows Loading all Entries in a list
<i>PageFindAction</i>	Allows Loading all Entries in a list
<i>ListSelectFilterAction</i>	
<i>CreateRecurrentAction</i>	
<i>ExportCsvAction</i>	
<i>SelectAllAction</i>	
<i>UnSelectAllAction</i>	
<i>SelectInvertAction</i>	

```

class vi.actions.list.AddAction(*args, **kwargs)
    Bases: flare.button.Button
    Allows adding an entry in a list-module.
    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    resetLoadingState(self)

```

```
class vi.actions.list.EditAction(*args, **kwargs)
    Bases: flare.button.Button
    Allows editing an entry in a list-module.
    onAttach(self)
    onDetach(self)
    onSelectionChanged(self, table, selection, *args, **kwargs)
    onSelectionActivated(self, table, selection)
    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    openEditor(self, key)
    resetLoadingState(self)

class vi.actions.list.CloneAction(*args, **kwargs)
    Bases: flare.button.Button
    Allows cloning an entry in a list-module.
    onAttach(self)
    onDetach(self)
    onSelectionChanged(self, table, selection, *args, **kwargs)
    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    openEditor(self, key)
    resetLoadingState(self)

class vi.actions.list.DeleteAction(*args, **kwargs)
    Bases: flare.button.Button
    Allows deleting an entry in a list-module.
    onAttach(self)
    onDetach(self)
    onSelectionChanged(self, table, selection, *args, **kwargs)
    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    doDelete(self, dialog)
    allDeletedSuccess(self, success)
    deletedSuccess(self, req=None, code=None)
```



```

    deletedFailed(self, req=None, code=None)

    resetLoadingState(self)

class vi.actions.list.ListPreviewAction(module, handler, actionName, *args, **kwargs)
    Bases: flare.html5.Span
    onChange(self, event)
    rebuildCB(self, *args, **kwargs)
    onAttach(self)
    onDetach(self)
    onSelectionChanged(self, table, selection, *args, **kwargs)
    onClick(self, sender=None)
    static isSuitableFor(module, handler, actionName)

class vi.actions.list.ListPreviewInlineAction(*args, **kwargs)
    Bases: flare.button.Button
    onAttach(self)
    onDetach(self)
    onSelectionChanged(self, table, selection, *args, **kwargs)
    onClick(self, sender=None)
    toggleIntPrev(self)
    static isSuitableFor(module, handler, actionName)

class vi.actions.list.CloseAction(*args, **kwargs)
    Bases: flare.button.Button
    onClick(self, sender=None)
    static isSuitableFor(module, handler, actionName)

class vi.actions.list.SelectAction(*args, **kwargs)
    Bases: flare.button.Button
    onClick(self, sender=None)
    static isSuitableFor(module, handler, actionName)

class vi.actions.list.SelectFieldsPopup(listWdg, *args, **kwargs)
    Bases: flare.popup.Popup
    doApply(self, *args, **kwargs)
    doSetFields(self, *args, **kwargs)
    doCancel(self, *args, **kwargs)
    doSelectAll(self, *args, **kwargs)

```

```
    doUnselectAll(self, *args, **kwargs)

    doInvertSelection(self, *args, **kwargs)

class vi.actions.list.SelectFieldsAction(*args, **kwargs)
    Bases: flare.button.Button
    onClick(self, sender=None)
    onAttach(self)
    onDetach(self)
    onTableChanged(self, table, count, *args, **kwargs)
    static isSuitableFor(module, handler, actionName)

class vi.actions.list.ReloadAction(*args, **kwargs)
    Bases: flare.button.Button
    Allows Reloading
    static isSuitableFor(module, handler, actionName)
    onClick(self, event=None)
    resetLoadingState(self)

class vi.actions.list.TableNextPage(*args, **kwargs)
    Bases: flare.button.Button
    postInit(self, widget=None)
    onClick(self, sender=None)
    static isSuitableFor(module, handler, actionName)
    resetLoadingState(self)

class vi.actions.list.TablePrevPage(*args, **kwargs)
    Bases: flare.button.Button
    postInit(self, widget=None)
    onClick(self, sender=None)
    static isSuitableFor(module, handler, actionName)

class vi.actions.list.TableItems(*args, **kwargs)
    Bases: flare.html5.Div
    postInit(self, widget=None)
    onTableChanged(self, table, rowCount, *args, **kwargs)
    static isSuitableFor(module, handler, actionName)

class vi.actions.list.SetPageRowAmountAction(*args, **kwargs)
    Bases: flare.html5.Div
    Load a bunch of pages
```

```

    onClick(self, sender=None)

    onChange(self, sender=None)

    setPageAmount(self)

    static isSuitableFor(module, handler, actionName)

    resetLoadingState(self)

class vi.actions.list.LoadNextBatchAction(*args, **kwargs)
    Bases: flare.html5.Div
    Load a bunch of pages
    registerScroll(self)

    onScroll(self, sender=None)

    onClick(self, sender=None)

    onChange(self, sender=None)

    loadnextPages(self, *args, **kwargs)

    static isSuitableFor(module, handler, actionName)

    resetLoadingState(self)

class vi.actions.list.LoadAllAction(*args, **kwargs)
    Bases: flare.button.Button
    Allows Loading all Entries in a list
    static isSuitableFor(module, handler, actionName)

    onClick(self, sender=None)

    loadAllRows(self)

    resetLoadingState(self)

class vi.actions.list.PageFindAction(*args, **kwargs)
    Bases: flare.html5.Div
    Allows Loading all Entries in a list
    onKeyPress(self, event)

    static isSuitableFor(module, handler, actionName)

    onClick(self, sender=None)

    startFind(self)

    findText(self)

    resetLoadingState(self)

class vi.actions.list.ListSelectFilterAction(*args, **kwargs)
    Bases: flare.button.Button

```

```
    onAttach(self)

    onClick(self, sender=None)

    static isSuitableFor(module, handler, actionName)

class vi.actions.list.CreateRecurrentAction(*args, **kwargs)
    Bases: flare.button.Button
    static isSuitableFor(module, handler, actionName)

    onClick(self, sender=None)

class vi.actions.list.ExportCsvAction(*args, **kwargs)
    Bases: flare.button.Button
    onClick(self, sender=None)

    static isSuitableFor(module, handler, actionName)

class vi.actions.list.SelectAllAction(*args, **kwargs)
    Bases: flare.button.Button
    static isSuitableFor(module, handler, actionName)

    onClick(self, sender=None)

    onAttach(self)

    onDetach(self)

    onTableChanged(self, table, count, *args, **kwargs)

class vi.actions.list.UnSelectAllAction(*args, **kwargs)
    Bases: flare.button.Button
    static isSuitableFor(module, handler, actionName)

    onClick(self, sender=None)

    onAttach(self)

    onDetach(self)

    onTableChanged(self, table, count, *args, **kwargs)

class vi.actions.list.SelectInvertAction(*args, **kwargs)
    Bases: flare.button.Button
    static isSuitableFor(module, handler, actionName)

    onClick(self, sender=None)

    onAttach(self)

    onDetach(self)

    onTableChanged(self, table, count, *args, **kwargs)
```

`vi.actions.list_order`

Module Contents

Classes

ShopMarkAction

ShopMarkPayedAction

ShopMarkSentAction

ShopMarkCanceledAction

```
class vi.actions.list_order.ShopMarkAction(action, title, cls="", txtQuestion=None, txtSuccess=None,
                                           txtFailure=None, *args, **kwargs)
```

```
    Bases: flare.button.Button
```

```
    onAttach(self)
```

```
    onDetach(self)
```

```
    onSelectionChanged(self, table, selection, *args, **kwargs)
```

```
    setPayed(self, order)
```

```
    setPayedSucceeded(self, response)
```

```
    setPayedFailed(self, response)
```

```
    doMarkPayed(self, *args, **kwargs)
```

```
    onClick(self, sender=None)
```

```
class vi.actions.list_order.ShopMarkPayedAction(*args, **kwargs)
```

```
    Bases: ShopMarkAction
```

```
    static isSuitableFor(module, handler, actionName)
```

```
class vi.actions.list_order.ShopMarkSentAction(*args, **kwargs)
```

```
    Bases: ShopMarkAction
```

```
    static isSuitableFor(module, handler, actionName)
```

```
class vi.actions.list_order.ShopMarkCanceledAction(*args, **kwargs)
```

```
    Bases: ShopMarkAction
```

```
    static isSuitableFor(module, handler, actionName)
```

vi.actions.tree

Module Contents

Classes

<i>AddLeafAction</i>	Creates a new leaf (ie. a file) for a tree application
<i>AddNodeAction</i>	Creates a new node (ie. a directory) for a tree application
<i>EditAction</i>	Edits an entry inside a tree application.
<i>DeleteAction</i>	Allows deleting an entry in a tree-module.
<i>ReloadAction</i>	Allows adding an entry in a list-module.
<i>SelectRootNode</i>	Allows selecting a different rootNode in Tree applications

```
class vi.actions.tree.AddLeafAction(*args, **kwargs)
```

```
    Bases: flare.button.Button
```

```
    Creates a new leaf (ie. a file) for a tree application
```

```
    static isSuitableFor(module, handler, actionName)
```

```
    onClick(self, sender=None)
```

```
    resetLoadingState(self)
```

```
class vi.actions.tree.AddNodeAction(*args, **kwargs)
```

```
    Bases: flare.button.Button
```

```
    Creates a new node (ie. a directory) for a tree application
```

```
    static isSuitableFor(module, handler, actionName)
```

```
    onClick(self, sender=None)
```

```
    resetLoadingState(self)
```

```
class vi.actions.tree.EditAction(*args, **kwargs)
```

```
    Bases: flare.button.Button
```

```
    Edits an entry inside a tree application. The type (node or leaf) of the entry is determined dynamically
```

```
    onAttach(self)
```

```
    onDetach(self)
```

```
    onSelectionActivated(self, table, selection)
```

```
    onSelectionChanged(self, table, selection, *args, **kwargs)
```

```
    static isSuitableFor(module, handler, actionName)
```

```
    onClick(self, sender=None)
```

```
    resetLoadingState(self)
```

```
class vi.actions.tree.DeleteAction(*args, **kwargs)
```

Bases: flare.button.Button

Allows deleting an entry in a tree-module. The type (node or leaf) of the entry is determined dynamically.

```
onAttach(self)
```

```
onDetach(self)
```

```
onSelectionChanged(self, table, selection, *args, **kwargs)
```

```
static isSuitableFor(module, handler, actionName)
```

```
onClick(self, sender=None)
```

```
doDelete(self, dialog)
```

```
allDeletedSuccess(self, success)
```

```
resetLoadingState(self)
```

```
class vi.actions.tree.ReloadAction(*args, **kwargs)
```

Bases: flare.button.Button

Allows adding an entry in a list-module.

```
static isSuitableFor(module, handler, actionName)
```

```
onClick(self, sender=None)
```

```
resetLoadingState(self)
```

```
class vi.actions.tree.SelectRootNode(module, handler, actionName, *args, **kwargs)
```

Bases: flare.html5.Select

Allows selecting a different rootNode in Tree applications

```
onAttach(self)
```

```
onDetach(self)
```

```
update(self)
```

```
onRootNodeChanged(self, newNode, *args, **kwargs)
```

```
onRootNodesAvailable(self, req)
```

```
onChange(self, event)
```

```
static isSuitableFor(module, handler, actionName)
```

vi.framework

Subpackages

vi.framework.components

Submodules

vi.framework.components.actionbar

Module Contents

Classes

<i>ActionBar</i>	Provides the container for actions (add,edit,...) suitable for one module (eg. for lists).
------------------	--

```
class vi.framework.components.actionbar.ActionBar(module=None, appType=None,  
                                                  currentAction=None, *args, **kwargs)
```

Bases: flare.html5.Div

Provides the container for actions (add,edit,...) suitable for one module (eg. for lists).

setActions(*self, actions, widget=None*)

Sets the list of valid actions for this module. This function tries to resolve a suitable action-widget for each given action-name and adds them on success. All previous actions are removed. :param actions: List of names of actions which should be available. :type actions: list of str

getActions(*self*)

Returns the list of action-names currently active for this module. May also contain action-names which couldn't be resolved and therefore not displayed. :returns: List of str

resetLoadingState(*self*)

Resets the loading-state of each child. Each child has the ability to provide visual feedback once it has been clicked and started working. This function is called from our parent once that action has finished, so we can tell our children to return to a sane state.

vi.framework.components.datatable

Module Contents

Classes

<i>SelectTable</i>	Provides an Html-Table which allows for row selections.
<i>DataTable</i>	

<i>ViewportDataTable</i>	
--------------------------	--

```
class vi.framework.components.datatable.SelectTable(checkboxes=False, indexes=False, *args,  
                                                    **kwargs)
```

Bases: flare.ignite.Table

Provides an Html-Table which allows for row selections.

Parent widgets can register for certain events:

- **selectionChanged:** called if the current **_multi_** selection changes. (Ie the user holds ctrl and clicks a row). The selection might contain no, one or multiple rows. Its also called if the cursor moves. Its

called if the user simply double clicks a row. So its possible to receive a selectionActivated event without an selectionChanged Event.

- **selectionActivated:** called if a selection is activated, ie. a row is double-clicked or **Return** is pressed.
- **cursorMoved:** called when the currently active row changes. The user can select the current row with a single click or by moving the cursor up and down using the arrow keys.

onAttach(*self*)

setHeader(*self, headers*)

Sets the table-headers to 'headers' :param headers: list of strings :type headers: list

getTrByIndex(*self, idx*)

Retrieves the TR element by the given row number :param idx: Rownumber to retrieve the tr of :type idx: int :returns: HTMLTableRowElement

getIndexByTr(*self, tr*)

Returns the rowNumber for the given tr element or None if the given tr element is invalid. :param tr: A HTMLTableRowElement of this table :type tr: HTMLTableRowElement :returns: int or None

_rowForEvent(*self, event*)

Determines the row number for the given event

onChange(*self, event*)

onMouseDown(*self, event*)

onMouseOut(*self, event*)

onMouseUp(*self, event*)

onKeyDown(*self, event*)

onKeyUp(*self, event*)

onDbClick(*self, event*)

addSelectedRow(*self, row*)

Marks a row as selected

removeSelectedRow(*self, row*)

Removes 'row' from the current selection (if any) :param row: Number of the row to unselect :type row: int

selectRow(*self, newRow*)

Sets the current selection to 'row'. Any previous selection is removed. :param newRow: Number of the row to select :type newRow: int

setCursorRow(*self, row, removeExistingSelection=True*)

Move the cursor to row 'row'. If removeExistingSelection is True, the current selection (if any) is invalidated.

focusRow(*self, row*)

getCurrentSelection(*self*)

Returns a list of currently selected row-numbers :returns: list

clear(*self*)

Hook the clear() method so we can reset some internal states, too

removeRow(*self*, *row*)

Hook the removeRow method so we can reset some internal states, too

_extraCols(*self*)

prepareCol(*self*, *row*, *col*)

Lets hook up the original removeRow function to optionally provide index and checkbox columns.

dropTableContent(*self*)

Drops content from the table, structure remains unchanged

setCell(*self*, *row*, *col*, *val*)

Interface for self["cell"] that directs to the correct cell if extra columns are configured for this SelectTable.

selectAll(*self*)

Selects all entries of the table.

unselectAll(*self*)

Unselects all entries of the table.

invertSelection(*self*)

Inverts the current selection on the whole table currently displayed.

class vi.framework.components.datatable.DataTable(*_loadOnDisplay=False*, *args, **kwargs)

Bases: flare.html5.Div

setDataProvider(*self*, *obj*)

Register's 'obj' as the provider for this table. It must provide a onNextBatchNeeded function, which must fetch and feed new rows using add() or reset the dataProvider to None if no more rows are available. Notice: If the bottom of the table is reached, onNextBatchNeeded will only be called once. No further calls will be made until add() or setDataProvider() has been called afterwards.

onCursorMoved(*self*, *table*, *row*)

Ensure the table scrolls according to the position of its cursor

getRowCount(*self*)

Returns the total amount of rows currently known. :returns: int

add(*self*, *obj*)

Adds an row to the model :param obj: Dictionary of values for this row :type obj: dict

update(*self*, *objList*, *writeToModel=True*)

Adds multiple rows at once. Much faster than calling add() multiple times.

extend(*self*, *objList*, *writeToModel=True*)

remove(*self*, *objOrIndex*)

Removes 'obj' from the table. 'obj' may be an row-index or an object recieved by any eventListener. It cannot be any original object passed to 'add' - it must be recieved by an eventListener!

clear(*self*, *keepModel=False*)

Flushes the whole table.

_renderObject(*self*, *obj*, *tableIsPrepared=False*, *recalculate=True*)

Renders the object to into the table. Does nothing if the list of _shownFields is empty. :param obj: Dictionary of values for this row :type obj: dict

rebuildTable(*self*, *recalculate=True*)

Rebuilds the entire table. Useful if something fundamental changed (ie. the cell renderer or the list of visible fields)

setShownFields(*self*, *fields*)

Sets the list of `_shownFields`. This causes the whole table to be rebuild. Be careful if calling this function often on a large table! :param fields: List of model-keys which will be displayed. :type fields: list

onSelectionChanged(*self*, *table*, *rows*, **args*, ***kwargs*)

Re-emit the event. Maps row-numbers to actual models.

onSelectionActivated(*self*, *table*, *rows*)

Re-emit the event. Maps row-numbers to actual models.

onTableChanged(*self*, *table*, *rowCount*, **args*, ***kwargs*)

Re-emit the event.

getCurrentSelection(*self*)

Override the `getCurrentSelection` method to yield actual models, not row-numbers.

setCellRender(*self*, *field*, *render*)

Sets the render for cells of 'field' to render. A cell render receives the data for a given cell and returns the appropriate widget to display that data for the table.

setCellRenderers(*self*, *renders*)

Like `setCellRender`, but sets multiple renders at one. Much faster than calling `setCellRender` repeatedly.

activateCurrentSelection(*self*)

Emits the `selectionActivated` event if there's currently a selection

class `vi.framework.components.datatable.ViewportDataTable`(*_loadOnDisplay=False*, *rows=99*, **args*, ***kwargs*)

Bases: `DataTable`

clear(*self*, *keepModel=False*)

Flushes the whole table. Override explanation - replaced `clear` with `dropTableContent`

rebuildTable(*self*, *recalculate=True*)

Rebuilds the entire table. Useful if something fundamental changed (ie. the cell renderer or the list of visible fields)#

Override explanation - uses the predefined `_rows` to prepare the grid - only load first rows from model

add(*self*, *obj*)

Adds an row to the model :param obj: Dictionary of values for this row :type obj: dict

Override explanation - `_renderObject` call is always prepared

extend(*self*, *objList*, *writeToModel=True*)

update(*self*, *objList*, *writeToModel=True*)

Adds multiple rows at once. Much faster than calling `add()` multiple times.

Override explanation - removed grid preparation

_renderObject(*self*, *obj*, *tableIsPrepared=True*, *recalculate=True*)

Renders the object to into the table. Does nothing if the list of `_shownFields` is empty. :param obj: Dictionary of values for this row :type obj: dict

Override explanation - removed Table preperation - `rowIndex` modulo `shownrows`

`vi.sidebarwidgets`

Submodules

`vi.sidebarwidgets.filterselector`

Module Contents

Classes

CompoundFilter

FilterSelector

```
class vi.sidebarwidgets.filterselector.CompoundFilter(view, module, embed=False, *args,
                                                    **kwargs)
```

```
    Bases: flare.html5.Div
```

```
    onFilterChanged(self, *args, **kwargs)
```

```
    reevaluate(self, *args, **kwargs)
```

```
    focus(self)
```

```
class vi.sidebarwidgets.filterselector.FilterSelector(module, *args, **kwargs)
```

```
    Bases: flare.html5.Div
```

```
    onClick(self, event)
```

```
        Handle event on filter selection (fold current active filter, expand selected filter and execute, if possible)
        :param event: :return:
```

```
    onAttach(self)
```

```
    onDetach(self)
```

```
    onStartSearch(self, searchText=None)
```

```
    setView(self, btn)
```

```
    applyFilter(self, filter, filterID, filterName)
```

`vi.sidebarwidgets.internalpreview`

Module Contents

Classes

InternalPreview

```
class vi.sidebarwidgets.internalpreview.InternalPreview(module, structure, item, *args, **kwargs)
    Bases: flare.html5.Ul
        onCopyKey(self, btn)
```

vi.translations

Submodules

vi.translations.de

Module Contents

vi.translations.de.lngDe

vi.translations.en

Module Contents

vi.translations.en.lngEn

Package Contents

vi.translations.lngDe

vi.translations.lngEn

vi.views

Submodules

vi.views.edit

Module Contents

Classes

editHandler

editHandlerWidget

```
class vi.views.edit.editHandler
    Bases: flare.views.view.View
```

```
class vi.views.edit.editHandlerWidget
    Bases: flare.views.view.ViewWidget
    def __init__(self):
        Here we start!
```

vi.views.hierarchy

Module Contents

Classes

hierarchyHandler

hierarchyHandlerWidget

```
class vi.views.hierarchy.hierarchyHandler
    Bases: flare.views.view.View
    static def canHandle(moduleName, moduleInfo)

class vi.views.hierarchy.hierarchyHandlerWidget
    Bases: flare.views.view.ViewWidget
    def __init__(self):
        Here we start!
```

vi.views.list

Module Contents

Classes

listHandler

listHandlerWidget

```
class vi.views.list.listHandler
    Bases: flare.views.view.View
    static def canHandle(moduleName, moduleInfo)

class vi.views.list.listHandlerWidget
    Bases: flare.views.view.ViewWidget
    def __init__(self):
        Here we start!
    def onViewfocusedChanged(self, viewname, *args, **kwargs)
```

`vi.views.log`

Module Contents

Classes

logHandler

logHandlerWidget

class `vi.views.log.logHandler`

Bases: `flare.views.view.View`

class `vi.views.log.logHandlerWidget`

Bases: `flare.views.view.ViewWidget`

initWidget(*self*)

Here we start!

`vi.views.notfound`

Module Contents

Classes

NotFound

NotFoundWidget

class `vi.views.notfound.NotFound`

Bases: `flare.views.view.View`

class `vi.views.notfound.NotFoundWidget`

Bases: `flare.views.view.ViewWidget`

initWidget(*self*)

Here we start!

`vi.views.overview`

Module Contents

Classes

Overview

OverviewWidget

```
class vi.views.overview.Overview
    Bases: flare.views.view.View

class vi.views.overview.OverviewWidget
    Bases: flare.views.view.ViewWidget

    initWidget(self)
        Here we start!
```

vi.views.singleton

Module Contents

Classes

singletonHandler

singletonHandlerWidget

```
class vi.views.singleton.singletonHandler
    Bases: flare.views.view.View

    static canHandle(moduleName, moduleInfo)

class vi.views.singleton.singletonHandlerWidget
    Bases: flare.views.view.ViewWidget

    initWidget(self)
        Here we start!

    onViewfocusedChanged(self, viewname, *args, **kwargs)
```

vi.views.tree

Module Contents

Classes

treeHandler

treeHandlerWidget

```

class vi.views.tree.treeHandler
    Bases: flare.views.view.View
    static canHandle(moduleName, moduleInfo)

class vi.views.tree.treeHandlerWidget
    Bases: flare.views.view.ViewWidget
    initWidget(self)
        Here we start!

```

`vi.widgets`

Submodules

`vi.widgets.accordion`

Module Contents

Classes

AccordionSegment

Accordion

```

class vi.widgets.accordion.AccordionSegment(ident, title=None)
    Bases: flare.html5.Fieldset
    checkVisibility(self)
    activate(self)
    deactivate(self)
    isActive(self)
    toggle(self)
    onClick(self, event)
    addWidget(self, widget)

```

```

class vi.widgets.accordion.Accordion
    Bases: flare.html5.Form
    addSegment(self, ident, title=None, directAdd=False, *args)
    clear(self)
    buildAccordion(self, order=None)

```

Parameters **sort** – None: sorted by Bones, “asc”:ascending, “desc”:descending, dict: {“category”:index,... }

Returns

vi.widgets.appnavigation

Module Contents

Classes

NavigationElement

NavigationSeperator

Navigationblock

AppNavigation

class vi.widgets.appnavigation.**NavigationElement**(name, icon=None, view=None, nav=None, closeable=False, opened=False)

Bases: flare.html5.Div

tpl = Multiline-String

```
1         <div [name]="item" class="item has-hover">
2             <a class="item-link" @click="navigationAction">
3                 <div class="item-image">
4                     <flare-icon value="{{icon}}" title="
↪ {{name}}"></flare-icon>
5                 </div>
6
7                 <div class="item-content">
8                     <div class="item-headline">{{name}}
↪ </div>
9                 </div>
10            </a>
11
12            <span [name]="itemArrow" class="item-open is-hidden
↪ " @click="ArrowAction">
13                <flare-svg-icon value="icon-arrow-left"></
↪ flare-svg-icon>
14            </span>
15            <span [name]="itemRemove" class="item-pin is-hidden
↪ " @click="RemoveAction">
16                <flare-svg-icon value="icon-cancel"></flare-
↪ svg-icon>
17            </span>
18
19        </div>
20        <div [name]="subItem" class="list list--sub">
21            </div>
```

onActiveViewChanged(self, e, wdg, *args, **kwargs)

navigationAction(*self*, *e=None*, *wdg=None*)

Handle Click on Navigation Button

RemoveAction(*self*, *e=None*)

remove this Nav Element

ArrowAction(*self*, *e*, *wdg=None*)

onActiveNavigationChanged(*self*, *e*, *wdg*, **args*, ***kwargs*)

What should happen if the State from the surrounding Navigation gets an update

onHasSubItemsChanged(*self*, *e*, *wdg*, **args*, ***kwargs*)

If subChild is added, show itemArrow, hide if no subitem present

appendSubChild(*self*, *element*)

class `vi.widgets.appnavigation.NavigationSeperator`(*name=None*)

Bases: `flare.html5.Div`

buildSeperator(*self*)

_setValue(*self*, *value*)

class `vi.widgets.appnavigation.Navigationblock`(*name*)

Bases: `flare.html5.Div`

addSeperator(*self*)

seperatorAction(*self*, *e*, *wdg=None*)

class `vi.widgets.appnavigation.AppNavigation`

Bases: `flare.html5.Nav`

getPreviousNavigationPoint(*self*, *view*)

getNavigationPoint(*self*, *view*)

addNavigationBlock(*self*, *name*)

addNavigationPoint(*self*, *name*, *icon*, *view=None*, *parent=None*, *closeable=False*, *opened=False*)

addNavigationPointAfter(*self*, *name*, *icon*, *view=None*, *beforeElement=None*, *closeable=False*, *opened=False*)

removeNavigationPoint(*self*, *view*)

`vi.widgets.code`

Module Contents

Classes

CodeHelpPopup

CodePopup

Codemirror

PythonCode

Scripter

```
class vi.widgets.code.CodeHelpPopup(title='Code Hilfe')
    Bases: flare.popup.Popup
class vi.widgets.code.CodePopup(title='Editor')
    Bases: flare.popup.Popup
class vi.widgets.code.Codemirror(syntax='python')
    Bases: flare.html5.Textarea
    _attachCodemirror(self)
    onAttach(self)
    onDetach(self)
    _getValue(self)
    _setValue(self, val)
    insertText(self, text)
    setCursor(self, line, char)
class vi.widgets.code.PythonCode(logger, scripter)
    Bases: flare.html5.Div
    addToLog(self, data, type='normal')
        allowed types: normal, info, warn, error
    workerFeedback(self, e)
    run(self)
    stop(self)
class vi.widgets.code.Scripter(coder=PythonCode, exampleCode=None, executable=True)
    Bases: flare.html5.Div
    runClick(self, event)
    killClick(self, event)
    openHelp(self, event)
    startFullscreen(self, event)
```

vi.widgets.csvexport

Module Contents

Classes

ExportCsv

ExportCsvStarter

```
class vi.widgets.csvexport.ExportCsv(widget, selection, encoding=None, language=None,
                                     separator=None, lineSeparator=None, *args, **kwargs)
```

```
    Bases: flare.html5.Progress
```

```
    nextChunk(self, cursor=None)
```

```
    nextChunkComplete(self, req)
```

```
    exportToFile(self)
```

```
    nextChunkFailure(self, req, code)
```

```
    replaceWithMessage(self, message, logClass='success')
```

```
class vi.widgets.csvexport.ExportCsvStarter(widget, *args, **kwargs)
```

```
    Bases: flare.popup.Popup
```

```
    onExportBtnClick(self, *args, **kwargs)
```

vi.widgets.edit

Module Contents

Classes

EditWidget

Functions

parseHashParameters(src, prefix="")

Converts a flat dictionary containing dotted properties into a multi-dimensional one.

```
vi.widgets.edit.parseHashParameters(src, prefix="")
```

```
    Converts a flat dictionary containing dotted properties into a multi-dimensional one.
```

```
    Example: { "a": "a", "b.a": "ba", "b.b": "bb" } -> { "a": "a", "b": { "a": "ba", "b": "bb" } }
```

If a dictionary contains only numeric indexes, it will be converted to a list: { "a.0.a": "a0a", "a.0.b": "a0b", a.1.a": "a1a" } -> { "a": [{"a": "a0a", "b": "a0b"}], {"a": "a1a"}] }

```
class vi.widgets.edit.EditWidget(module, applicationType, key=0, node=None, skelType=None,
                                clone=False, hashArgs=None, context=None, logAction='Entry saved!',
                                skel=None, *args, **kwargs)
```

Bases: flare.html5.Div

appList = list

appHierarchy = hierarchy

appTree = tree

appSingleton = singleton

__editIdx__ = 0

onDetach(self)

onAttach(self)

onChange(self, event)

onBoneChange(self, bone)

showErrorMsg(self, req=None, code=None)

Removes all currently visible elements and displays an error message

reloadData(self)

_save(self, data)

Creates the actual NetworkService request used to transmit our data. If data is None, it fetches a clean add/edit form.

Parameters data (*dict* or *None*) – The values to transmit or None to fetch a new, clean add/edit form.

clear(self)

Removes all visible bones/forms/fieldsets.

closeOrContinue(self, sender=None)

doCloneHierarchy(self, sender=None)

cloneComplete(self, req)

setData(self, request=None, data=None, askHierarchyCloning=True)

Rebuilds the UI according to the skeleton received from server

Parameters

- **request** (*NetworkService*) – A finished NetworkService request
- **data** (*dict*) – The data received

doSave(self, closeOnSuccess=False, *args, **kwargs)

Starts serializing and transmitting values to the server.

vi.widgets.file**Module Contents****Classes**

<i>FileImagePopup</i>	
<i>FilePreviewImage</i>	
<i>Uploader</i>	Uploads a file to the server while providing visual feedback of the progress.
<i>MultiUploader</i>	
<i>FileLeafWidget</i>	
<i>FileNodeWidget</i>	
<i>FileWidget</i>	Base Widget that renders a tree.

class vi.widgets.file.**FileImagePopup**(*preview, *args, **kwargs*)

Bases: flare.popup.Popup

onClick(*self, event*)

onDownloadBtnClick(*self, sender=None*)

class vi.widgets.file.**FilePreviewImage**(*file=None, size=150, *args, **kwargs*)

Bases: flare.html5.Div

setFile(*self, file*)

download(*self*)

onClick(*self, sender=None*)

class vi.widgets.file.**Uploader**(*file, node, context=None, module='file', *args, **kwargs*)

Bases: flare.html5.Div

Uploads a file to the server while providing visual feedback of the progress.

onUploadUrlAvailable(*self, req*)

Internal callback - the actual upload url (retrieved by calling /file/getUploadURL) is known.

onSkeyAvailable(*self, req*)

Internal callback - the Security-Key is known. Only for core 2.x needed

onLoad(*self, *args, **kwargs*)

Internal callback - The state of our upload changed.

onUploadAdded(*self, req*)

onProgress(*self, event*)

Internal callback - further bytes have been transmitted

onSuccess(*self*, *args, **kwargs)

Internal callback - The upload succeeded.

onFailed(*self*, *errorCode*, *args, **kwargs)

replaceWithMessage(*self*, *message*, *isSuccess*)

class *vi.widgets.file.MultiUploader*(*files*, *node*, *context=None*, *module='file'*, *args, **kwargs)

Bases: *flare.html5.Div*

handleFile(*self*, *file*)

onUploadUrlAvailable(*self*, *req*)

Internal callback - the actual upload url (retrieved by calling */file/getUploadURL*) is known.

onLoad(*self*, *args, **kwargs)

Internal callback - The state of our upload changed.

onUploadAdded(*self*, *req*)

onSuccess(*self*, *args, **kwargs)

Internal callback - The upload succeeded.

onFailed(*self*, *errorCode*, *args, **kwargs)

replaceWithMessage(*self*, *message*, *isSuccess*)

closeMessage(*self*)

class *vi.widgets.file.FileLeafWidget*

Bases: *vi.widgets.tree.TreeLeafWidget*

EntryIcon(*self*)

setStyle(*self*)

class *vi.widgets.file.FileNodeWidget*

Bases: *vi.widgets.tree.TreeNodeWidget*

setStyle(*self*)

class *vi.widgets.file.FileWidget*(*module*, *rootNode=None*, *selectMode=None*, *node=None*, *context=None*, *args, **kwargs)

Bases: *vi.widgets.tree.TreeBrowserWidget*

Base Widget that renders a tree.

leafWidget

nodeWidget

searchWidget(*self*)

onStartSearch(*self*, *searchStr*, *args, **kwargs)

getChildKey(*self*, *widget*)

Derives a string used to sort the entries on each level

onDrop(*self*, *event*)

We got a drop event. Make that item a direct child of our *rootNode*

static canHandle(*module*, *moduleInfo*)

vi.widgets.hierarchy

Module Contents

Classes

HierarchyWidget

A Hierarchy is a Tree without leaf distiction!

```
class vi.widgets.hierarchy.HierarchyWidget(*args, **kwargs)
```

Bases: *vi.widgets.tree.TreeWidget*

A Hierarchy is a Tree without leaf distiction!

leafWidget

reloadData(*self*)

Reload the data were displaying.

reloadListWidget(*self*)

toggleListView(*self*)

setListView(*self*, *visible=False*)

showListView(*self*)

hideListView(*self*)

onSelectionChanged(*self*, *widget*, *selection*, *args, **kwargs)

static canHandle(*moduleName*, *moduleInfo*)

vi.widgets.htmleditor

Module Contents

Classes

TextInsertImageAction

HtmlEditor

```
class vi.widgets.htmleditor.TextInsertImageAction(summernote=None, boneName="", *args,
                                                    **kwargs)
```

Bases: *flare.button.Button*

onClick(*self*, *sender=None*)

onSelectionActivated(*self*, *selectWdg*, *selection*)

static isSuitableFor(*modul*, *handler*, *actionName*)

```
    resetLoadingState(self)

class vi.widgets.htmleditor.HtmlEditor(*args, **kwargs)
    Bases: flare.html5.Textarea
    initSources = False
    _attachSummernote(self, retry=0)
    onAttach(self)
    onDetach(self)
    onEditorChange(self, e, *args, **kwargs)
    _getValue(self)
    _setValue(self, val)
    enable(self)
    disable(self)
```

`vi.widgets.internaledit`

Module Contents

Classes

ParsedErrorItem

PassiveErrorItem

InternalEdit

Functions

checkErrors(bone) → Tuple[bool, List[str]]

```
class vi.widgets.internaledit.ParsedErrorItem(error)
    Bases: flare.html5.Li
    style = []
```

```
class vi.widgets.internaledit.PassiveErrorItem(error)
    Bases: flare.html5.Li
    style = []
```

```
vi.widgets.internaledit.checkErrors(bone) → Tuple[bool, List[str]]
```

```
class vi.widgets.internaedit.InternalEdit(skelStructure, values=None, errorInformation=None,
                                         readOnly=False, context=None, defaultCat="",
                                         module=None, boneparams=None, errorQueue=None,
                                         prefix=None)
```

Bases: flare.html5.Div

renderStructure(*self, readOnly=False*)

serializeForPost(*self, validityCheck=False*)

serializeForDocument(*self*)

doSave(*self, closeOnSuccess=False, *args, **kwargs*)

Starts serializing and transmitting our values to the server.

unserialize(*self, data=None*)

Applies the actual data to the bones.

onChange(*self, event*)

onKeyDown(*self, event*)

performLogics(*self*)

vi.widgets.list

Module Contents

Classes

<i>ListWidget</i>	Provides the interface to list-applications.
<i>ViewportListWidget</i>	Provides the interface to list-applications.

```
class vi.widgets.list.ListWidget(module, filter=None, columns=None, filterID=None, filterDescr=None,
                                batchSize=None, context=None, autoload=True, *args, **kwargs)
```

Bases: flare.html5.Div

Provides the interface to list-applications. It acts as a data-provider for a DataTable and binds an action-bar to this table.

setSelector(*self, callback, multi=True, allow=None*)

Configures the widget as selector for a relationalBone and shows it.

selectorReturn(*self*)

Returns the current selection to the callback configured with *setSelector*.

tableInitialization(*self, *args, **kwargs*)

Instantiates the table :param args: ListWidget Parameter :param kwargs: ListWidget Parameter :return:

setAmount(*self, amount*)

setPage(*self, page=0*)

sets targetpage. if not enough loadedpages this pages will be requested :param page: sets targetpage :return:

onRequestingFinished(*self*, *args, **kwargs)

onClick(*self*, event)

setTableActionBar(*self*)

getDefaultEntryActions(*self*)
Returns the list of actions available in our actionBar

getActions(*self*)
Returns the list of actions available in our actionBar

getAllActions(*self*, view=None)
Returns the list of actions available in the action bar.

showErrorMsg(*self*, req=None, code=None)
Removes all currently visible elements and displays an error message

onNextBatchNeeded(*self*)
Requests the next rows from the server and feed them to the table.

onAttach(*self*)

onDetach(*self*)

onDataChanged(*self*, module, *args, **kwargs)
Refresh our view if element(s) in this module have changed

requestStructure(*self*)

receivedStructure(*self*, resp)

reloadData(*self*)
Removes all currently displayed data and refetches the first batch from the server.

setFilter(*self*, filter, filterID=None, filterDescr=None)
Applies a new filter.

setContext(*self*, context)
Applies a new context.

getFilter(*self*)

updateEmptyNotification(*self*)

onCompletion(*self*, req)
Pass the rows received to the datatable. :param req: The network request that succeed.

setFields(*self*, fields)

getFields(*self*)

onSelectionActivated(*self*, table, selection)

activateSelection(*self*)

static canHandle(moduleName, moduleInfo)

```
class vi.widgets.list.ViewportListWidget(module, filter=None, columns=None, filterID=None,  
                                         filterDescr=None, batchSize=None, context=None,  
                                         autoload=True, *args, **kwargs)
```

Bases: [ListWidget](#)

Provides the interface to list-applications. It acts as a data-provider for a DataTable and binds an action-bar to this table.

```
tableInitialization(self, *args, **kwargs)
```

Instantiates the table :param args: ListWidget Parameter :param kwargs: ListWidget Parameter

Override explanation

- use ViewPort DataTable with rows parameter

```
setAmount(self, amount)
```

```
setPage(self, page=0)
```

sets targetpage. if not enough loadedpages this pages will be requested else

Parameters **page** – sets targetpage

Returns

```
_setPage(self, page=0)
```

render page to table :param page: :return:

```
onRequestingFinished(self, *args, **kwargs)
```

```
setTableActionBar(self)
```

```
static canHandle(moduleName, moduleInfo)
```

vi.widgets.preview

Module Contents

Classes

[Preview](#)

```
class vi.widgets.preview.Preview(urls, entry, modul, *args, **kwargs)
```

Bases: `flare.html5.Div`

```
onChange(self, event)
```

```
setUrl(self, url)
```

```
doClose(self, *args, **kwargs)
```

vi.widgets.repeatdate

Module Contents

Classes

RepeatDatePopup

```
class vi.widgets.repeatdate.RepeatDatePopup(module, key)
    Bases: flare.html5.Div
    __editIdx_ = 0
    reloadData(self)
    save(self, data)
    setData(self, request=None, data=None, ignoreMissing=False)
        Rebuilds the UI according to the skeleton received from server
        Parameters
        • request (NetworkService) – A finished NetworkService request
        • data (dict) – The data received
    clear(self)
        Removes all visible bones/forms/fieldsets.
    showErrorMsg(self, req=None, code=None)
        Removes all currently visible elements and displays an error message
    doSave(self, closeOnSuccess=False)
```

vi.widgets.search

Module Contents

Classes

Search

```
class vi.widgets.search.Search(*args, **kwargs)
    Bases: flare.html5.Div
    doSearch(self, *args, **kwargs)
    resetSearch(self)
    onKeyDown(self, event)
```

`resetLoadingState(self)`

`reevaluate(self)`

`focus(self)`

`vi.widgets.sidebar`

Module Contents

Classes

SideBar

class `vi.widgets.sidebar.SideBar(*args, **kwargs)`

Bases: `flare.html5.Div`

`onAttach(self)`

`onDetach(self)`

`setWidget(self, widget)`

`getWidget(self)`

`close(self, *args, **kwargs)`

`vi.widgets.table`

Module Contents

Classes

SelectTable

Provides an Html-Table which allows for row selections.

DataTable

Provides kind of MVC on top of SelectTable.

class `vi.widgets.table.SelectTable(checkboxes=False, indexes=False, *args, **kwargs)`

Bases: `flare.ignite.Table`

Provides an Html-Table which allows for row selections.

Parent widgets can register for certain events:

- **selectionChanged:** called if the current `_multi_` selection changes. (If the user holds ctrl and clicks a row). The selection might contain no, one or multiple rows. Its also called if the cursor moves. Its called if the user simply double clicks a row. So its possible to receive a `selectionActivated` event without an `selectionChanged` Event.
- **selectionActivated:** called if a selection is activated, ie. a row is double-clicked or **Return** is pressed.

- **cursorMoved:** called when the currently active row changes. The user can select the current row with a single click or by moving the cursor up and down using the arrow keys.

onAttach(*self*)

setHeader(*self*, *headers*)

Sets the table-headers to 'headers' :param headers: list of strings :type headers: list

getTrByIndex(*self*, *idx*)

Retrieves the TR element by the given row number :param idx: Rownumber to retrieve the tr of :type idx: int :returns: HTMLTableRowElement

getIndexByTr(*self*, *tr*)

Returns the rowNumber for the given tr element or None if the given tr element is invalid. :param tr: A HTMLTableRowElement of this table :type tr: HTMLTableRowElement :returns: int or None

_rowForEvent(*self*, *event*)

Determines the row number for the given event

onChange(*self*, *event*)

onMouseDown(*self*, *event*)

onMouseOut(*self*, *event*)

onMouseUp(*self*, *event*)

onKeyDown(*self*, *event*)

onKeyUp(*self*, *event*)

onDbClick(*self*, *event*)

addSelectedRow(*self*, *row*)

Marks a row as selected

removeSelectedRow(*self*, *row*)

Removes 'row' from the current selection (if any) :param row: Number of the row to unselect :type row: int

selectRow(*self*, *newRow*)

Sets the current selection to 'row'. Any previous selection is removed. :param newRow: Number of the row to select :type newRow: int

setCursorRow(*self*, *row*, *removeExistingSelection=True*)

Move the cursor to row 'row'. If removeExistingSelection is True, the current selection (if any) is invalidated.

focusRow(*self*, *row*)

getCurrentSelection(*self*)

Returns a list of currently selected row-numbers :returns: list

clear(*self*)

Hook the clear() method so we can reset some internal states, too

removeRow(*self*, *row*)

Hook the removeRow method so we can reset some internal states, too

_extraCols(*self*)

prepareCol(*self*, *row*, *col*)

Lets hook up the original `removeRow` function to optionally provide index and checkbox columns.

setCell(*self*, *row*, *col*, *val*)

Interface for `self["cell"]` that directs to the correct cell if extra columns are configured for this `SelectTable`.

selectAll(*self*)

Selects all entries of the table.

unselectAll(*self*)

Unselects all entries of the table.

invertSelection(*self*)

Inverts the current selection on the whole table currently displayed.

class `vi.widgets.table.DataTable`(*_loadOnDisplay=False*, **args*, ***kwargs*)

Bases: `flare.html5.Div`

Provides kind of MVC on top of `SelectTable`.

recalcHeight(*self*, **args*, ***kwargs*)

setDataProvider(*self*, *obj*)

Register's 'obj' as the provider for this table. It must provide a `onNextBatchNeeded` function, which must fetch and feed new rows using `add()` or reset the `dataProvider` to `None` if no more rows are available. Notice: If the bottom of the table is reached, `onNextBatchNeeded` will only be called once. No further calls will be made until `add()` or `setDataProvider()` has been called afterwards.

onCursorMoved(*self*, *table*, *row*)

Ensure the table scrolls according to the position of its cursor

getRowCount(*self*)

Returns the total amount of rows currently known. :returns: `int`

add(*self*, *obj*)

Adds an row to the model :param *obj*: Dictionary of values for this row :type *obj*: `dict`

extend(*self*, *objList*)

Adds multiple rows at once. Much faster than calling `add()` multiple times.

testIfNextBatchNeededImmediately(*self*)

Test if we display enough entries so that our contents are scrollable. Otherwise, we'll never request a second batch

remove(*self*, *objOrIndex*)

Removes 'obj' from the table. 'obj' may be an row-index or an object recieved by any `eventListener`. It `_cannot_` be any original object passed to 'add' - it `_must_` be recieved by an `eventListener`!

clear(*self*, *keepModel=False*)

Flushes the whole table.

_renderObject(*self*, *obj*, *tableIsPrepared=False*)

Renders the object to into the table. Does nothing if the list of `_shownFields` is empty. :param *obj*: Dictionary of values for this row :type *obj*: `dict`

rebuildTable(*self*)

Rebuilds the entire table. Useful if something fundamental changed (ie. the cell renderer or the list of visible fields)

setShownFields(*self*, *fields*)

Sets the list of `_shownFields`. This causes the whole table to be rebuild. Be careful if calling this function often on a large table! :param fields: List of model-keys which will be displayed. :type fields: list

onScroll(*self*, *event*)

Check if we got a scroll event and need to fetch another set of rows from our dataProvider

onSelectionChanged(*self*, *table*, *rows*, **args*, *kwargs*)**

Re-emit the event. Maps row-numbers to actual models.

onSelectionActivated(*self*, *table*, *rows*)

Re-emit the event. Maps row-numbers to actual models.

onTableChanged(*self*, *table*, *rowCount*, **args*, *kwargs*)**

Re-emit the event.

getCurrentSelection(*self*)

Override the `getCurrentSelection` method to yield actual models, not row-numbers.

setCellRender(*self*, *field*, *render*)

Sets the render for cells of 'field' to render. A cell render receives the data for a given cell and returns the appropriate widget to display that data for the table.

setCellRenderers(*self*, *renders*)

Like `setCellRender`, but sets multiple renders at one. Much faster than calling `setCellRender` repeatedly.

activateSelection(*self*)

Emits the `selectionActivated` event if there's currently a selection

vi.widgets.task**Module Contents****Classes**

TaskWidget

ServerTaskWidget

TaskSelectWidget

class vi.widgets.task.**TaskWidget**(*title*)

Bases: flare.popup.Popup

class vi.widgets.task.**ServerTaskWidget**(*title*, *key*)

Bases: *TaskWidget*

```
class vi.widgets.task.TaskSelectWidget
    Bases: TaskWidget
    getSelectedTask(self)
    setActiveTask(self)
    onChange(self, event)
    invokeTask(self, *args, **kwargs)
```

vi.widgets.tooltip

Module Contents

Classes

ToolTip	Small utility class for providing tooltips
---------	--

```
class vi.widgets.tooltip.ToolTip(shortText="", longText="", *args, **kwargs)
    Bases: flare.html5.Div
    Small utility class for providing tooltips
    onClick(self, event)
    _setDisabled(self, disabled)
    _getDisabled(self)
```

vi.widgets.topbar

Module Contents

Classes

TopBarWidget	Provides the top-bar of VI
UserState	
Tasks	
Logout	
Scripter	

```
class vi.widgets.topbar.TopBarWidget
    Bases: flare.html5.Header
    Provides the top-bar of VI
```

```
    invoke(self)

    setTitle(self, title=None)

    onClick(self, event)

    setCurrentModulDescr(self, descr="", iconURL=None, iconClasses=None, path=None)

class vi.widgets.topbar.UserState(*args, **kwargs)
    Bases: flare.html5.Div
    onCurrentUserAvailable(self, req)

    update(self)

    static canHandle(action)

    onClick(self, sender=None)

    openEdit(self, key)

class vi.widgets.topbar.Tasks(*args, **kwargs)
    Bases: flare.button.Button
    onTaskListAvailable(self, req)

    onTaskListFailure(self)

    onCurrentUserAvailable(self, req)

    update(self)

    onClick(self, event)

    static canHandle(action)

class vi.widgets.topbar.Logout(*args, **kwargs)
    Bases: flare.button.Button
    onClick(self, event)

    logout(self)

    static canHandle(action)

class vi.widgets.topbar.Scripter(*args, **kwargs)
    Bases: flare.button.Button
    onCurrentUserAvailable(self, req)

    updateUser(self)

    onClick(self, event)

    static canHandle(action)
```

vi.widgets.tree**Module Contents****Classes**

<i>TreeWidget</i>	Base Widget that renders a tree.
<i>BrowserLeafWidget</i>	
<i>BrowserNodeWidget</i>	
<i>BreadcrumbNodeWidget</i>	
<i>TreeBrowserWidget</i>	Base Widget that renders a tree.

class `vi.widgets.tree.TreeWidget`(*module*, *rootNode=None*, *node=None*, *context=None*, *args, **kwargs)

Bases: `flare.html5.Div`

Base Widget that renders a tree.

nodeWidget

leafWidget

requestStructure(*self*)

receivedStructure(*self*, *resp*)

setSelector(*self*, *callback*, *multi=True*, *allow=None*)

Configures the widget as selector for a relationalBone and shows it.

selectorReturn(*self*)

Returns the current selection to the callback configured with *setSelector*.

onKeyDown(*self*, *event*)

onKeyUp(*self*, *event*)

getActions(*self*)

Returns a list of actions that are being set for the ActionBar. Override this to provide additional actions.

clearSelection(*self*)

Empties the current selection.

extendSelection(*self*, *element*)

Extends the current selection to element.

This is normally done by clicking or tabbing on an element.

activateSelection(*self*, *element*)

Activates the current selection or element.

An activation mostly is an action like selecting or editing an item. This is normally done by double-clicking an element.

requestChildren(*self*, *element*)

showErrorMsg(*self*, *req=None*, *code=None*)

Removes all currently visible elements and displays an error message

onDataChanged(*self*, *module*, **args*, ***kwargs*)

onAttach(*self*)

onDetach(*self*)

itemForKey(*self*, *key*, *elem=None*)

Returns the HierarchyWidget displaying the entry with the given key. :param key: The key (id) of the item.
:type key: str :returns: HierarchyItem

onSetDefaultRootNode(*self*, *req*)

We requested the list of rootNodes for that module and that request just finished. Parse the response and set our rootNode to the first rootNode received.

setRootNode(*self*, *rootNode*, *node=None*)

Set the currently displayed hierarchy to 'rootNode'. :param rootNode: Key of the rootNode which children we shall display :type rootNode: str

reloadData(*self*)

Reload the data were displaying.

loadNode(*self*, *node*, *cursor=None*, *reqType=None*, *overrideParams=None*)

Fetch the (direct) children of the given node. Once the list is received, append them to their parent node.
:param node: Key of the node to fetch :type node: str

onRequestSucceeded(*self*, *req*)

The NetworkRequest for a (sub)node finished. Create a new HierarchyItem for each entry received and add them to our view

onDrop(*self*, *event*)

We got a drop event. Make that item a direct child of our rootNode

onDragOver(*self*, *event*)

Allow dropping children on the rootNode

getChildKey(*self*, *widget*)

Order by sortindex

static canHandle(*moduleName*, *moduleInfo*)

class `vi.widgets.tree.BrowserLeafWidget`

Bases: `flare.viur.widgets.tree.TreeLeafWidget`

setStyle(*self*)

class `vi.widgets.tree.BrowserNodeWidget`

Bases: `flare.viur.widgets.tree.TreeNodeWidget`

setStyle(*self*)

class `vi.widgets.tree.BreadcrumbNodeWidget`

Bases: `flare.viur.widgets.tree.TreeNodeWidget`

setStyle(*self*)

```
class vi.widgets.tree.TreeBrowserWidget(module, rootNode=None, node=None, context=None, *args,
                                         **kwargs)
```

Bases: [*TreeWidget*](#)

Base Widget that renders a tree.

leafWidget

nodeWidget

reloadData(*self*)

Reload the data were displaying.

rebuildPath(*self*)

Rebuild the displayed path-list.

onPathRequestSucceded(*self*, *req*)

Rebuild the displayed path-list according to request data

activateSelection(*self*, *element*)

Activates the current selection or element.

An activation mostly is an action like selecting or editing an item. This is normally done by double-clicking an element.

static canHandle(*module*, *moduleInfo*)

[**vi.widgets.userlogoutmsg**](#)

Module Contents

Classes

[*UserLogoutMsg*](#)

```
class vi.widgets.userlogoutmsg.UserLogoutMsg(*args, **kwargs)
```

Bases: `flare.popup.Popup`

pollInterval = 120

checkIntervall

visibilityChanged(*self*, *e*)

stopInterval(*self*)

hideMessage(*self*)

Make this popup invisible

showMessage(*self*)

Show this popup

showLoginWindow(*self*, *args, **kwargs)

Return to the login window.

checkForSuspendResume(*self*, *args, **kwargs)

Test if at least self.pollIntervall seconds have passed and query the server if

startPolling(*self*, *args, **kwargs)

Start querying the server

onUserTestSuccess(*self*, req)

We received a response from the server

onUserTestFail(*self*, text, ns)

Error retrieving the current user response from the server

Package Contents

Classes

<i>Accordion</i>	
<i>TopBarWidget</i>	Provides the top-bar of VI
<i>ListWidget</i>	Provides the interface to list-applications.
<i>EditWidget</i>	
<i>ToolTip</i>	Small utility class for providing tooltips
<i>DataTable</i>	Provides kind of MVC on top of SelectTable.
<i>Search</i>	
<i>SideBar</i>	
<i>UserLogoutMsg</i>	
<i>TaskWidget</i>	
<i>TaskSelectWidget</i>	
<i>InternalEdit</i>	
<i>HtmlEditor</i>	
<i>ExportCsv</i>	
<i>ExportCsvStarter</i>	
<i>TreeWidget</i>	Base Widget that renders a tree.
<i>TreeBrowserWidget</i>	Base Widget that renders a tree.
<i>HierarchyWidget</i>	A Hierarchy is a Tree without leaf distiction!
<i>FileWidget</i>	Base Widget that renders a tree.

class vi.widgets.**Accordion**

Bases: flare.html5.Form

addSegment(*self*, ident, title=None, directAdd=False, *args)

clear(*self*)

buildAccordion(*self*, *order=None*)

Parameters **sort** – None: sorted by Bones, “asc”:ascending, “desc”:descending, dict: {“category”:index,... }

Returns

class **vi.widgets.TopBarWidget**

Bases: flare.html5.Header

Provides the top-bar of VI

invoke(*self*)

setTitle(*self*, *title=None*)

onClick(*self*, *event*)

setCurrentModulDescr(*self*, *descr="*, *iconURL=None*, *iconClasses=None*, *path=None*)

class **vi.widgets.ListWidget**(*module*, *filter=None*, *columns=None*, *filterID=None*, *filterDescr=None*, *batchSize=None*, *context=None*, *autoload=True*, **args*, ***kwargs*)

Bases: flare.html5.Div

Provides the interface to list-applications. It acts as a data-provider for a DataTable and binds an action-bar to this table.

setSelector(*self*, *callback*, *multi=True*, *allow=None*)

Configures the widget as selector for a relationalBone and shows it.

selectorReturn(*self*)

Returns the current selection to the callback configured with *setSelector*.

tableInitialization(*self*, **args*, ***kwargs*)

Instantiates the table :param args: ListWidget Parameter :param kwargs: ListWidget Parameter :return:

setAmount(*self*, *amount*)

setPage(*self*, *page=0*)

sets targetpage. if not enough loadedpages this pages will be requested :param page: sets targetpage :return:

onRequestingFinished(*self*, **args*, ***kwargs*)

onClick(*self*, *event*)

setTableActionBar(*self*)

getDefaultEntryActions(*self*)

Returns the list of actions available in our actionBar

getActions(*self*)

Returns the list of actions available in our actionBar

getAllActions(*self*, *view=None*)

Returns the list of actions available in the action bar.

showErrorMsg(*self*, *req=None*, *code=None*)

Removes all currently visible elements and displays an error message

onNextBatchNeeded(*self*)

Requests the next rows from the server and feed them to the table.

onAttach(*self*)

onDetach(*self*)

onDataChanged(*self*, *module*, **args*, ***kwargs*)

Refresh our view if element(s) in this module have changed

requestStructure(*self*)

receivedStructure(*self*, *resp*)

reloadData(*self*)

Removes all currently displayed data and refetches the first batch from the server.

setFilter(*self*, *filter*, *filterID=None*, *filterDescr=None*)

Applies a new filter.

setContext(*self*, *context*)

Applies a new context.

getFilter(*self*)

updateEmptyNotification(*self*)

onCompletion(*self*, *req*)

Pass the rows received to the datatable. :param req: The network request that succeed.

setFields(*self*, *fields*)

getFields(*self*)

onSelectionActivated(*self*, *table*, *selection*)

activateSelection(*self*)

static canHandle(*moduleName*, *moduleInfo*)

```
class vi.widgets.EditWidget(module, applicationType, key=0, node=None, skelType=None, clone=False,  
                             hashArgs=None, context=None, logAction='Entry saved!', skel=None, *args,  
                             **kwargs)
```

Bases: flare.html5.Div

appList = list

appHierarchy = hierarchy

appTree = tree

appSingleton = singleton

__editIdx = 0

onDetach(*self*)

onAttach(*self*)

onChange(*self*, *event*)

onBoneChange(*self*, *bone*)

showErrorMsg(*self*, *req=None*, *code=None*)

Removes all currently visible elements and displays an error message

reloadData(*self*)

_save(*self*, *data*)

Creates the actual NetworkService request used to transmit our data. If data is None, it fetches a clean add/edit form.

Parameters *data* (*dict* or *None*) – The values to transmit or None to fetch a new, clean add/edit form.

clear(*self*)

Removes all visible bones/forms/fieldsets.

closeOrContinue(*self*, *sender=None*)

doCloneHierarchy(*self*, *sender=None*)

cloneComplete(*self*, *req*)

setData(*self*, *request=None*, *data=None*, *askHierarchyCloning=True*)

Rebuilds the UI according to the skeleton received from server

Parameters

- **request** (*NetworkService*) – A finished NetworkService request
- **data** (*dict*) – The data received

doSave(*self*, *closeOnSuccess=False*, **args*, ***kwargs*)

Starts serializing and transmitting values to the server.

class `vi.widgets.ToolTip`(*shortText=""*, *longText=""*, **args*, ***kwargs*)

Bases: `flare.html5.Div`

Small utility class for providing tooltips

onClick(*self*, *event*)

_setDisabled(*self*, *disabled*)

_getDisabled(*self*)

class `vi.widgets.DataTable`(*_loadOnDisplay=False*, **args*, ***kwargs*)

Bases: `flare.html5.Div`

Provides kind of MVC on top of SelectTable.

recalcHeight(*self*, **args*, ***kwargs*)

setDataProvider(*self*, *obj*)

Register's 'obj' as the provider for this table. It must provide a `onNextBatchNeeded` function, which must fetch and feed new rows using `add()` or reset the `dataProvider` to `None` if no more rows are available. Notice: If the bottom of the table is reached, `onNextBatchNeeded` will only be called once. No further calls will be made until `add()` or `setDataProvider()` has been called afterwards.

onCursorMoved(*self*, *table*, *row*)

Ensure the table scrolls according to the position of its cursor

getRowCount(*self*)

Returns the total amount of rows currently known. :returns: int

add(*self*, *obj*)

Adds an row to the model :param *obj*: Dictionary of values for this row :type *obj*: dict

extend(*self*, *objList*)

Adds multiple rows at once. Much faster than calling add() multiple times.

testIfNextBatchNeededImmediately(*self*)

Test if we display enough entries so that our contents are scrollable. Otherwise, we'll never request a second batch

remove(*self*, *objOrIndex*)

Removes 'obj' from the table. 'obj' may be an row-index or an object recieved by any eventListener. It cannot be any original object passed to 'add' - it must be recieved by an eventListener!

clear(*self*, *keepModel=False*)

Flushes the whole table.

_renderObject(*self*, *obj*, *tableIsPrepared=False*)

Renders the object to into the table. Does nothing if the list of _shownFields is empty. :param *obj*: Dictionary of values for this row :type *obj*: dict

rebuildTable(*self*)

Rebuilds the entire table. Useful if something fundamental changed (ie. the cell renderer or the list of visible fields)

setShownFields(*self*, *fields*)

Sets the list of _shownFields. This causes the whole table to be rebuild. Be careful if calling this function often on a large table! :param *fields*: List of model-keys which will be displayed. :type *fields*: list

onScroll(*self*, *event*)

Check if we got a scroll event and need to fetch another set of rows from our dataProvider

onSelectionChanged(*self*, *table*, *rows*, **args*, *kwargs*)**

Re-emit the event. Maps row-numbers to actual models.

onSelectionActivated(*self*, *table*, *rows*)

Re-emit the event. Maps row-numbers to actual models.

onTableChanged(*self*, *table*, *rowCount*, **args*, *kwargs*)**

Re-emit the event.

getCurrentSelection(*self*)

Override the getCurrentSelection method to yield actual models, not row-numbers.

setCellRender(*self*, *field*, *render*)

Sets the render for cells of 'field' to render. A cell render receives the data for a given cell and returns the appropriate widget to display that data for the table.

setCellRenderers(*self*, *renders*)

Like setCellRender, but sets multiple renders at one. Much faster than calling setCellRender repeatedly.

activateSelection(*self*)

Emits the selectionActivated event if there's currently a selection

```

class vi.widgets.Search(*args, **kwargs)
    Bases: flare.html5.Div
    doSearch(self, *args, **kwargs)
    resetSearch(self)
    onKeyDown(self, event)
    resetLoadingState(self)
    reevaluate(self)
    focus(self)

class vi.widgets.SideBar(*args, **kwargs)
    Bases: flare.html5.Div
    onAttach(self)
    onDetach(self)
    setWidget(self, widget)
    getWidget(self)
    close(self, *args, **kwargs)

class vi.widgets.UserLogoutMsg(*args, **kwargs)
    Bases: flare.popup.Popup
    pollInterval = 120
    checkIntervall
    visibilityChanged(self, e)
    stopInterval(self)
    hideMessage(self)
        Make this popup invisible
    showMessage(self)
        Show this popup
    showLoginWindow(self, *args, **kwargs)
        Return to the login window.
    checkForSuspendResume(self, *args, **kwargs)
        Test if at least self.pollIntervall seconds have passed and query the server if
    startPolling(self, *args, **kwargs)
        Start querying the server
    onUserTestSuccess(self, req)
        We received a response from the server
    onUserTestFail(self, text, ns)
        Error retrieving the current user response from the server

```

```
class vi.widgets.TaskWidget(title)
    Bases: flare.popup.Popup

class vi.widgets.TaskSelectWidget
    Bases: TaskWidget

    getSelectedTask(self)

    setActiveTask(self)

    onChange(self, event)

    invokeTask(self, *args, **kwargs)

class vi.widgets.InternalEdit(skelStructure, values=None, errorInformation=None, readOnly=False,
                              context=None, defaultCat="", module=None, boneparams=None,
                              errorQueue=None, prefix=None)

    Bases: flare.html5.Div

    renderStructure(self, readOnly=False)

    serializeForPost(self, validityCheck=False)

    serializeForDocument(self)

    doSave(self, closeOnSuccess=False, *args, **kwargs)
        Starts serializing and transmitting our values to the server.

    unserialize(self, data=None)
        Applies the actual data to the bones.

    onChange(self, event)

    onKeyDown(self, event)

    performLogics(self)

class vi.widgets.HtmlEditor(*args, **kwargs)
    Bases: flare.html5.Textarea

    initSources = False

    _attachSummernote(self, retry=0)

    onAttach(self)

    onDetach(self)

    onEditorChange(self, e, *args, **kwargs)

    _getValue(self)

    _setValue(self, val)

    enable(self)

    disable(self)
```

```

class vi.widgets.ExportCsv(widget, selection, encoding=None, language=None, separator=None,
                           lineSeparator=None, *args, **kwargs)
    Bases: flare.html5.Progress
    nextChunk(self, cursor=None)
    nextChunkComplete(self, req)
    exportToFile(self)
    nextChunkFailure(self, req, code)
    replaceWithMessage(self, message, logClass='success')

class vi.widgets.ExportCsvStarter(widget, *args, **kwargs)
    Bases: flare.popup.Popup
    onExportBtnClick(self, *args, **kwargs)

class vi.widgets.TreeWidget(module, rootNode=None, node=None, context=None, *args, **kwargs)
    Bases: flare.html5.Div
    Base Widget that renders a tree.
    nodeWidget
    leafWidget
    requestStructure(self)
    receivedStructure(self, resp)
    setSelector(self, callback, multi=True, allow=None)
        Configures the widget as selector for a relationalBone and shows it.
    selectorReturn(self)
        Returns the current selection to the callback configured with setSelector.
    onKeyDown(self, event)
    onKeyUp(self, event)
    getActions(self)
        Returns a list of actions that are being set for the ActionBar. Override this to provide additional actions.
    clearSelection(self)
        Empties the current selection.
    extendSelection(self, element)
        Extends the current selection to element.
        This is normally done by clicking or tabbing on an element.
    activateSelection(self, element)
        Activates the current selection or element.
        An activation mostly is an action like selecting or editing an item. This is normally done by double-clicking
        an element.
    requestChildren(self, element)

```

showErrorMsg(*self*, *req=None*, *code=None*)

Removes all currently visible elements and displays an error message

onDataChanged(*self*, *module*, **args*, ***kwargs*)

onAttach(*self*)

onDetach(*self*)

itemForKey(*self*, *key*, *elem=None*)

Returns the HierarchyWidget displaying the entry with the given key. :param key: The key (id) of the item.
:type key: str :returns: HierarchyItem

onSetDefaultRootNode(*self*, *req*)

We requested the list of rootNodes for that module and that request just finished. Parse the response and set our rootNode to the first rootNode received.

setRootNode(*self*, *rootNode*, *node=None*)

Set the currently displayed hierarchy to 'rootNode'. :param rootNode: Key of the rootNode which children we shall display :type rootNode: str

reloadData(*self*)

Reload the data were displaying.

loadNode(*self*, *node*, *cursor=None*, *reqType=None*, *overrideParams=None*)

Fetch the (direct) children of the given node. Once the list is received, append them to their parent node.
:param node: Key of the node to fetch :type node: str

onRequestSucceeded(*self*, *req*)

The NetworkRequest for a (sub)node finished. Create a new HierarchyItem for each entry received and add them to our view

onDrop(*self*, *event*)

We got a drop event. Make that item a direct child of our rootNode

onDragOver(*self*, *event*)

Allow dropping children on the rootNode

getChildKey(*self*, *widget*)

Order by sortindex

static canHandle(*moduleName*, *moduleInfo*)

class **vi.widgets.TreeBrowserWidget**(*module*, *rootNode=None*, *node=None*, *context=None*, **args*, ***kwargs*)

Bases: [TreeWidget](#)

Base Widget that renders a tree.

leafWidget

nodeWidget

reloadData(*self*)

Reload the data were displaying.

rebuildPath(*self*)

Rebuild the displayed path-list.

onPathRequestSucceeded(*self, req*)

Rebuild the displayed path-list according to request data

activateSelection(*self, element*)

Activates the current selection or element.

An activation mostly is an action like selecting or editing an item. This is normally done by double-clicking an element.

static canHandle(*module, moduleInfo*)

class `vi.widgets.HierarchyWidget(*args, **kwargs)`

Bases: `vi.widgets.tree.TreeWidget`

A Hierarchy is a Tree without leaf distinction!

leafWidget

reloadData(*self*)

Reload the data were displaying.

reloadListWidget(*self*)

toggleListView(*self*)

setListView(*self, visible=False*)

showListView(*self*)

hideListView(*self*)

onSelectionChanged(*self, widget, selection, *args, **kwargs*)

static canHandle(*moduleName, moduleInfo*)

class `vi.widgets.FileWidget(module, rootNode=None, selectMode=None, node=None, context=None, *args, **kwargs)`

Bases: `vi.widgets.tree.TreeBrowserWidget`

Base Widget that renders a tree.

leafWidget

nodeWidget

searchWidget(*self*)

onStartSearch(*self, searchStr, *args, **kwargs*)

getChildKey(*self, widget*)

Derives a string used to sort the entries on each level

onDrop(*self, event*)

We got a drop event. Make that item a direct child of our rootNode

static canHandle(*module, moduleInfo*)

Submodules

`vi.admin`

Module Contents

Classes

<i>AdminScreen</i>	This is the screen superclass.
--------------------	--------------------------------

Attributes

<i>viInitializedEvent</i>

```
class vi.admin.AdminScreen(*args, **kwargs)
    Bases: vi.screen.Screen
    This is the screen superclass.
    It represents a basic screen and its functionality.
    onClick(self, event)
    reset(self)
    invoke(self)
        Is called to show the screen
    getCurrentUser(self)
    getCurrentUserSuccess(self, req)
    getCurrentUserFailure(self, req, code)
    startup(self)
    initializeViews(self)
    initializeConfig(self)
    appendNavList(self, NavList, target, parentInfo=())
    openView(self, name, icon, viewName, moduleName, actionName, data, focusView=True, append=False,
        target='mainNav')
    openNewMainView(self, name, icon, viewName, moduleName, actionName, data, focusView=True,
        append=False)
    openNewPopup(self, name, icon, viewName, moduleName, actionName, data, focusView=True,
        append=False)
    log(self, type, msg, icon=None, modul=None, action=None, key=None, data=None)
```

checkInitialHash(*self*, *args, **kwargs)

execCall(*self*, path, params=None)

Performs an execution call.

Parameters

- **path** – Path to the module and action
- **params** – Parameters passed to the module

stackWidget(*self*, widget, disableOtherWidgets=True)

We dont stack widgets anymore. We use now Popups.

removeWidget(*self*, widget)

switchFullscreen(*self*, fullscreen=True)

isFullscreen(*self*)

onError(*self*, req, code)

vi.admin.viInitializedEvent

vi.config

Module Contents

Functions

updateConf(_conf)

getConf()

Attributes

vi_conf

conf

vi.config.vi_conf

vi.config.updateConf(_conf)

vi.config.getConf()

vi.config.conf

vi.exception

Module Contents

exception vi.exception.InvalidBoneValueException

Bases: ValueError

Inappropriate argument value (of correct type).

vi.log

Module Contents

Classes

<i>logEntry</i>	PopOut Elements
<i>logA</i>	click handler for loglist
<i>logWidget</i>	
<i>LogButton</i>	
<i>Log</i>	Provides the "messaging" center displayed at the bottom of VI

Attributes

<i>iddbTableName</i>

vi.log.iddbTableName = vi_log3

class vi.log.logEntry(logObj=None)

Bases: flare.html5.Span

PopOut Elements

class vi.log.logA(logObj=None)

Bases: flare.html5.A

click handler for loglist

onClick(self, sender=None)

openEditor(self, key)

class vi.log.logWidget(logList)

Bases: flare.html5.Div

buildDataTable(self)

```

class vi.log.LogButton
    Bases: flare.html5.Div
    idbdata(self, event)
    cleanLog(self)
    cleanLogAction(self, event)
    renderPopOut(self)
    onClick(self, sender=None)
    openLog(self)
        apane = Pane( translate("Log"), closeable=True, iconClasses=[ "apptype_list"], collapseable=True
        )
        wg = logWidget(self.logsList )
        apane.addWidget(wg)
        conf["mainWindow"].addPane(apane) conf["mainWindow"].focusPane(apane)
    log(self, type, msg, icon=None, modul=None, action=None, key=None, data=None, date=None,
        onlyLoad=False)
    msgOverlay(self, logObj)
    removeInfo(self, wrap)
    reset(self)
    static canHandle(action)

class vi.log.Log
    Bases: flare.html5.Div
    Provides the "messaging" center displayed at the bottom of VI
    toggleMsgCenter(self, *args, **kwargs)
    log(self, type, msg, icon=None, date=None)
        Adds a message to the log :param type: The type of the message. :type type: "success", "error", "warning",
        "info", "progress" :param msg: The message to append :type msg: str
    removeNewCls(self, span)
    reset(self)

```

[vi.login](#)

Module Contents

Classes

LoginInputField

BaseLoginHandler

UserPasswordLoginHandler

GoogleAccountLoginHandler

*LoginScreen*This is the screen superclass.

```
class vi.login.LoginInputField(notifier, *args, **kwargs)
    Bases: flare.html5.Input
    onKeyPress(self, event)

class vi.login.BaseLoginHandler(loginScreen, *args, **kwargs)
    Bases: flare.html5.Li
    onClick(self, event)
    enable(self)
    disable(self)
    lock(self)
    unlock(self)
    login(self)
    reset(self)
    parseAnswer(self, req)

class vi.login.UserPasswordLoginHandler(loginScreen, *args, **kwargs)
    Bases: BaseLoginHandler
    cssname = userpassword
    onKeyPress(self, event)
    onLoginClick(self, sender=None)
    doLoginSuccess(self, req)
    doLoginFailure(self, req, code, *args, **kwargs)
    onVerifyClick(self, sender=None)
    doVerifySuccess(self, req)
    doVerifyFailure(self, *args, **kwargs)
    onSendClick(self, sender=None)
```

```

    reset(self)

    enable(self)

    focusLaterIdiot(self)

    static canHandle(method, secondFactor)

class vi.login.GoogleAccountLoginHandler(loginScreen, *args, **kwargs)
    Bases: BaseLoginHandler
    cssname = googleaccount

    onLoginClick(self, sender=None)

    static canHandle(method, secondFactor)

class vi.login.LoginScreen(*args, **kwargs)
    Bases: vi.screen.Screen
    This is the screen superclass.
    It represents a basic screen and its functionality.

    invoke(self, logout=False)
        Is called to show the screen

    onLogoutSuccess(self, *args, **kwargs)

    doShowLogin(self, req, code, *args, **kwargs)

    insufficientRights(self)

    doSkipLogin(self, req)

    onGetAuthMethodsSuccess(self, req)

    selectHandler(self, handler=None)

    onGetAuthMethodsFailure(self, *args, **kwargs)

    redirectNoAdmin(self)

```

`vi.pane`

Module Contents

Classes

<code>Pane</code>	Base class for Panes.
<code>GroupPane</code>	This pane groups subpanes; it cannot have direct childrens

```
class vi.pane.Pane(descr=None, iconURL=None, iconClasses=None, closeable=False, collapseable=True,
                  focusable=True, path=None)
```

Bases: flare.html5.Div

Base class for Panes.

A pane represents a entry in the module list as well as a list of widgets associated with this pane.

It is possible to stack panes on-top of each other. If a pane is active, `_all_` its child widgets are visible (through they might overlap).

```
__setattr__(self, key, value)
```

```
setImage(self, loading=False)
```

```
lock(self)
```

```
unlock(self)
```

```
setText(self, descr=None, iconURL=None, loading=False)
```

```
onBtnCloseReleased(self, *args, **kwargs)
```

```
addChildPane(self, pane)
```

Stack a pane under this one. It gets displayed as a subpane. :param pane: Another pane :type pane: pane

```
removeChildPane(self, pane)
```

Removes a subpane. :param pane: The pane to remove. Must be a direct child of this pane :type pane: Pane

```
onDetach(self)
```

```
addWidget(self, widget, disableOtherWidgets=True)
```

Adds a widget to this pane. Note: all widgets of a pane are visible at the same time! :param widget: The widget to add :type widget: Widget

```
rebuildChildrenClassInfo(self)
```

```
removeWidget(self, widget)
```

Removes a widget. :param widget: The widget to remove. Must be a direct child of this pane. :type widget: Widget

```
containsWidget(self, widget)
```

Tests wherever widget is a direct child of this pane. :returns: bool

```
onClick(self, event=None, *args, **kwargs)
```

```
expand(self)
```

```
collapse(self)
```

```
focus(self)
```

```
class vi.pane.GroupPane(*args, **kwargs)
```

Bases: [Pane](#)

This pane groups subpanes; it cannot have direct childrens

```
loadChildren(self)
```

```
DeferredLoadChildren(self, delay=1000)
```



```

onClick(self, event=None, *args, **kwargs)

expand(self)

collapse(self)

onFocus(self, event)

```

vi.priorityqueue

Module Contents

Classes

StartupQueue

Attributes

HandlerClassSelector

actionDelegateSelector

initialHashHandler

extendedSearchWidgetSelector

toplevelActionSelector

loginHandlerSelector

startupQueue

```

vi.priorityqueue.HandlerClassSelector
vi.priorityqueue.actionDelegateSelector
vi.priorityqueue.initialHashHandler
vi.priorityqueue.extendedSearchWidgetSelector
vi.priorityqueue.toplevelActionSelector
vi.priorityqueue.loginHandlerSelector
class vi.priorityqueue.StartupQueue
    Bases: object
    reset(self)

```

```
setFinalElem(self, elem)
insertElem(self, priority, elem)
run(self)
next(self)
```

```
vi.priorityqueue.startupQueue
```

```
vi.screen
```

Module Contents

Classes

Screen

This is the screen superclass.

```
class vi.screen.Screen(*args, **kwargs)
    Bases: flare.html5.Div
    This is the screen superclass.
    It represents a basic screen and its functionality.
    lock(self)
    unlock(self)
    invoke(self)
        Is called to show the screen
    remove(self)
        Remove the screen from its parent
    setTitle(self, title=None)
```

```
vi.serversideaction
```

Module Contents

Classes

ServerSideActionWdg

```
class vi.serversideaction.ServerSideActionWdg(module, handler, actionName, actionData)
    Bases: flare.button.Button
    switchDisabledState(self, disabled)
```

```

onAttach(self)
onDetach(self)
onSelectionChanged(self, table, selection, *args, **kwargs)
onClick(self, sender=None)
fetchNext(self)
fetchSucceeded(self, req)
fetchFailed(self, req, code)
resetLoadingState(self)

```

vi.utils

Module Contents

Classes

indexeddbConnector

indexeddb

Functions

<i>formatString</i> (<i>format</i> , <i>data</i> , <i>structure=None</i> , <i>language=None</i>)	Parses a string given by format and substitutes placeholders using values specified by data.
--	--

getImagePreview(*data*, *cropped=False*, *size=150*)

setPreventUnloading(*mode=True*)

mergeDict(*original*, *target*)

vi.utils.formatString(*format*, *data*, *structure=None*, *language=None*)

Parses a string given by format and substitutes placeholders using values specified by data.

vi.utils.getImagePreview(*data*, *cropped=False*, *size=150*)

vi.utils.setPreventUnloading(*mode=True*)

class vi.utils.indexeddbConnector(*dbName*, *version=None*)

dbResult

dbTransaction

```
    connect(self)
    db_error(self, event)
    db_blocked(self, events)
    db_version(self, event)
    db_onupgradeneeded(self, event)
    db_success(self, event)
class vi.utils.indexeddb(dbName, dbVersion=None)
    queue = []
    dbqueue = []
    connect(self)
    getList(self, name)
    _getList(self, event)
    getListKeys(self, name)
    _getListKey(self, event)
    db_success(self, event)
    dbAction(self, action, name, key=None, obj=None)
    _processDbUpdate(self, event)
    _processQueue(self, event)
    _writeToStore(self, item, dbResult, dbTransaction)
    _deleteFromStore(self, item, dbResult, dbTransaction)
    _updateToStore(self, item, dbResult, dbTransaction)
    _deleteObjectStore(self, item, dbResult, dbTransaction)
    _registerObjectStore(self, item, dbResult, dbTransaction)
vi.utils.mergeDict(original, target)
```

Package Contents

Classes

<i>LoginScreen</i>	This is the screen superclass.
<i>AdminScreen</i>	This is the screen superclass.
<i>Application</i>	

Functions

start()

Attributes

vi_conf

conf

s

a

d

sc

scinv

s

vi.vi_conf

class `vi.LoginScreen(*args, **kwargs)`

Bases: `vi.screen.Screen`

This is the screen superclass.

It represents a basic screen and its functionality.

invoke(*self*, *logout=False*)

Is called to show the screen

onLogoutSuccess(*self*, *args, **kwargs)

doShowLogin(*self*, *req*, *code*, *args, **kwargs)

insufficientRights(*self*)

doSkipLogin(*self*, *req*)

onGetAuthMethodsSuccess(*self*, *req*)

selectHandler(*self*, *handler=None*)

onGetAuthMethodsFailure(*self*, *args, **kwargs)

redirectNoAdmin(*self*)

```
class vi.AdminScreen(*args, **kwargs)
```

Bases: [vi.screen.Screen](#)

This is the screen superclass.

It represents a basic screen and its functionality.

```
onClick(self, event)
```

```
reset(self)
```

```
invoke(self)
```

Is called to show the screen

```
getCurrentUser(self)
```

```
getCurrentUserSuccess(self, req)
```

```
getCurrentUserFailure(self, req, code)
```

```
startup(self)
```

```
initializeViews(self)
```

```
initializeConfig(self)
```

```
appendNavList(self, NavList, target, parentInfo=())
```

```
openView(self, name, icon, viewName, moduleName, actionName, data, focusView=True, append=False, target='mainNav')
```

```
openNewMainView(self, name, icon, viewName, moduleName, actionName, data, focusView=True, append=False)
```

```
openNewPopup(self, name, icon, viewName, moduleName, actionName, data, focusView=True, append=False)
```

```
log(self, type, msg, icon=None, modul=None, action=None, key=None, data=None)
```

```
checkInitialHash(self, *args, **kwargs)
```

```
execCall(self, path, params=None)
```

Performs an execution call.

Parameters

- **path** – Path to the module and action
- **params** – Parameters passed to the module

```
stackWidget(self, widget, disableOtherWidgets=True)
```

We dont stack widgets anymore. We use now Popups.

```
removeWidget(self, widget)
```

```
switchFullscreen(self, fullscreen=True)
```

```
isFullscreen(self)
```

```
onError(self, req, code)
```

```

vi.conf
class vi.Application
    Bases: flare.html5.Div
    startup(self, *args, **kwargs)
    getVersionSuccess(self, req)
    getConfigSuccess(self, req)
    startupFailure(self, req, err)
    login(self, logout=False)
    admin(self)
    logout(self)
    setTitle(self, title=None)
    setPath(self, path='')
vi.start()
vi.s
vi.a
vi.d
vi.sc
vi.scinv
vi.s

```

1.4.2 webworker_scripts

WARNING! THIS SCRIPTS ARE USED IN A SANDBOX SO ALL DEPENDENCIES SHOULD BE HANDELED HERE!

THIS USES PYODIDE V0.17!

Module Contents

Classes

requestList

csvWriter

weblog

HTTPRequest

Wrapper around XMLHttpRequest

SimpleNetwork

Functions

<code>request(url, params=None, jsonResult=True, whitelist=('/list', '/view'))</code>	A very simple version of the NetworkService to request synchronous data
---	---

Attributes

`log`

`webworker_scripts.request(url, params=None, jsonResult=True, whitelist=('/list', '/view'))`

A very simple version of the NetworkService to request synchronous data

`class webworker_scripts.requestList(url, params=None, maxRequests=999)`

`requestData(self)`

`next(self)`

`running(self)`

`class webworker_scripts.csvWriter(delimiter=';')`

`delimiter = ;`

`writeRow(self, row)`

`writeRows(self, rows)`

`download(self, name='export.csv')`

`class webworker_scripts.weblog`

`static info(text)`

`static warn(text)`

`static error(text)`

`webworker_scripts.log`

`class webworker_scripts.HTTPRequest(method, url, callbackSuccess=None, callbackFailure=None, payload=None, content_type=None, asynchronous=True)`

Bases: object

Wrapper around XMLHttpRequest

`onReadyStateChange(self, *args, **kwargs)`

Internal callback.

`class webworker_scripts.SimpleNetwork`

Bases: object

`genReqStr(self, params)`

request(*self*, *url*, *params*)

onCompletion(*self*, *text*)

onError(*self*, *text*, *code*)

PYTHON MODULE INDEX

V

- [vi](#), 4
- [vi.actions](#), 4
- [vi.actions.context](#), 4
- [vi.actions.edit](#), 5
- [vi.actions.file](#), 6
- [vi.actions.hierarchy](#), 8
- [vi.actions.list](#), 10
- [vi.actions.list_order](#), 17
- [vi.actions.tree](#), 18
- [vi.admin](#), 62
- [vi.config](#), 63
- [vi.exception](#), 64
- [vi.framework](#), 19
- [vi.framework.components](#), 19
- [vi.framework.components.actionbar](#), 20
- [vi.framework.components.datatable](#), 20
- [vi.log](#), 64
- [vi.login](#), 65
- [vi.pane](#), 67
- [vi.priorityqueue](#), 69
- [vi.screen](#), 70
- [vi.serversideaction](#), 70
- [vi.sidebarwidgets](#), 24
- [vi.sidebarwidgets.filterselector](#), 24
- [vi.sidebarwidgets.internalpreview](#), 24
- [vi.translations](#), 25
- [vi.translations.de](#), 25
- [vi.translations.en](#), 25
- [vi.utils](#), 71
- [vi.views](#), 25
- [vi.views.edit](#), 25
- [vi.views.hierarchy](#), 26
- [vi.views.list](#), 26
- [vi.views.log](#), 27
- [vi.views.notfound](#), 27
- [vi.views.overview](#), 27
- [vi.views.singleton](#), 28
- [vi.views.tree](#), 28
- [vi.widgets](#), 29
- [vi.widgets.accordion](#), 29
- [vi.widgets.appnavigation](#), 30

- [vi.widgets.code](#), 31
- [vi.widgets.csvexport](#), 33
- [vi.widgets.edit](#), 33
- [vi.widgets.file](#), 35
- [vi.widgets.hierarchy](#), 37
- [vi.widgets.htmleditor](#), 37
- [vi.widgets.internaledit](#), 38
- [vi.widgets.list](#), 39
- [vi.widgets.preview](#), 41
- [vi.widgets.repeatdate](#), 42
- [vi.widgets.search](#), 42
- [vi.widgets.sidebar](#), 43
- [vi.widgets.table](#), 43
- [vi.widgets.task](#), 46
- [vi.widgets.tooltip](#), 47
- [vi.widgets.topbar](#), 47
- [vi.widgets.tree](#), 49
- [vi.widgets.userlogoutmsg](#), 51

W

- [webworker_scripts](#), 75

Symbols

__editIdx_ (vi.widgets.EditWidget attribute), 54
 __editIdx_ (vi.widgets.edit.EditWidget attribute), 34
 __editIdx_ (vi.widgets.repeatdate.RepeatDatePopup attribute), 42
 __setattr__ (vi.pane.Pane method), 68
 _attachCodemirror() (vi.widgets.code.Codemirror method), 32
 _attachSummernote() (vi.widgets.HtmlEditor method), 58
 _attachSummernote() (vi.widgets.htmleditor.HtmlEditor method), 38
 _deleteFromStore() (vi.utils.indexeddb method), 72
 _deleteObjectStore() (vi.utils.indexeddb method), 72
 _extraCols() (vi.framework.components.datatable.SelectTable method), 22
 _extraCols() (vi.widgets.table.SelectTable method), 44
 _getDisabled() (vi.widgets.ToolTip method), 55
 _getDisabled() (vi.widgets.tooltip.ToolTip method), 47
 _getList() (vi.utils.indexeddb method), 72
 _getListKey() (vi.utils.indexeddb method), 72
 _getValue() (vi.widgets.HtmlEditor method), 58
 _getValue() (vi.widgets.code.Codemirror method), 32
 _getValue() (vi.widgets.htmleditor.HtmlEditor method), 38
 _processDbUpdate() (vi.utils.indexeddb method), 72
 _processQueue() (vi.utils.indexeddb method), 72
 _registerObjectStore() (vi.utils.indexeddb method), 72
 _renderObject() (vi.framework.components.datatable.DataTable method), 22
 _renderObject() (vi.framework.components.datatable.ViewportListWidget method), 23
 _renderObject() (vi.widgets.DataTable method), 56
 _renderObject() (vi.widgets.table.DataTable method), 45
 _rowForEvent() (vi.framework.components.datatable.SelectTable method), 21
 _rowForEvent() (vi.widgets.table.SelectTable method), 44
 _save() (vi.widgets.EditWidget method), 55
 _save() (vi.widgets.edit.EditWidget method), 34
 _setDisabled() (vi.widgets.ToolTip method), 55
 _setDisabled() (vi.widgets.tooltip.ToolTip method), 47
 _setPage() (vi.widgets.list.ViewportListWidget method), 41
 _setValue() (vi.widgets.HtmlEditor method), 58
 _setValue() (vi.widgets.appnavigation.NavigationSeparator method), 31
 _setValue() (vi.widgets.code.Codemirror method), 32
 _setValue() (vi.widgets.htmleditor.HtmlEditor method), 38
 _updateToStore() (vi.utils.indexeddb method), 72
 _writeToStore() (vi.utils.indexeddb method), 72

A

Accordion (class in vi.widgets), 52
 Accordion (class in vi.widgets.accordion), 29
 AccordionSegment (class in vi.widgets.accordion), 29
 ActionBar (class in vi.framework.components.actionbar), 20
 actionDelegateSelector (in module vi.priorityqueue), 69
 activate() (vi.widgets.accordion.AccordionSegment method), 29
 activateCurrentSelection() (vi.framework.components.datatable.DataTable method), 23
 activateSelection() (vi.widgets.DataTable method), 56
 activateSelection() (vi.widgets.list.ListWidget method), 40
 activateSelection() (vi.widgets.ListWidget method), 54
 activateSelection() (vi.widgets.table.DataTable method), 46
 activateSelection() (vi.widgets.tree.TreeBrowserWidget method), 51
 activateSelection() (vi.widgets.tree.TreeWidget method), 49

- `activateSelection()` (*vi.widgets.TreeBrowserWidget method*), 61
 - `activateSelection()` (*vi.widgets.TreeWidget method*), 59
 - `add()` (*vi.framework.components.datatable.DataTable method*), 22
 - `add()` (*vi.framework.components.datatable.ViewportDataTable method*), 23
 - `add()` (*vi.widgets.DataTable method*), 56
 - `add()` (*vi.widgets.table.DataTable method*), 45
 - `AddAction` (*class in vi.actions.hierarchy*), 8
 - `AddAction` (*class in vi.actions.list*), 11
 - `addChildPane()` (*vi.pane.Pane method*), 68
 - `AddLeafAction` (*class in vi.actions.file*), 7
 - `AddLeafAction` (*class in vi.actions.tree*), 18
 - `addNavigationBlock()` (*vi.widgets.appnavigation.AppNavigation method*), 31
 - `addNavigationPoint()` (*vi.widgets.appnavigation.AppNavigation method*), 31
 - `addNavigationPointAfter()` (*vi.widgets.appnavigation.AppNavigation method*), 31
 - `AddNodeAction` (*class in vi.actions.file*), 6
 - `AddNodeAction` (*class in vi.actions.tree*), 18
 - `addSegment()` (*vi.widgets.Accordion method*), 52
 - `addSegment()` (*vi.widgets.accordion.Accordion method*), 29
 - `addSelectedRow()` (*vi.framework.components.datatable.SelectTable method*), 21
 - `addSelectedRow()` (*vi.widgets.table.SelectTable method*), 44
 - `addSeperator()` (*vi.widgets.appnavigation.Navigationblock method*), 31
 - `addToLog()` (*vi.widgets.code.PythonCode method*), 32
 - `addWidget()` (*vi.pane.Pane method*), 68
 - `addWidget()` (*vi.widgets.accordion.AccordionSegment method*), 29
 - `admin()` (*vi.Application method*), 75
 - `AdminScreen` (*class in vi*), 73
 - `AdminScreen` (*class in vi.admin*), 62
 - `allDeletedSuccess()` (*vi.actions.hierarchy.DeleteAction method*), 9
 - `allDeletedSuccess()` (*vi.actions.list.DeleteAction method*), 12
 - `allDeletedSuccess()` (*vi.actions.tree.DeleteAction method*), 19
 - `appendNavList()` (*vi.admin.AdminScreen method*), 62
 - `appendNavList()` (*vi.AdminScreen method*), 74
 - `appendSubChild()` (*vi.widgets.appnavigation.NavigationElement method*), 31
 - `appHierarchy` (*vi.widgets.edit.EditWidget attribute*), 34
 - `appHierarchy` (*vi.widgets.EditWidget attribute*), 54
 - `Application` (*class in vi*), 75
 - `appList` (*vi.widgets.edit.EditWidget attribute*), 34
 - `appList` (*vi.widgets.EditWidget attribute*), 54
 - `applyFilter()` (*vi.sidebarwidgets.filterselector.FilterSelector method*), 24
 - `AppNavigation` (*class in vi.widgets.appnavigation*), 31
 - `appSingleton` (*vi.widgets.edit.EditWidget attribute*), 34
 - `appSingleton` (*vi.widgets.EditWidget attribute*), 54
 - `appTree` (*vi.widgets.edit.EditWidget attribute*), 34
 - `appTree` (*vi.widgets.EditWidget attribute*), 54
 - `ArrowAction()` (*vi.widgets.appnavigation.NavigationElement method*), 31
- ## B
- `BaseLoginHandler` (*class in vi.login*), 66
 - `BreadcrumbNodeWidget` (*class in vi.widgets.tree*), 50
 - `BrowserLeafWidget` (*class in vi.widgets.tree*), 50
 - `BrowserNodeWidget` (*class in vi.widgets.tree*), 50
 - `buildAccordion()` (*vi.widgets.Accordion method*), 53
 - `buildAccordion()` (*vi.widgets.accordion.Accordion method*), 29
 - `buildDataTable()` (*vi.log.logWidget method*), 64
 - `buildSeperator()` (*vi.widgets.appnavigation.NavigationSeperator method*), 31
- ## C
- `CancelClose` (*class in vi.actions.edit*), 6
 - `canHandle()` (*vi.log.LogButton static method*), 65
 - `canHandle()` (*vi.login.GoogleAccountLoginHandler static method*), 67
 - `canHandle()` (*vi.login.UserPasswordLoginHandler static method*), 67
 - `canHandle()` (*vi.views.hierarchy.hierarchyHandler static method*), 26
 - `canHandle()` (*vi.views.list.listHandler static method*), 26
 - `canHandle()` (*vi.views.singleton.singletonHandler static method*), 28
 - `canHandle()` (*vi.views.tree.treeHandler static method*), 29
 - `canHandle()` (*vi.widgets.file.FileWidget static method*), 36
 - `canHandle()` (*vi.widgets.FileWidget static method*), 61
 - `canHandle()` (*vi.widgets.hierarchy.HierarchyWidget static method*), 37
 - `canHandle()` (*vi.widgets.HierarchyWidget static method*), 61
 - `canHandle()` (*vi.widgets.list.ListWidget static method*), 40
 - `canHandle()` (*vi.widgets.list.ViewportListWidget static method*), 41
 - `canHandle()` (*vi.widgets.ListWidget static method*), 54
 - `canHandle()` (*vi.widgets.topbar.Logout static method*), 48

- [canHandle\(\) \(vi.widgets.topbar.Scripiter static method\), 48](#)
[canHandle\(\) \(vi.widgets.topbar.Tasks static method\), 48](#)
[canHandle\(\) \(vi.widgets.topbar.UserState static method\), 48](#)
[canHandle\(\) \(vi.widgets.tree.TreeBrowserWidget static method\), 51](#)
[canHandle\(\) \(vi.widgets.tree.TreeWidget static method\), 50](#)
[canHandle\(\) \(vi.widgets.TreeBrowserWidget static method\), 61](#)
[canHandle\(\) \(vi.widgets.TreeWidget static method\), 60](#)
[checkErrors\(\) \(in module vi.widgets.internaledit\), 38](#)
[checkForSuspendResume\(\) \(vi.widgets.UserLogoutMsg method\), 57](#)
[checkForSuspendResume\(\) \(vi.widgets.userlogoutmsg.UserLogoutMsg method\), 51](#)
[checkInitialHash\(\) \(vi.admin.AdminScreen method\), 62](#)
[checkInitialHash\(\) \(vi.AdminScreen method\), 74](#)
[checkIntervall \(vi.widgets.UserLogoutMsg attribute\), 57](#)
[checkIntervall \(vi.widgets.userlogoutmsg.UserLogoutMsg attribute\), 51](#)
[checkVisibility\(\) \(vi.widgets.accordion.AccordionSegment method\), 29](#)
[cleanLog\(\) \(vi.log.LogButton method\), 65](#)
[cleanLogAction\(\) \(vi.log.LogButton method\), 65](#)
[clear\(\) \(vi.framework.components.datatable.DataTable method\), 22](#)
[clear\(\) \(vi.framework.components.datatable.SelectTable method\), 21](#)
[clear\(\) \(vi.framework.components.datatable.ViewportDataTable method\), 23](#)
[clear\(\) \(vi.widgets.Accordion method\), 52](#)
[clear\(\) \(vi.widgets.accordion.Accordion method\), 29](#)
[clear\(\) \(vi.widgets.DataTable method\), 56](#)
[clear\(\) \(vi.widgets.edit.EditWidget method\), 34](#)
[clear\(\) \(vi.widgets.EditWidget method\), 55](#)
[clear\(\) \(vi.widgets.repeatdate.RepeatDatePopup method\), 42](#)
[clear\(\) \(vi.widgets.table.DataTable method\), 45](#)
[clear\(\) \(vi.widgets.table.SelectTable method\), 44](#)
[clearSelection\(\) \(vi.widgets.tree.TreeWidget method\), 49](#)
[clearSelection\(\) \(vi.widgets.TreeWidget method\), 59](#)
[CloneAction \(class in vi.actions.hierarchy\), 9](#)
[CloneAction \(class in vi.actions.list\), 12](#)
[cloneComplete\(\) \(vi.widgets.edit.EditWidget method\), 34](#)
[cloneComplete\(\) \(vi.widgets.EditWidget method\), 55](#)
[close\(\) \(vi.widgets.SideBar method\), 57](#)
[close\(\) \(vi.widgets.sidebar.SideBar method\), 43](#)
[CloseAction \(class in vi.actions.list\), 13](#)
[closeMessage\(\) \(vi.widgets.file.MultiUploader method\), 36](#)
[closeOrContinue\(\) \(vi.widgets.edit.EditWidget method\), 34](#)
[closeOrContinue\(\) \(vi.widgets.EditWidget method\), 55](#)
[CodeHelpPopup \(class in vi.widgets.code\), 32](#)
[Codemirror \(class in vi.widgets.code\), 32](#)
[CodePopup \(class in vi.widgets.code\), 32](#)
[collapse\(\) \(vi.pane.GroupPane method\), 69](#)
[collapse\(\) \(vi.pane.Pane method\), 68](#)
[CompoundFilter \(class in vi.sidebarwidgets.filterselector\), 24](#)
[conf \(in module vi\), 74](#)
[conf \(in module vi.config\), 63](#)
[connect\(\) \(vi.utils.indexeddb method\), 72](#)
[connect\(\) \(vi.utils.indexeddbConnector method\), 71](#)
[containsWidget\(\) \(vi.pane.Pane method\), 68](#)
[ContextAction \(class in vi.actions.context\), 4](#)
[createDir\(\) \(vi.actions.file.AddNodeAction method\), 7](#)
[CreateRecurrentAction \(class in vi.actions.list\), 16](#)
[cssname \(vi.login.GoogleAccountLoginHandler attribute\), 67](#)
[cssname \(vi.login.UserPasswordLoginHandler attribute\), 66](#)
[csswriter \(class in webworker_scripts\), 76](#)
- ## D
- [d \(in module vi\), 75](#)
[DataTable \(class in vi.framework.components.datatable\), 22](#)
[DataTable \(class in vi.widgets\), 55](#)
[DataTable \(class in vi.widgets.table\), 45](#)
[dbBlocked\(\) \(vi.utils.indexeddbConnector method\), 72](#)
[db_error\(\) \(vi.utils.indexeddbConnector method\), 72](#)
[db_onupgradeneeded\(\) \(vi.utils.indexeddbConnector method\), 72](#)
[db_success\(\) \(vi.utils.indexeddb method\), 72](#)
[db_success\(\) \(vi.utils.indexeddbConnector method\), 72](#)
[db_version\(\) \(vi.utils.indexeddbConnector method\), 72](#)
[dbAction\(\) \(vi.utils.indexeddb method\), 72](#)
[dbqueue \(vi.utils.indexeddb attribute\), 72](#)
[dbResult \(vi.utils.indexeddbConnector attribute\), 71](#)
[dbTransaction \(vi.utils.indexeddbConnector attribute\), 71](#)
[deactivate\(\) \(vi.widgets.accordion.AccordionSegment method\), 29](#)
[DeferredLoadChildren\(\) \(vi.pane.GroupPane method\), 68](#)
[DeleteAction \(class in vi.actions.hierarchy\), 9](#)
[DeleteAction \(class in vi.actions.list\), 12](#)
[DeleteAction \(class in vi.actions.tree\), 18](#)
[deletedFailed\(\) \(vi.actions.list.DeleteAction method\), 12](#)

deletedSuccess() (vi.actions.list.DeleteAction method), 12
 delimiter (webworker_scripts.csvWriter attribute), 76
 disable() (vi.login.BaseLoginHandler method), 66
 disable() (vi.widgets.HtmlEditor method), 58
 disable() (vi.widgets.htmleditor.HtmlEditor method), 38
 disableViUnloadingWarning() (vi.actions.file.DownloadAction method), 8
 doApply() (vi.actions.list.SelectFieldsPopup method), 13
 doCancel() (vi.actions.list.SelectFieldsPopup method), 13
 doCloneHierarchy() (vi.widgets.edit.EditWidget method), 34
 doCloneHierarchy() (vi.widgets.EditWidget method), 55
 doClose() (vi.widgets.preview.Preview method), 41
 doDelete() (vi.actions.hierarchy.DeleteAction method), 9
 doDelete() (vi.actions.list.DeleteAction method), 12
 doDelete() (vi.actions.tree.DeleteAction method), 19
 doDownload() (vi.actions.file.DownloadAction method), 8
 doInvertSelection() (vi.actions.list.SelectFieldsPopup method), 14
 doLoginFailure() (vi.login.UserPasswordLoginHandler method), 66
 doLoginSuccess() (vi.login.UserPasswordLoginHandler method), 66
 doMarkPayed() (vi.actions.list_order.ShopMarkAction method), 17
 doSave() (vi.widgets.edit.EditWidget method), 34
 doSave() (vi.widgets.EditWidget method), 55
 doSave() (vi.widgets.InternalEdit method), 58
 doSave() (vi.widgets.internaedit.InternalEdit method), 39
 doSave() (vi.widgets.repeatdate.RepeatDatePopup method), 42
 doSearch() (vi.widgets.Search method), 57
 doSearch() (vi.widgets.search.Search method), 42
 doSelectAll() (vi.actions.list.SelectFieldsPopup method), 13
 doSetFields() (vi.actions.list.SelectFieldsPopup method), 13
 doShowLogin() (vi.login.LoginScreen method), 67
 doShowLogin() (vi.LoginScreen method), 73
 doSkipLogin() (vi.login.LoginScreen method), 67
 doSkipLogin() (vi.LoginScreen method), 73
 doUnselectAll() (vi.actions.list.SelectFieldsPopup method), 13
 doVerifyFailure() (vi.login.UserPasswordLoginHandler method), 66
 doVerifySuccess() (vi.login.UserPasswordLoginHandler method), 66
 download() (vi.widgets.file.FilePreviewImage method), 35
 download() (webworker_scripts.csvWriter method), 76
 DownloadAction (class in vi.actions.file), 7
 dropTableContent() (vi.framework.components.datatable.SelectTable method), 22

E

EditAction (class in vi.actions.file), 7
 EditAction (class in vi.actions.hierarchy), 8
 EditAction (class in vi.actions.list), 11
 EditAction (class in vi.actions.tree), 18
 editDir() (vi.actions.file.EditAction method), 7
 editHandler (class in vi.views.edit), 25
 editHandlerWidget (class in vi.views.edit), 25
 EditWidget (class in vi.widgets), 54
 EditWidget (class in vi.widgets.edit), 34
 enable() (vi.login.BaseLoginHandler method), 66
 enable() (vi.login.UserPasswordLoginHandler method), 67
 enable() (vi.widgets.HtmlEditor method), 58
 enable() (vi.widgets.htmleditor.HtmlEditor method), 38
 enableViUnloadingWarning() (vi.actions.file.DownloadAction method), 8
 EntryIcon() (vi.widgets.file.FileLeafWidget method), 36
 error() (webworker_scripts.weblog static method), 76
 execCall() (vi.admin.AdminScreen method), 63
 execCall() (vi.AdminScreen method), 74
 ExecuteSingleton (class in vi.actions.edit), 5
 expand() (vi.pane.GroupPane method), 69
 expand() (vi.pane.Pane method), 68
 ExportCsv (class in vi.widgets), 58
 ExportCsv (class in vi.widgets.csvexport), 33
 ExportCsvAction (class in vi.actions.list), 16
 ExportCsvStarter (class in vi.widgets), 59
 ExportCsvStarter (class in vi.widgets.csvexport), 33
 exportToFile() (vi.widgets.csvexport.ExportCsv method), 33
 exportToFile() (vi.widgets.ExportCsv method), 59
 extend() (vi.framework.components.datatable.DataTable method), 22
 extend() (vi.framework.components.datatable.ViewportDataTable method), 23
 extend() (vi.widgets.DataTable method), 56
 extend() (vi.widgets.table.DataTable method), 45
 extendedSearchWidgetSelector (in module vi.priorityqueue), 69
 extendSelection() (vi.widgets.tree.TreeWidget method), 49

- `extendSelection()` (*vi.widgets.TreeWidget method*), 59
- ## F
- `fetchFailed()` (*vi.serversideaction.ServerSideActionWdg method*), 71
- `fetchNext()` (*vi.serversideaction.ServerSideActionWdg method*), 71
- `fetchSucceeded()` (*vi.serversideaction.ServerSideActionWdg method*), 71
- `FileImagePopup` (*class in vi.widgets.file*), 35
- `FileLeafWidget` (*class in vi.widgets.file*), 36
- `FileNodeWidget` (*class in vi.widgets.file*), 36
- `FilePreviewImage` (*class in vi.widgets.file*), 35
- `FileSelectUploader` (*class in vi.actions.file*), 6
- `FileWidget` (*class in vi.widgets*), 61
- `FileWidget` (*class in vi.widgets.file*), 36
- `FilterSelector` (*class in vi.sidebarwidgets.filterselector*), 24
- `findText()` (*vi.actions.list.PageFindAction method*), 15
- `focus()` (*vi.pane.Pane method*), 68
- `focus()` (*vi.sidebarwidgets.filterselector.CompoundFilter method*), 24
- `focus()` (*vi.widgets.Search method*), 57
- `focus()` (*vi.widgets.search.Search method*), 43
- `focusLaterIdiot()` (*vi.login.UserPasswordLoginHandler method*), 67
- `focusRow()` (*vi.framework.components.datatable.SelectTable method*), 21
- `focusRow()` (*vi.widgets.table.SelectTable method*), 44
- `formatString()` (*in module vi.utils*), 71
- ## G
- `genReqStr()` (*webworker_scripts.SimpleNetwork method*), 76
- `getActions()` (*vi.framework.components.actionbar.ActionBar method*), 20
- `getActions()` (*vi.widgets.list.ListWidget method*), 40
- `getActions()` (*vi.widgets.ListWidget method*), 53
- `getActions()` (*vi.widgets.tree.TreeWidget method*), 49
- `getActions()` (*vi.widgets.TreeWidget method*), 59
- `getAllActions()` (*vi.widgets.list.ListWidget method*), 40
- `getAllActions()` (*vi.widgets.ListWidget method*), 53
- `getChildKey()` (*vi.widgets.file.FileWidget method*), 36
- `getChildKey()` (*vi.widgets.FileWidget method*), 61
- `getChildKey()` (*vi.widgets.tree.TreeWidget method*), 50
- `getChildKey()` (*vi.widgets.TreeWidget method*), 60
- `getConf()` (*in module vi.config*), 63
- `getConfigSuccess()` (*vi.Application method*), 75
- `getCurrentSelection()` (*vi.framework.components.datatable.DataTable method*), 23
- `getCurrentSelection()` (*vi.framework.components.datatable.SelectTable method*), 21
- `getCurrentSelection()` (*vi.widgets.DataTable method*), 56
- `getCurrentSelection()` (*vi.widgets.table.DataTable method*), 46
- `getCurrentSelection()` (*vi.widgets.table.SelectTable method*), 44
- `getCurrentUser()` (*vi.admin.AdminScreen method*), 62
- `getCurrentUser()` (*vi.AdminScreen method*), 74
- `getCurrentUserFailure()` (*vi.admin.AdminScreen method*), 62
- `getCurrentUserFailure()` (*vi.AdminScreen method*), 74
- `getCurrentUserSuccess()` (*vi.admin.AdminScreen method*), 62
- `getCurrentUserSuccess()` (*vi.AdminScreen method*), 74
- `getDefaultEntryActions()` (*vi.widgets.list.ListWidget method*), 40
- `getDefaultEntryActions()` (*vi.widgets.ListWidget method*), 53
- `getFields()` (*vi.widgets.list.ListWidget method*), 40
- `getFields()` (*vi.widgets.ListWidget method*), 54
- `getFilter()` (*vi.widgets.list.ListWidget method*), 40
- `getFilter()` (*vi.widgets.ListWidget method*), 54
- `getImagePreview()` (*in module vi.utils*), 71
- `getIndexByTr()` (*vi.framework.components.datatable.SelectTable method*), 21
- `getIndexByTr()` (*vi.widgets.table.SelectTable method*), 44
- `getList()` (*vi.utils.indexeddb method*), 72
- `getListKeys()` (*vi.utils.indexeddb method*), 72
- `getNavigationPoint()` (*vi.widgets.appnavigation.AppNavigation method*), 31
- `getPreviousNavigationPoint()` (*vi.widgets.appnavigation.AppNavigation method*), 31
- `getRowCount()` (*vi.framework.components.datatable.DataTable method*), 22
- `getRowCount()` (*vi.widgets.DataTable method*), 55
- `getRowCount()` (*vi.widgets.table.DataTable method*), 45
- `getSelectedTask()` (*vi.widgets.task.TaskSelectWidget method*), 47
- `getSelectedTask()` (*vi.widgets.TaskSelectWidget method*), 58
- `getTrByIndex()` (*vi.framework.components.datatable.SelectTable method*), 21
- `getTrByIndex()` (*vi.widgets.table.SelectTable method*), 44
- `getVersionSuccess()` (*vi.Application method*), 75
- `getWidget()` (*vi.widgets.SideBar method*), 57

`getWidget()` (*vi.widgets.sidebar.SideBar* method), 43
GoogleAccountLoginHandler (class in *vi.login*), 67
GroupPane (class in *vi.pane*), 68

H

`handleFile()` (*vi.widgets.file.MultiUploader* method), 36
HandlerClassSelector (in module *vi.priorityqueue*), 69
`hideListView()` (*vi.widgets.hierarchy.HierarchyWidget* method), 37
`hideListView()` (*vi.widgets.HierarchyWidget* method), 61
`hideMessage()` (*vi.widgets.UserLogoutMsg* method), 57
`hideMessage()` (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), 51
hierarchyHandler (class in *vi.views.hierarchy*), 26
hierarchyHandlerWidget (class in *vi.views.hierarchy*), 26
HierarchyWidget (class in *vi.widgets*), 61
HierarchyWidget (class in *vi.widgets.hierarchy*), 37
HtmlEditor (class in *vi.widgets*), 58
HtmlEditor (class in *vi.widgets.htmleditor*), 38
HttpRequest (class in *webworker_scripts*), 76

I

`idbdata()` (*vi.log.LogButton* method), 65
`idbTableName` (in module *vi.log*), 64
indexeddb (class in *vi.utils*), 72
indexeddbConnector (class in *vi.utils*), 71
`info()` (*webworker_scripts.weblog* static method), 76
initialHashHandler (in module *vi.priorityqueue*), 69
`initializeConfig()` (*vi.admin.AdminScreen* method), 62
`initializeConfig()` (*vi.AdminScreen* method), 74
`initializeViews()` (*vi.admin.AdminScreen* method), 62
`initializeViews()` (*vi.AdminScreen* method), 74
`initSources` (*vi.widgets.HtmlEditor* attribute), 58
`initSources` (*vi.widgets.htmleditor.HtmlEditor* attribute), 38
`initWidget()` (*vi.views.edit.editHandlerWidget* method), 26
`initWidget()` (*vi.views.hierarchy.hierarchyHandlerWidget* method), 26
`initWidget()` (*vi.views.list.listHandlerWidget* method), 26
`initWidget()` (*vi.views.log.logHandlerWidget* method), 27
`initWidget()` (*vi.views.notfound.NotFoundWidget* method), 27
`initWidget()` (*vi.views.overview.OverviewWidget* method), 28

`initWidget()` (*vi.views.singleton.singletonHandlerWidget* method), 28
`initWidget()` (*vi.views.tree.treeHandlerWidget* method), 29
`insertElem()` (*vi.priorityqueue.StartupQueue* method), 70
`insertText()` (*vi.widgets.code.Codemirror* method), 32
`insufficientRights()` (*vi.login.LoginScreen* method), 67
`insufficientRights()` (*vi.LoginScreen* method), 73
InternalEdit (class in *vi.widgets*), 58
InternalEdit (class in *vi.widgets.internaedit*), 38
InternalPreview (class in *vi.sidebarwidgets.internalpreview*), 24
InvalidBoneValueException, 64
`invertSelection()` (*vi.framework.components.datatable.SelectTable* method), 22
`invertSelection()` (*vi.widgets.table.SelectTable* method), 45
`invoke()` (*vi.admin.AdminScreen* method), 62
`invoke()` (*vi.AdminScreen* method), 74
`invoke()` (*vi.login.LoginScreen* method), 67
`invoke()` (*vi.LoginScreen* method), 73
`invoke()` (*vi.screen.Screen* method), 70
`invoke()` (*vi.widgets.topbar.TopBarWidget* method), 47
`invoke()` (*vi.widgets.TopBarWidget* method), 53
`invokeTask()` (*vi.widgets.task.TaskSelectWidget* method), 47
`invokeTask()` (*vi.widgets.TaskSelectWidget* method), 58
`isActive()` (*vi.widgets.accordion.AccordionSegment* method), 29
`isFullscreen()` (*vi.admin.AdminScreen* method), 63
`isFullscreen()` (*vi.AdminScreen* method), 74
`isSuitableFor()` (*vi.actions.context.ContextAction* static method), 5
`isSuitableFor()` (*vi.actions.edit.CancelClose* static method), 6
`isSuitableFor()` (*vi.actions.edit.ExecuteSingleton* static method), 5
`isSuitableFor()` (*vi.actions.edit.Refresh* static method), 6
`isSuitableFor()` (*vi.actions.edit.SaveClose* static method), 6
`isSuitableFor()` (*vi.actions.edit.SaveContinue* static method), 5
`isSuitableFor()` (*vi.actions.edit.SaveSingleton* static method), 5
`isSuitableFor()` (*vi.actions.file.AddLeafAction* static method), 7
`isSuitableFor()` (*vi.actions.file.AddNodeAction* static method), 7
`isSuitableFor()` (*vi.actions.file.DownloadAction* static method), 7
`isSuitableFor()` (*vi.actions.file.EditAction* static

method), 7

isSuitableFor() (vi.actions.hierarchy.AddAction static method), 8

isSuitableFor() (vi.actions.hierarchy.CloneAction static method), 9

isSuitableFor() (vi.actions.hierarchy.DeleteAction static method), 9

isSuitableFor() (vi.actions.hierarchy.EditAction static method), 8

isSuitableFor() (vi.actions.hierarchy.ListViewAction static method), 10

isSuitableFor() (vi.actions.hierarchy.ReloadAction static method), 9

isSuitableFor() (vi.actions.hierarchy.SelectRootNode static method), 10

isSuitableFor() (vi.actions.list.AddAction static method), 11

isSuitableFor() (vi.actions.list.CloneAction static method), 12

isSuitableFor() (vi.actions.list.CloseAction static method), 13

isSuitableFor() (vi.actions.list.CreateRecurrentAction static method), 16

isSuitableFor() (vi.actions.list.DeleteAction static method), 12

isSuitableFor() (vi.actions.list.EditAction static method), 12

isSuitableFor() (vi.actions.list.ExportCsvAction static method), 16

isSuitableFor() (vi.actions.list.ListPreviewAction static method), 13

isSuitableFor() (vi.actions.list.ListPreviewInlineAction static method), 13

isSuitableFor() (vi.actions.list.ListSelectFilterAction static method), 16

isSuitableFor() (vi.actions.list.LoadAllAction static method), 15

isSuitableFor() (vi.actions.list.LoadNextBatchAction static method), 15

isSuitableFor() (vi.actions.list.PageFindAction static method), 15

isSuitableFor() (vi.actions.list.ReloadAction static method), 14

isSuitableFor() (vi.actions.list.SelectAction static method), 13

isSuitableFor() (vi.actions.list.SelectAllAction static method), 16

isSuitableFor() (vi.actions.list.SelectFieldsAction static method), 14

isSuitableFor() (vi.actions.list.SelectInvertAction static method), 16

isSuitableFor() (vi.actions.list.SetPageRowAmountAction static method), 15

isSuitableFor() (vi.actions.list.TableItems static method), 14

isSuitableFor() (vi.actions.list.TableNextPage static method), 14

isSuitableFor() (vi.actions.list.TablePrevPage static method), 14

isSuitableFor() (vi.actions.list.UnSelectAllAction static method), 16

isSuitableFor() (vi.actions.list_order.ShopMarkCanceledAction static method), 17

isSuitableFor() (vi.actions.list_order.ShopMarkPayedAction static method), 17

isSuitableFor() (vi.actions.list_order.ShopMarkSentAction static method), 17

isSuitableFor() (vi.actions.tree.AddLeafAction static method), 18

isSuitableFor() (vi.actions.tree.AddNodeAction static method), 18

isSuitableFor() (vi.actions.tree.DeleteAction static method), 19

isSuitableFor() (vi.actions.tree.EditAction static method), 18

isSuitableFor() (vi.actions.tree.ReloadAction static method), 19

isSuitableFor() (vi.actions.tree.SelectRootNode static method), 19

isSuitableFor() (vi.widgets.htmleditor.TextInsertImageAction static method), 37

itemForKey() (vi.widgets.tree.TreeWidget method), 50

itemForKey() (vi.widgets.TreeWidget method), 60

K

killClick() (vi.widgets.code.Scripiter method), 32

L

leafWidget (vi.widgets.file.FileWidget attribute), 36

leafWidget (vi.widgets.FileWidget attribute), 61

leafWidget (vi.widgets.hierarchy.HierarchyWidget attribute), 37

leafWidget (vi.widgets.HierarchyWidget attribute), 61

leafWidget (vi.widgets.tree.TreeBrowserWidget attribute), 51

leafWidget (vi.widgets.tree.TreeWidget attribute), 49

leafWidget (vi.widgets.TreeBrowserWidget attribute), 60

leafWidget (vi.widgets.TreeWidget attribute), 59

listHandler (class in vi.views.list), 26

listHandlerWidget (class in vi.views.list), 26

ListPreviewAction (class in vi.actions.list), 13

ListPreviewInlineAction (class in vi.actions.list), 13

ListSelectFilterAction (class in vi.actions.list), 15

ListViewAction (class in vi.actions.hierarchy), 10

ListWidget (class in vi.widgets), 53

ListWidget (class in vi.widgets.list), 39

lngDe (in module vi.translations), 25

lngDe (in module *vi.translations.de*), 25
lngEn (in module *vi.translations*), 25
lngEn (in module *vi.translations.en*), 25
LoadAllAction (class in *vi.actions.list*), 15
loadAllRows() (*vi.actions.list.LoadAllAction* method), 15
loadChildren() (*vi.pane.GroupPane* method), 68
LoadNextBatchAction (class in *vi.actions.list*), 15
loadnextPages() (*vi.actions.list.LoadNextBatchAction* method), 15
loadNode() (*vi.widgets.tree.TreeWidget* method), 50
loadNode() (*vi.widgets.TreeWidget* method), 60
lock() (*vi.login.BaseLoginHandler* method), 66
lock() (*vi.pane.Pane* method), 68
lock() (*vi.screen.Screen* method), 70
Log (class in *vi.log*), 65
log (in module *webworker_scripts*), 76
log() (*vi.admin.AdminScreen* method), 62
log() (*vi.AdminScreen* method), 74
log() (*vi.log.Log* method), 65
log() (*vi.log.LogButton* method), 65
logA (class in *vi.log*), 64
LogButton (class in *vi.log*), 64
logEntry (class in *vi.log*), 64
logHandler (class in *vi.views.log*), 27
logHandlerWidget (class in *vi.views.log*), 27
login() (*vi.Application* method), 75
login() (*vi.login.BaseLoginHandler* method), 66
loginHandlerSelector (in module *vi.priorityqueue*), 69
LoginInputField (class in *vi.login*), 66
LoginScreen (class in *vi*), 73
LoginScreen (class in *vi.login*), 67
Logout (class in *vi.widgets.topbar*), 48
logout() (*vi.Application* method), 75
logout() (*vi.widgets.topbar.Logout* method), 48
logWidget (class in *vi.log*), 64

M

mergeDict() (in module *vi.utils*), 72
module
 vi, 4
 vi.actions, 4
 vi.actions.context, 4
 vi.actions.edit, 5
 vi.actions.file, 6
 vi.actions.hierarchy, 8
 vi.actions.list, 10
 vi.actions.list_order, 17
 vi.actions.tree, 18
 vi.admin, 62
 vi.config, 63
 vi.exception, 64
 vi.framework, 19

vi.framework.components, 19
 vi.framework.components.actionbar, 20
 vi.framework.components.datatable, 20
 vi.log, 64
 vi.login, 65
 vi.pane, 67
 vi.priorityqueue, 69
 vi.screen, 70
 vi.serversideaction, 70
 vi.sidebarwidgets, 24
 vi.sidebarwidgets.filterselector, 24
 vi.sidebarwidgets.internalpreview, 24
 vi.translations, 25
 vi.translations.de, 25
 vi.translations.en, 25
 vi.utils, 71
 vi.views, 25
 vi.views.edit, 25
 vi.views.hierarchy, 26
 vi.views.list, 26
 vi.views.log, 27
 vi.views.notfound, 27
 vi.views.overview, 27
 vi.views.singleton, 28
 vi.views.tree, 28
 vi.widgets, 29
 vi.widgets.accordion, 29
 vi.widgets.appnavigation, 30
 vi.widgets.code, 31
 vi.widgets.csvexport, 33
 vi.widgets.edit, 33
 vi.widgets.file, 35
 vi.widgets.hierarchy, 37
 vi.widgets.htmleditor, 37
 vi.widgets.internaedit, 38
 vi.widgets.list, 39
 vi.widgets.preview, 41
 vi.widgets.repeatdate, 42
 vi.widgets.search, 42
 vi.widgets.sidebar, 43
 vi.widgets.table, 43
 vi.widgets.task, 46
 vi.widgets.tooltip, 47
 vi.widgets.topbar, 47
 vi.widgets.tree, 49
 vi.widgets.userlogoutmsg, 51
 webworker_scripts, 75
msgOverlay() (*vi.log.LogButton* method), 65
MultiUploader (class in *vi.widgets.file*), 36

N

navigationAction() (*vi.widgets.appnavigation.NavigationElement* method), 30

Navigationblock (class in *vi.widgets.appnavigation*), 31
 NavigationElement (class in *vi.widgets.appnavigation*), 30
 NavigationSeperator (class in *vi.widgets.appnavigation*), 31
 next() (*vi.priorityqueue.StartupQueue* method), 70
 next() (*webworker_scripts.requestList* method), 76
 nextChunk() (*vi.widgets.csvexport.ExportCsv* method), 33
 nextChunk() (*vi.widgets.ExportCsv* method), 59
 nextChunkComplete() (*vi.widgets.csvexport.ExportCsv* method), 33
 nextChunkComplete() (*vi.widgets.ExportCsv* method), 59
 nextChunkFailure() (*vi.widgets.csvexport.ExportCsv* method), 33
 nextChunkFailure() (*vi.widgets.ExportCsv* method), 59
 nodeWidget (*vi.widgets.file.FileWidget* attribute), 36
 nodeWidget (*vi.widgets.FileWidget* attribute), 61
 nodeWidget (*vi.widgets.tree.TreeBrowserWidget* attribute), 51
 nodeWidget (*vi.widgets.tree.TreeWidget* attribute), 49
 nodeWidget (*vi.widgets.TreeBrowserWidget* attribute), 60
 nodeWidget (*vi.widgets.TreeWidget* attribute), 59
 NotFound (class in *vi.views.notfound*), 27
 NotFoundWidget (class in *vi.views.notfound*), 27
O
 onActiveNavigationChanged() (*vi.widgets.appnavigation.NavigationElement* method), 31
 onActiveViewChanged() (*vi.widgets.appnavigation.NavigationElement* method), 30
 onAttach() (*vi.actions.context.ContextAction* method), 5
 onAttach() (*vi.actions.file.DownloadAction* method), 7
 onAttach() (*vi.actions.file.EditAction* method), 7
 onAttach() (*vi.actions.hierarchy.CloneAction* method), 9
 onAttach() (*vi.actions.hierarchy.DeleteAction* method), 9
 onAttach() (*vi.actions.hierarchy.EditAction* method), 8
 onAttach() (*vi.actions.hierarchy.SelectRootNode* method), 9
 onAttach() (*vi.actions.list.CloneAction* method), 12
 onAttach() (*vi.actions.list.DeleteAction* method), 12
 onAttach() (*vi.actions.list.EditAction* method), 12
 onAttach() (*vi.actions.list.ListPreviewAction* method), 13
 onAttach() (*vi.actions.list.ListPreviewInlineAction* method), 13
 onAttach() (*vi.actions.list.ListSelectFilterAction* method), 15
 onAttach() (*vi.actions.list.SelectAllAction* method), 16
 onAttach() (*vi.actions.list.SelectFieldsAction* method), 14
 onAttach() (*vi.actions.list.SelectInvertAction* method), 16
 onAttach() (*vi.actions.list.UnSelectAllAction* method), 16
 onAttach() (*vi.actions.list_order.ShopMarkAction* method), 17
 onAttach() (*vi.actions.tree.DeleteAction* method), 19
 onAttach() (*vi.actions.tree.EditAction* method), 18
 onAttach() (*vi.actions.tree.SelectRootNode* method), 19
 onAttach() (*vi.framework.components.datatable.SelectTable* method), 21
 onAttach() (*vi.serversideaction.ServerSideActionWdg* method), 70
 onAttach() (*vi.sidebarwidgets.filterselector.FilterSelector* method), 24
 onAttach() (*vi.widgets.code.Codemirror* method), 32
 onAttach() (*vi.widgets.edit.EditWidget* method), 34
 onAttach() (*vi.widgets.EditWidget* method), 54
 onAttach() (*vi.widgets.HtmlEditor* method), 58
 onAttach() (*vi.widgets.htmleditor.HtmlEditor* method), 38
 onAttach() (*vi.widgets.list.ListWidget* method), 40
 onAttach() (*vi.widgets.ListWidget* method), 54
 onAttach() (*vi.widgets.SideBar* method), 57
 onAttach() (*vi.widgets.sidebar.SideBar* method), 43
 onAttach() (*vi.widgets.table.SelectTable* method), 44
 onAttach() (*vi.widgets.tree.TreeWidget* method), 50
 onAttach() (*vi.widgets.TreeWidget* method), 60
 onBoneChange() (*vi.widgets.edit.EditWidget* method), 34
 onBoneChange() (*vi.widgets.EditWidget* method), 54
 onBtnCloseReleased() (*vi.pane.Pane* method), 68
 onChange() (*vi.actions.file.FileSelectUploader* method), 6
 onChange() (*vi.actions.hierarchy.SelectRootNode* method), 10
 onChange() (*vi.actions.list.ListPreviewAction* method), 13
 onChange() (*vi.actions.list.LoadNextBatchAction* method), 15
 onChange() (*vi.actions.list.SetPageRowAmountAction* method), 15
 onChange() (*vi.actions.tree.SelectRootNode* method), 19
 onChange() (*vi.framework.components.datatable.SelectTable* method), 21
 onChange() (*vi.widgets.edit.EditWidget* method), 34
 onChange() (*vi.widgets.EditWidget* method), 54

- onChange() (vi.widgets.InternalEdit method), 58
- onChange() (vi.widgets.internaedit.InternalEdit method), 39
- onChange() (vi.widgets.preview.Preview method), 41
- onChange() (vi.widgets.table.SelectTable method), 44
- onChange() (vi.widgets.task.TaskSelectWidget method), 47
- onChange() (vi.widgets.TaskSelectWidget method), 58
- onClick() (vi.actions.context.ContextAction method), 5
- onClick() (vi.actions.edit.CancelClose method), 6
- onClick() (vi.actions.edit.ExecuteSingleton method), 6
- onClick() (vi.actions.edit.Refresh method), 6
- onClick() (vi.actions.edit.SaveClose method), 6
- onClick() (vi.actions.edit.SaveContinue method), 5
- onClick() (vi.actions.edit.SaveSingleton method), 5
- onClick() (vi.actions.file.AddLeafAction method), 7
- onClick() (vi.actions.file.AddNodeAction method), 7
- onClick() (vi.actions.file.DownloadAction method), 7
- onClick() (vi.actions.file.EditAction method), 7
- onClick() (vi.actions.hierarchy.AddAction method), 8
- onClick() (vi.actions.hierarchy.CloneAction method), 9
- onClick() (vi.actions.hierarchy.DeleteAction method), 9
- onClick() (vi.actions.hierarchy.EditAction method), 8
- onClick() (vi.actions.hierarchy.ListViewAction method), 10
- onClick() (vi.actions.hierarchy.ReloadAction method), 9
- onClick() (vi.actions.list.AddAction method), 11
- onClick() (vi.actions.list.CloneAction method), 12
- onClick() (vi.actions.list.CloseAction method), 13
- onClick() (vi.actions.list.CreateRecurrentAction method), 16
- onClick() (vi.actions.list.DeleteAction method), 12
- onClick() (vi.actions.list.EditAction method), 12
- onClick() (vi.actions.list.ExportCsvAction method), 16
- onClick() (vi.actions.list.ListPreviewAction method), 13
- onClick() (vi.actions.list.ListPreviewInlineAction method), 13
- onClick() (vi.actions.list.ListSelectFilterAction method), 16
- onClick() (vi.actions.list.LoadAllAction method), 15
- onClick() (vi.actions.list.LoadNextBatchAction method), 15
- onClick() (vi.actions.list.PageFindAction method), 15
- onClick() (vi.actions.list.ReloadAction method), 14
- onClick() (vi.actions.list.SelectAction method), 13
- onClick() (vi.actions.list.SelectAllAction method), 16
- onClick() (vi.actions.list.SelectFieldsAction method), 14
- onClick() (vi.actions.list.SelectInvertAction method), 16
- onClick() (vi.actions.list.SetPageRowAmountAction method), 14
- onClick() (vi.actions.list.TableNextPage method), 14
- onClick() (vi.actions.list.TablePrevPage method), 14
- onClick() (vi.actions.list.UnSelectAllAction method), 16
- onClick() (vi.actions.list_order.ShopMarkAction method), 17
- onClick() (vi.actions.tree.AddLeafAction method), 18
- onClick() (vi.actions.tree.AddNodeAction method), 18
- onClick() (vi.actions.tree.DeleteAction method), 19
- onClick() (vi.actions.tree.EditAction method), 18
- onClick() (vi.actions.tree.ReloadAction method), 19
- onClick() (vi.admin.AdminScreen method), 62
- onClick() (vi.AdminScreen method), 74
- onClick() (vi.log.logA method), 64
- onClick() (vi.log.LogButton method), 65
- onClick() (vi.login.BaseLoginHandler method), 66
- onClick() (vi.pane.GroupPane method), 68
- onClick() (vi.pane.Pane method), 68
- onClick() (vi.serversideaction.ServerSideActionWdg method), 71
- onClick() (vi.sidebarwidgets.filtersselector.FilterSelector method), 24
- onClick() (vi.widgets.accordion.AccordionSegment method), 29
- onClick() (vi.widgets.file.FileImagePopup method), 35
- onClick() (vi.widgets.file.FilePreviewImage method), 35
- onClick() (vi.widgets.htmleditor.TextInsertImageAction method), 37
- onClick() (vi.widgets.list.ListWidget method), 40
- onClick() (vi.widgets.ListWidget method), 53
- onClick() (vi.widgets.ToolTip method), 55
- onClick() (vi.widgets.tooltip.ToolTip method), 47
- onClick() (vi.widgets.topbar.Logout method), 48
- onClick() (vi.widgets.topbar.Scripiter method), 48
- onClick() (vi.widgets.topbar.Tasks method), 48
- onClick() (vi.widgets.topbar.TopBarWidget method), 48
- onClick() (vi.widgets.topbar.UserState method), 48
- onClick() (vi.widgets.TopBarWidget method), 53
- onCompletion() (vi.widgets.list.ListWidget method), 40
- onCompletion() (vi.widgets.ListWidget method), 54
- onCompletion() (webworker_scripts.SimpleNetwork method), 77
- onCopyKey() (vi.sidebarwidgets.internalpreview.InternalPreview method), 25
- onCurrentUserAvailable() (vi.widgets.topbar.Scripiter method), 48
- onCurrentUserAvailable() (vi.widgets.topbar.Tasks method), 48
- onCurrentUserAvailable() (vi.widgets.topbar.UserState method), 48
- onCursorMoved() (vi.framework.components.datatable.DataTable method), 22
- onCursorMoved() (vi.widgets.DataTable method), 55

- `onCursorMoved()` (*vi.widgets.table.DataTable* method), 45
- `onDataChanged()` (*vi.widgets.list.ListWidget* method), 40
- `onDataChanged()` (*vi.widgets.ListWidget* method), 54
- `onDataChanged()` (*vi.widgets.tree.TreeWidget* method), 50
- `onDataChanged()` (*vi.widgets.TreeWidget* method), 60
- `onDbClick()` (*vi.framework.components.datatable.SelectTable* method), 21
- `onDbClick()` (*vi.widgets.table.SelectTable* method), 44
- `onDetach()` (*vi.actions.context.ContextAction* method), 5
- `onDetach()` (*vi.actions.file.DownloadAction* method), 7
- `onDetach()` (*vi.actions.file.EditAction* method), 7
- `onDetach()` (*vi.actions.hierarchy.CloneAction* method), 9
- `onDetach()` (*vi.actions.hierarchy.DeleteAction* method), 9
- `onDetach()` (*vi.actions.hierarchy.EditAction* method), 8
- `onDetach()` (*vi.actions.hierarchy.SelectRootNode* method), 9
- `onDetach()` (*vi.actions.list.CloneAction* method), 12
- `onDetach()` (*vi.actions.list.DeleteAction* method), 12
- `onDetach()` (*vi.actions.list.EditAction* method), 12
- `onDetach()` (*vi.actions.list.ListPreviewAction* method), 13
- `onDetach()` (*vi.actions.list.ListPreviewInlineAction* method), 13
- `onDetach()` (*vi.actions.list.SelectAllAction* method), 16
- `onDetach()` (*vi.actions.list.SelectFieldsAction* method), 14
- `onDetach()` (*vi.actions.list.SelectInvertAction* method), 16
- `onDetach()` (*vi.actions.list.UnSelectAllAction* method), 16
- `onDetach()` (*vi.actions.list_order.ShopMarkAction* method), 17
- `onDetach()` (*vi.actions.tree.DeleteAction* method), 19
- `onDetach()` (*vi.actions.tree.EditAction* method), 18
- `onDetach()` (*vi.actions.tree.SelectRootNode* method), 19
- `onDetach()` (*vi.pane.Pane* method), 68
- `onDetach()` (*vi.serversideaction.ServerSideActionWdg* method), 71
- `onDetach()` (*vi.sidebarwidgets.filterselector.FilterSelector* method), 24
- `onDetach()` (*vi.widgets.code.Codemirror* method), 32
- `onDetach()` (*vi.widgets.edit.EditWidget* method), 34
- `onDetach()` (*vi.widgets.EditWidget* method), 54
- `onDetach()` (*vi.widgets.HtmlEditor* method), 58
- `onDetach()` (*vi.widgets.htmleditor.HtmlEditor* method), 38
- `onDetach()` (*vi.widgets.list.ListWidget* method), 40
- `onDetach()` (*vi.widgets.ListWidget* method), 54
- `onDetach()` (*vi.widgets.SideBar* method), 57
- `onDetach()` (*vi.widgets.sidebar.SideBar* method), 43
- `onDetach()` (*vi.widgets.tree.TreeWidget* method), 50
- `onDetach()` (*vi.widgets.TreeWidget* method), 60
- `onDownloadBtnClick()` (*vi.widgets.file.FileImagePopup* method), 35
- `onDragOver()` (*vi.widgets.tree.TreeWidget* method), 50
- `onDragOver()` (*vi.widgets.TreeWidget* method), 60
- `onDrop()` (*vi.widgets.file.FileWidget* method), 36
- `onDrop()` (*vi.widgets.FileWidget* method), 61
- `onDrop()` (*vi.widgets.tree.TreeWidget* method), 50
- `onDrop()` (*vi.widgets.TreeWidget* method), 60
- `onEditorChange()` (*vi.widgets.HtmlEditor* method), 58
- `onEditorChange()` (*vi.widgets.htmleditor.HtmlEditor* method), 38
- `onError()` (*vi.admin.AdminScreen* method), 63
- `onError()` (*vi.AdminScreen* method), 74
- `onError()` (*webworker_scripts.SimpleNetwork* method), 77
- `onExportBtnClick()` (*vi.widgets.csvexport.ExportCsvStarter* method), 33
- `onExportBtnClick()` (*vi.widgets.ExportCsvStarter* method), 59
- `onFailed()` (*vi.widgets.file.MultiUploader* method), 36
- `onFailed()` (*vi.widgets.file.Uploader* method), 36
- `onFilterChanged()` (*vi.sidebarwidgets.filterselector.CompoundFilter* method), 24
- `onFocus()` (*vi.pane.GroupPane* method), 69
- `onGetAuthMethodsFailure()` (*vi.login.LoginScreen* method), 67
- `onGetAuthMethodsFailure()` (*vi.LoginScreen* method), 73
- `onGetAuthMethodsSuccess()` (*vi.login.LoginScreen* method), 67
- `onGetAuthMethodsSuccess()` (*vi.LoginScreen* method), 73
- `onHasSubItemsChanged()` (*vi.widgets.appnavigation.NavigationElement* method), 31
- `onKeyDown()` (*vi.framework.components.datatable.SelectTable* method), 21
- `onKeyDown()` (*vi.widgets.InternalEdit* method), 58
- `onKeyDown()` (*vi.widgets.internaedit.InternalEdit* method), 39
- `onKeyDown()` (*vi.widgets.Search* method), 57
- `onKeyDown()` (*vi.widgets.search.Search* method), 42
- `onKeyDown()` (*vi.widgets.table.SelectTable* method), 44
- `onKeyDown()` (*vi.widgets.tree.TreeWidget* method), 49
- `onKeyDown()` (*vi.widgets.TreeWidget* method), 59
- `onKeyPress()` (*vi.actions.list.PageFindAction* method), 15
- `onKeyPress()` (*vi.login.LoginInputField* method), 66
- `onKeyPress()` (*vi.login.UserPasswordLoginHandler* method), 66

<i>method</i>), 66	<i>(vi.actions.hierarchy.SelectRootNode method)</i> , 10
onKeyUp() (<i>vi.framework.components.datatable.SelectTable method</i>), 21	onRootNodesAvailable()
onKeyUp() (<i>vi.widgets.table.SelectTable method</i>), 44	<i>(vi.actions.tree.SelectRootNode method)</i> , 19
onKeyUp() (<i>vi.widgets.tree.TreeWidget method</i>), 49	onScroll() (<i>vi.actions.list.LoadNextBatchAction method</i>), 15
onKeyUp() (<i>vi.widgets.TreeWidget method</i>), 59	onScroll() (<i>vi.widgets.DataTable method</i>), 56
onLoad() (<i>vi.widgets.file.MultiUploader method</i>), 36	onScroll() (<i>vi.widgets.table.DataTable method</i>), 46
onLoad() (<i>vi.widgets.file.Uploader method</i>), 35	onSelectionActivated() (<i>vi.actions.file.EditAction method</i>), 7
onLoginClick() (<i>vi.login.GoogleAccountLoginHandler method</i>), 67	onSelectionActivated()
onLoginClick() (<i>vi.login.UserPasswordLoginHandler method</i>), 66	<i>(vi.actions.hierarchy.EditAction method)</i> , 8
onLogoutSuccess() (<i>vi.login.LoginScreen method</i>), 67	onSelectionActivated() (<i>vi.actions.list.EditAction method</i>), 12
onLogoutSuccess() (<i>vi.LoginScreen method</i>), 73	onSelectionActivated() (<i>vi.actions.tree.EditAction method</i>), 18
onMkDir() (<i>vi.actions.file.AddNodeAction method</i>), 7	onSelectionActivated()
onMouseDown() (<i>vi.framework.components.datatable.SelectTable method</i>), 21	<i>(vi.framework.components.datatable.DataTable method)</i> , 23
onMouseDown() (<i>vi.widgets.table.SelectTable method</i>), 44	onSelectionActivated() (<i>vi.widgets.DataTable method</i>), 56
onMouseOut() (<i>vi.framework.components.datatable.SelectTable method</i>), 21	onSelectionActivated()
onMouseOut() (<i>vi.widgets.table.SelectTable method</i>), 44	<i>(vi.widgets.htmleditor.TextInsertImageAction method)</i> , 37
onMouseUp() (<i>vi.framework.components.datatable.SelectTable method</i>), 21	onSelectionActivated() (<i>vi.widgets.list.ListWidget method</i>), 40
onMouseUp() (<i>vi.widgets.table.SelectTable method</i>), 44	onSelectionActivated() (<i>vi.widgets.ListWidget method</i>), 54
onNextBatchNeeded() (<i>vi.widgets.list.ListWidget method</i>), 40	onSelectionActivated() (<i>vi.widgets.table.DataTable method</i>), 46
onNextBatchNeeded() (<i>vi.widgets.ListWidget method</i>), 53	onSelectionChanged()
onPathRequestSucceeded()	<i>(vi.actions.context.ContextAction method)</i> , 5
<i>(vi.widgets.tree.TreeBrowserWidget method)</i> , 51	onSelectionChanged()
onPathRequestSucceeded()	<i>(vi.actions.file.DownloadAction method)</i> , 7
<i>(vi.widgets.TreeBrowserWidget method)</i> , 60	onSelectionChanged() (<i>vi.actions.file.EditAction method</i>), 7
onProgress() (<i>vi.widgets.file.Uploader method</i>), 35	onSelectionChanged()
onReadyStateChange() (<i>web-worker_scripts.HTTPRequest method</i>), 76	<i>(vi.actions.hierarchy.CloneAction method)</i> , 9
onRequestingFinished() (<i>vi.widgets.list.ListWidget method</i>), 39	onSelectionChanged() (<i>vi.actions.hierarchy.DeleteAction method</i>), 9
onRequestingFinished()	onSelectionChanged()
<i>(vi.widgets.list.ViewportListWidget method)</i> , 41	<i>(vi.actions.hierarchy.EditAction method)</i> , 8
onRequestingFinished() (<i>vi.widgets.ListWidget method</i>), 53	onSelectionChanged() (<i>vi.actions.list.CloneAction method</i>), 12
onRequestSucceeded() (<i>vi.widgets.tree.TreeWidget method</i>), 50	onSelectionChanged() (<i>vi.actions.list.DeleteAction method</i>), 12
onRequestSucceeded() (<i>vi.widgets.TreeWidget method</i>), 60	onSelectionChanged() (<i>vi.actions.list.EditAction method</i>), 12
onRootNodeChanged()	
<i>(vi.actions.hierarchy.SelectRootNode method)</i> , 10	
onRootNodeChanged() (<i>vi.actions.tree.SelectRootNode method</i>), 19	
onRootNodesAvailable()	

onSelectionChanged()
 (vi.actions.list.ListPreviewAction method), 13
 onSelectionChanged()
 (vi.actions.list.ListPreviewInlineAction method), 13
 onSelectionChanged()
 (vi.actions.list_order.ShopMarkAction method), 17
 onSelectionChanged() (vi.actions.tree.DeleteAction method), 19
 onSelectionChanged() (vi.actions.tree.EditAction method), 18
 onSelectionChanged()
 (vi.framework.components.datatable.DataTable method), 23
 onSelectionChanged()
 (vi.serversideaction.ServerSideActionWdg method), 71
 onSelectionChanged() (vi.widgets.DataTable method), 56
 onSelectionChanged()
 (vi.widgets.hierarchy.HierarchyWidget method), 37
 onSelectionChanged() (vi.widgets.HierarchyWidget method), 61
 onSelectionChanged() (vi.widgets.table.DataTable method), 46
 onSendClick() (vi.login.UserPasswordLoginHandler method), 66
 onSetDefaultRootNode() (vi.widgets.tree.TreeWidget method), 50
 onSetDefaultRootNode() (vi.widgets.TreeWidget method), 60
 onSkeyAvailable() (vi.widgets.file.Uploader method), 35
 onStartSearch() (vi.sidebarwidgets.filterselector.FilterSelector method), 24
 onStartSearch() (vi.widgets.file.FileWidget method), 36
 onStartSearch() (vi.widgets.FileWidget method), 61
 onSuccess() (vi.widgets.file.MultiUploader method), 36
 onSuccess() (vi.widgets.file.Uploader method), 35
 onTableChanged() (vi.actions.list.SelectAllAction method), 16
 onTableChanged() (vi.actions.list.SelectFieldsAction method), 14
 onTableChanged() (vi.actions.list.SelectInvertAction method), 16
 onTableChanged() (vi.actions.list.TableItems method), 14
 onTableChanged() (vi.actions.list.UnSelectAllAction method), 16
 onTableChanged() (vi.framework.components.datatable.DataTable method), 23
 onTableChanged() (vi.widgets.DataTable method), 56
 onTableChanged() (vi.widgets.table.DataTable method), 46
 onTaskListAvailable() (vi.widgets.topbar.Tasks method), 48
 onTaskListFailure() (vi.widgets.topbar.Tasks method), 48
 onUploadAdded() (vi.widgets.file.MultiUploader method), 36
 onUploadAdded() (vi.widgets.file.Uploader method), 35
 onUploadUrlAvailable() (vi.widgets.file.MultiUploader method), 36
 onUploadUrlAvailable() (vi.widgets.file.Uploader method), 35
 onUserTestFail() (vi.widgets.UserLogoutMsg method), 57
 onUserTestFail() (vi.widgets.userlogoutmsg.UserLogoutMsg method), 52
 onUserTestSuccess() (vi.widgets.UserLogoutMsg method), 57
 onUserTestSuccess() (vi.widgets.userlogoutmsg.UserLogoutMsg method), 52
 onVerifyClick() (vi.login.UserPasswordLoginHandler method), 66
 onViewfocusedChanged() (vi.views.list.listHandlerWidget method), 26
 onViewfocusedChanged() (vi.views.singleton.singletonHandlerWidget method), 28
 openEdit() (vi.widgets.topbar.UserState method), 48
 openEditor() (vi.actions.hierarchy.CloneAction method), 9
 openEditor() (vi.actions.hierarchy.EditAction method), 8
 openEditor() (vi.actions.list.CloneAction method), 12
 openEditor() (vi.actions.list.EditAction method), 12
 openEditor() (vi.log.logA method), 64
 openHelp() (vi.widgets.code.Scripter method), 32
 openLog() (vi.log.LogButton method), 65
 openModule() (vi.actions.context.ContextAction method), 5
 openNewMainView() (vi.admin.AdminScreen method), 62
 openNewMainView() (vi.AdminScreen method), 74
 openNewPopup() (vi.admin.AdminScreen method), 62
 openNewPopup() (vi.AdminScreen method), 74
 openView() (vi.admin.AdminScreen method), 62
 openView() (vi.AdminScreen method), 74
 Overview (class in vi.views.overview), 28
 OverviewWidget (class in vi.views.overview), 28

P

PageFindAction (class in *vi.actions.list*), 15
 Pane (class in *vi.pane*), 67
 parseAnswer() (*vi.login.BaseLoginHandler* method), 66
 ParsedErrorItem (class in *vi.widgets.internaedit*), 38
 parseHashParameters() (in module *vi.widgets.edit*), 33
 PassiveErrorItem (class in *vi.widgets.internaedit*), 38
 performLogics() (*vi.widgets.InternalEdit* method), 58
 performLogics() (*vi.widgets.internaedit.InternalEdit* method), 39
 performReload() (*vi.actions.edit.Refresh* method), 6
 pollInterval (*vi.widgets.UserLogoutMsg* attribute), 57
 pollInterval (*vi.widgets.userlogoutmsg.UserLogoutMsg* attribute), 51
 postInit() (*vi.actions.list.TableItems* method), 14
 postInit() (*vi.actions.list.TableNextPage* method), 14
 postInit() (*vi.actions.list.TablePrevPage* method), 14
 prepareCol() (*vi.framework.components.datatable.SelectTable* method), 22
 prepareCol() (*vi.widgets.table.SelectTable* method), 45
 Preview (class in *vi.widgets.preview*), 41
 PythonCode (class in *vi.widgets.code*), 32

Q

queue (*vi.utils.indexedddb* attribute), 72

R

rebuildCB() (*vi.actions.list.ListPreviewAction* method), 13
 rebuildChildrenClassInfo() (*vi.pane.Pane* method), 68
 rebuildPath() (*vi.widgets.tree.TreeBrowserWidget* method), 51
 rebuildPath() (*vi.widgets.TreeBrowserWidget* method), 60
 rebuildTable() (*vi.framework.components.datatable.DataTable* method), 22
 rebuildTable() (*vi.framework.components.datatable.ViewportDataTable* method), 23
 rebuildTable() (*vi.widgets.DataTable* method), 56
 rebuildTable() (*vi.widgets.table.DataTable* method), 45
 recalcHeight() (*vi.widgets.DataTable* method), 55
 recalcHeight() (*vi.widgets.table.DataTable* method), 45
 receivedStructure() (*vi.widgets.list.ListWidget* method), 40
 receivedStructure() (*vi.widgets.ListWidget* method), 54
 receivedStructure() (*vi.widgets.tree.TreeWidget* method), 49
 receivedStructure() (*vi.widgets.TreeWidget* method), 59
 redirectNoAdmin() (*vi.login.LoginScreen* method), 67
 redirectNoAdmin() (*vi.LoginScreen* method), 73
 reevaluate() (*vi.sidebarwidgets.filterselector.CompoundFilter* method), 24
 reevaluate() (*vi.widgets.Search* method), 57
 reevaluate() (*vi.widgets.search.Search* method), 43
 Refresh (class in *vi.actions.edit*), 6
 registerScroll() (*vi.actions.list.LoadNextBatchAction* method), 15
 ReloadAction (class in *vi.actions.hierarchy*), 9
 ReloadAction (class in *vi.actions.list*), 14
 ReloadAction (class in *vi.actions.tree*), 19
 reloadData() (*vi.widgets.edit.EditWidget* method), 34
 reloadData() (*vi.widgets.EditWidget* method), 55
 reloadData() (*vi.widgets.hierarchy.HierarchyWidget* method), 37
 reloadData() (*vi.widgets.HierarchyWidget* method), 61
 reloadData() (*vi.widgets.list.ListWidget* method), 40
 reloadData() (*vi.widgets.ListWidget* method), 54
 reloadData() (*vi.widgets.repeatdate.RepeatDatePopup* method), 42
 reloadData() (*vi.widgets.tree.TreeBrowserWidget* method), 51
 reloadData() (*vi.widgets.tree.TreeWidget* method), 50
 reloadData() (*vi.widgets.TreeBrowserWidget* method), 60
 reloadData() (*vi.widgets.TreeWidget* method), 60
 reloadListWidget() (*vi.widgets.hierarchy.HierarchyWidget* method), 37
 reloadListWidget() (*vi.widgets.HierarchyWidget* method), 61
 remove() (*vi.framework.components.datatable.DataTable* method), 22
 remove() (*vi.screen.Screen* method), 70
 remove() (*vi.widgets.DataTable* method), 56
 remove() (*vi.widgets.table.DataTable* method), 45
 removeAction() (*vi.widgets.appnavigation.NavigationElement* method), 31
 removeChildPane() (*vi.pane.Pane* method), 68
 removeInfo() (*vi.log.LogButton* method), 65
 removeNavigationPoint() (*vi.widgets.appnavigation.AppNavigation* method), 31
 removeNewCls() (*vi.log.Log* method), 65
 removeRow() (*vi.framework.components.datatable.SelectTable* method), 21
 removeRow() (*vi.widgets.table.SelectTable* method), 44
 removeSelectedRow() (*vi.framework.components.datatable.SelectTable* method), 21
 removeSelectedRow() (*vi.widgets.table.SelectTable* method), 44

removeWidget() (*vi.admin.AdminScreen* method), 63
 removeWidget() (*vi.AdminScreen* method), 74
 removeWidget() (*vi.pane.Pane* method), 68
 renderPopOut() (*vi.log.LogButton* method), 65
 renderStructure() (*vi.widgets.InternalEdit* method), 58
 renderStructure() (*vi.widgets.internaedit.InternalEdit* method), 39
 RepeatDatePopup (*class in vi.widgets.repeatdate*), 42
 replaceWithMessage() (*vi.widgets.csvexport.ExportCsv* method), 33
 replaceWithMessage() (*vi.widgets.ExportCsv* method), 59
 replaceWithMessage() (*vi.widgets.file.MultiUploader* method), 36
 replaceWithMessage() (*vi.widgets.file.Uploader* method), 36
 request() (*in module webworker_scripts*), 76
 request() (*webworker_scripts.SimpleNetwork* method), 76
 requestChildren() (*vi.widgets.tree.TreeWidget* method), 49
 requestChildren() (*vi.widgets.TreeWidget* method), 59
 requestData() (*webworker_scripts.requestList* method), 76
 requestList (*class in webworker_scripts*), 76
 requestStructure() (*vi.widgets.list.ListWidget* method), 40
 requestStructure() (*vi.widgets.ListWidget* method), 54
 requestStructure() (*vi.widgets.tree.TreeWidget* method), 49
 requestStructure() (*vi.widgets.TreeWidget* method), 59
 reset() (*vi.admin.AdminScreen* method), 62
 reset() (*vi.AdminScreen* method), 74
 reset() (*vi.log.Log* method), 65
 reset() (*vi.log.LogButton* method), 65
 reset() (*vi.login.BaseLoginHandler* method), 66
 reset() (*vi.login.UserPasswordLoginHandler* method), 66
 reset() (*vi.priorityqueue.StartupQueue* method), 69
 resetLoadingState() (*vi.actions.edit.CancelClose* method), 6
 resetLoadingState() (*vi.actions.edit.ExecuteSingleton* method), 6
 resetLoadingState() (*vi.actions.edit.Refresh* method), 6
 resetLoadingState() (*vi.actions.edit.SaveClose* method), 6
 resetLoadingState() (*vi.actions.edit.SaveContinue* method), 5
 resetLoadingState() (*vi.actions.edit.SaveSingleton* method), 5
 resetLoadingState() (*vi.actions.file.AddLeafAction* method), 7
 resetLoadingState() (*vi.actions.file.AddNodeAction* method), 7
 resetLoadingState() (*vi.actions.file.DownloadAction* method), 8
 resetLoadingState() (*vi.actions.file.EditAction* method), 7
 resetLoadingState() (*vi.actions.hierarchy.AddAction* method), 8
 resetLoadingState() (*vi.actions.hierarchy.CloneAction* method), 9
 resetLoadingState() (*vi.actions.hierarchy.DeleteAction* method), 9
 resetLoadingState() (*vi.actions.hierarchy.EditAction* method), 8
 resetLoadingState() (*vi.actions.hierarchy.ListViewAction* method), 10
 resetLoadingState() (*vi.actions.hierarchy.ReloadAction* method), 9
 resetLoadingState() (*vi.actions.list.AddAction* method), 11
 resetLoadingState() (*vi.actions.list.CloneAction* method), 12
 resetLoadingState() (*vi.actions.list.DeleteAction* method), 13
 resetLoadingState() (*vi.actions.list.EditAction* method), 12
 resetLoadingState() (*vi.actions.list.LoadAllAction* method), 15
 resetLoadingState() (*vi.actions.list.LoadNextBatchAction* method), 15
 resetLoadingState() (*vi.actions.list.PageFindAction* method), 15
 resetLoadingState() (*vi.actions.list.ReloadAction* method), 14
 resetLoadingState() (*vi.actions.list.SetPageRowAmountAction* method), 15
 resetLoadingState() (*vi.actions.list.TableNextPage* method), 14
 resetLoadingState() (*vi.actions.tree.AddLeafAction* method), 18
 resetLoadingState() (*vi.actions.tree.AddNodeAction* method), 18
 resetLoadingState() (*vi.actions.tree.DeleteAction* method), 19
 resetLoadingState() (*vi.actions.tree.EditAction* method), 19

- method), 18
- resetLoadingState() (vi.actions.tree.ReloadAction method), 19
- resetLoadingState() (vi.framework.components.actionbar.ActionBar method), 20
- resetLoadingState() (vi.serversideaction.ServerSideActionWdg method), 71
- resetLoadingState() (vi.widgets.htmleditor.TextInsertImageAction method), 37
- resetLoadingState() (vi.widgets.Search method), 57
- resetLoadingState() (vi.widgets.search.Search method), 42
- resetSearch() (vi.widgets.Search method), 57
- resetSearch() (vi.widgets.search.Search method), 42
- run() (vi.priorityqueue.StartupQueue method), 70
- run() (vi.widgets.code.PythonCode method), 32
- runClick() (vi.widgets.code.Scripiter method), 32
- running() (webworker_scripts.requestList method), 76
- ## S
- s (in module vi), 75
- save() (vi.widgets.repeatdate.RepeatDatePopup method), 42
- SaveClose (class in vi.actions.edit), 6
- SaveContinue (class in vi.actions.edit), 5
- SaveSingleton (class in vi.actions.edit), 5
- sc (in module vi), 75
- scinv (in module vi), 75
- Screen (class in vi.screen), 70
- Scripiter (class in vi.widgets.code), 32
- Scripiter (class in vi.widgets.topbar), 48
- Search (class in vi.widgets), 56
- Search (class in vi.widgets.search), 42
- searchWidget() (vi.widgets.file.FileWidget method), 36
- searchWidget() (vi.widgets.FileWidget method), 61
- SelectAction (class in vi.actions.list), 13
- selectAll() (vi.framework.components.datatable.SelectTable method), 22
- selectAll() (vi.widgets.table.SelectTable method), 45
- SelectAllAction (class in vi.actions.list), 16
- SelectFieldsAction (class in vi.actions.list), 14
- SelectFieldsPopup (class in vi.actions.list), 13
- selectHandler() (vi.login.LoginScreen method), 67
- selectHandler() (vi.LoginScreen method), 73
- SelectInvertAction (class in vi.actions.list), 16
- selectorReturn() (vi.widgets.list.ListWidget method), 39
- selectorReturn() (vi.widgets.ListWidget method), 53
- selectorReturn() (vi.widgets.tree.TreeWidget method), 49
- selectorReturn() (vi.widgets.TreeWidget method), 59
- SelectRootNode (class in vi.actions.hierarchy), 9
- SelectRootNode (class in vi.actions.tree), 19
- selectRow() (vi.framework.components.datatable.SelectTable method), 21
- selectRow() (vi.widgets.table.SelectTable method), 44
- SelectTable (class in vi.framework.components.datatable), 20
- SelectTable (class in vi.widgets.table), 43
- seperatorAction() (vi.widgets.appnavigation.Navigationblock method), 31
- serializeForDocument() (vi.widgets.InternalEdit method), 58
- serializeForDocument() (vi.widgets.internaedit.InternalEdit method), 39
- serializeForPost() (vi.widgets.InternalEdit method), 58
- serializeForPost() (vi.widgets.internaedit.InternalEdit method), 39
- ServerSideActionWdg (class in vi.serversideaction), 70
- ServerTaskWidget (class in vi.widgets.task), 46
- setActions() (vi.framework.components.actionbar.ActionBar method), 20
- setActiveTask() (vi.widgets.task.TaskSelectWidget method), 47
- setActiveTask() (vi.widgets.TaskSelectWidget method), 58
- setAmount() (vi.widgets.list.ListWidget method), 39
- setAmount() (vi.widgets.list.ViewportListWidget method), 41
- setAmount() (vi.widgets.ListWidget method), 53
- setCell() (vi.framework.components.datatable.SelectTable method), 22
- setCell() (vi.widgets.table.SelectTable method), 45
- setCellRender() (vi.framework.components.datatable.DataTable method), 23
- setCellRender() (vi.widgets.DataTable method), 56
- setCellRender() (vi.widgets.table.DataTable method), 46
- setCellRenderers() (vi.framework.components.datatable.DataTable method), 23
- setCellRenderers() (vi.widgets.DataTable method), 56
- setCellRenderers() (vi.widgets.table.DataTable method), 46
- setContext() (vi.widgets.list.ListWidget method), 40
- setContext() (vi.widgets.ListWidget method), 54
- setCurrentModulDescr() (vi.widgets.topbar.TopBarWidget method), 48
- setCurrentModulDescr() (vi.widgets.TopBarWidget method), 53
- setCursor() (vi.widgets.code.Codemirror method), 32
- setCursorPosition() (vi.framework.components.datatable.SelectTable method), 21

- [setCursorRow\(\)](#) (*vi.widgets.table.SelectTable* method), [44](#)
[setData\(\)](#) (*vi.widgets.edit.EditWidget* method), [34](#)
[setData\(\)](#) (*vi.widgets.EditWidget* method), [55](#)
[setData\(\)](#) (*vi.widgets.repeatdate.RepeatDatePopup* method), [42](#)
[setDataProvider\(\)](#) (*vi.framework.components.datatable.DataTable* method), [22](#)
[setDataProvider\(\)](#) (*vi.widgets.DataTable* method), [55](#)
[setDataProvider\(\)](#) (*vi.widgets.table.DataTable* method), [45](#)
[setFields\(\)](#) (*vi.widgets.list.ListWidget* method), [40](#)
[setFields\(\)](#) (*vi.widgets.ListWidget* method), [54](#)
[setFile\(\)](#) (*vi.widgets.file.FilePreviewImage* method), [35](#)
[setFilter\(\)](#) (*vi.widgets.list.ListWidget* method), [40](#)
[setFilter\(\)](#) (*vi.widgets.ListWidget* method), [54](#)
[setFinalElem\(\)](#) (*vi.priorityqueue.StartupQueue* method), [69](#)
[setHeader\(\)](#) (*vi.framework.components.datatable.SelectTable* method), [21](#)
[setHeader\(\)](#) (*vi.widgets.table.SelectTable* method), [44](#)
[setImage\(\)](#) (*vi.pane.Pane* method), [68](#)
[setListView\(\)](#) (*vi.widgets.hierarchy.HierarchyWidget* method), [37](#)
[setListView\(\)](#) (*vi.widgets.HierarchyWidget* method), [61](#)
[setPage\(\)](#) (*vi.widgets.list.ListWidget* method), [39](#)
[setPage\(\)](#) (*vi.widgets.list.ViewportListWidget* method), [41](#)
[setPage\(\)](#) (*vi.widgets.ListWidget* method), [53](#)
[setPageAmount\(\)](#) (*vi.actions.list.SetPageRowAmountAction* method), [15](#)
[SetPageRowAmountAction](#) (class in *vi.actions.list*), [14](#)
[setPath\(\)](#) (*vi.Application* method), [75](#)
[setPayed\(\)](#) (*vi.actions.list_order.ShopMarkAction* method), [17](#)
[setPayedFailed\(\)](#) (*vi.actions.list_order.ShopMarkAction* method), [17](#)
[setPayedSucceeded\(\)](#) (*vi.actions.list_order.ShopMarkAction* method), [17](#)
[setPreventUnloading\(\)](#) (in module *vi.utils*), [71](#)
[setRootNode\(\)](#) (*vi.widgets.tree.TreeWidget* method), [50](#)
[setRootNode\(\)](#) (*vi.widgets.TreeWidget* method), [60](#)
[setSelector\(\)](#) (*vi.widgets.list.ListWidget* method), [39](#)
[setSelector\(\)](#) (*vi.widgets.ListWidget* method), [53](#)
[setSelector\(\)](#) (*vi.widgets.tree.TreeWidget* method), [49](#)
[setSelector\(\)](#) (*vi.widgets.TreeWidget* method), [59](#)
[setShownFields\(\)](#) (*vi.framework.components.datatable.DataTable* method), [23](#)
[setShownFields\(\)](#) (*vi.widgets.DataTable* method), [56](#)
[setShownFields\(\)](#) (*vi.widgets.table.DataTable* method), [46](#)
[setStyle\(\)](#) (*vi.widgets.file.FileLeafWidget* method), [36](#)
[setStyle\(\)](#) (*vi.widgets.file.FileNodeWidget* method), [36](#)
[setStyle\(\)](#) (*vi.widgets.tree.BreadcrumbNodeWidget* method), [50](#)
[setStyle\(\)](#) (*vi.widgets.tree.BrowserLeafWidget* method), [50](#)
[setStyle\(\)](#) (*vi.widgets.tree.BrowserNodeWidget* method), [50](#)
[setTableActionBar\(\)](#) (*vi.widgets.list.ListWidget* method), [40](#)
[setTableActionBar\(\)](#) (*vi.widgets.list.ViewportListWidget* method), [41](#)
[setTableActionBar\(\)](#) (*vi.widgets.ListWidget* method), [53](#)
[setText\(\)](#) (*vi.pane.Pane* method), [68](#)
[setTitle\(\)](#) (*vi.Application* method), [75](#)
[setTitle\(\)](#) (*vi.screen.Screen* method), [70](#)
[setTitle\(\)](#) (*vi.widgets.topbar.TopBarWidget* method), [48](#)
[setTitle\(\)](#) (*vi.widgets.TopBarWidget* method), [53](#)
[setUrl\(\)](#) (*vi.widgets.preview.Preview* method), [41](#)
[setView\(\)](#) (*vi.sidebarwidgets.filtersselector.FilterSelector* method), [24](#)
[setWidget\(\)](#) (*vi.widgets.SideBar* method), [57](#)
[setWidget\(\)](#) (*vi.widgets.sidebar.SideBar* method), [43](#)
[ShopMarkAction](#) (class in *vi.actions.list_order*), [17](#)
[ShopMarkCanceledAction](#) (class in *vi.actions.list_order*), [17](#)
[ShopMarkPayedAction](#) (class in *vi.actions.list_order*), [17](#)
[ShopMarkSentAction](#) (class in *vi.actions.list_order*), [17](#)
[showErrorMsg\(\)](#) (*vi.widgets.edit.EditWidget* method), [34](#)
[showErrorMsg\(\)](#) (*vi.widgets.EditWidget* method), [55](#)
[showErrorMsg\(\)](#) (*vi.widgets.list.ListWidget* method), [40](#)
[showErrorMsg\(\)](#) (*vi.widgets.ListWidget* method), [53](#)
[showErrorMsg\(\)](#) (*vi.widgets.repeatdate.RepeatDatePopup* method), [42](#)
[showErrorMsg\(\)](#) (*vi.widgets.tree.TreeWidget* method), [49](#)
[showErrorMsg\(\)](#) (*vi.widgets.TreeWidget* method), [59](#)
[showListView\(\)](#) (*vi.widgets.hierarchy.HierarchyWidget* method), [37](#)
[showListView\(\)](#) (*vi.widgets.HierarchyWidget* method), [61](#)
[showLoginWindow\(\)](#) (*vi.widgets.UserLogoutMsg* method), [57](#)
[showLoginWindow\(\)](#) (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), [51](#)
[showMessage\(\)](#) (*vi.widgets.UserLogoutMsg* method), [57](#)
[showMessage\(\)](#) (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), [51](#)
[SideBar](#) (class in *vi.widgets*), [57](#)
[SideBar](#) (class in *vi.widgets.sidebar*), [43](#)

- SimpleNetwork (class in webworker_scripts), 76
 singletonHandler (class in vi.views.singleton), 28
 singletonHandlerWidget (class in vi.views.singleton), 28
 stackWidget() (vi.admin.AdminScreen method), 63
 stackWidget() (vi.AdminScreen method), 74
 start() (in module vi), 75
 startFind() (vi.actions.list.PageFindAction method), 15
 startFullscreen() (vi.widgets.code.Scripter method), 32
 startPolling() (vi.widgets.UserLogoutMsg method), 57
 startPolling() (vi.widgets.userlogoutmsg.UserLogoutMsg method), 52
 startup() (vi.admin.AdminScreen method), 62
 startup() (vi.AdminScreen method), 74
 startup() (vi.Application method), 75
 startupFailure() (vi.Application method), 75
 StartupQueue (class in vi.priorityqueue), 69
 startupQueue (in module vi.priorityqueue), 70
 stop() (vi.widgets.code.PythonCode method), 32
 stopInterval() (vi.widgets.UserLogoutMsg method), 57
 stopInterval() (vi.widgets.userlogoutmsg.UserLogoutMsg method), 51
 style (vi.widgets.internaedit.ParsedErrorItem attribute), 38
 style (vi.widgets.internaedit.PassiveErrorItem attribute), 38
 switchDisabledState() (vi.serversideaction.ServerSideActionWdg method), 70
 switchFullscreen() (vi.admin.AdminScreen method), 63
 switchFullscreen() (vi.AdminScreen method), 74
- ## T
- tableInitialization() (vi.widgets.list.ListWidget method), 39
 tableInitialization() (vi.widgets.list.ViewportListWidget method), 41
 tableInitialization() (vi.widgets.ListWidget method), 53
 TableItems (class in vi.actions.list), 14
 TableNextPage (class in vi.actions.list), 14
 TablePrevPage (class in vi.actions.list), 14
 Tasks (class in vi.widgets.topbar), 48
 TaskSelectWidget (class in vi.widgets), 58
 TaskSelectWidget (class in vi.widgets.task), 46
 TaskWidget (class in vi.widgets), 57
 TaskWidget (class in vi.widgets.task), 46
 testIfNextBatchNeededImmediately() (vi.widgets.DataTable method), 56
 testIfNextBatchNeededImmediately() (vi.widgets.table.DataTable method), 45
 TextInsertImageAction (class in vi.widgets.htmleditor), 37
 toggle() (vi.widgets.accordion.AccordionSegment method), 29
 toggleIntPrev() (vi.actions.list.ListPreviewInlineAction method), 13
 toggleListView() (vi.widgets.hierarchy.HierarchyWidget method), 37
 toggleListView() (vi.widgets.HierarchyWidget method), 61
 toggleMsgCenter() (vi.log.Log method), 65
 Tooltip (class in vi.widgets), 55
 Tooltip (class in vi.widgets.tooltip), 47
 TopBarWidget (class in vi.widgets), 53
 TopBarWidget (class in vi.widgets.topbar), 47
 toplevelActionSelector (in module vi.priorityqueue), 69
 tpl (vi.widgets.appnavigation.NavigationElement attribute), 30
 TreeBrowserWidget (class in vi.widgets), 60
 TreeBrowserWidget (class in vi.widgets.tree), 50
 treeHandler (class in vi.views.tree), 29
 treeHandlerWidget (class in vi.views.tree), 29
 TreeWidget (class in vi.widgets), 59
 TreeWidget (class in vi.widgets.tree), 49
- ## U
- unlock() (vi.login.BaseLoginHandler method), 66
 unlock() (vi.pane.Pane method), 68
 unlock() (vi.screen.Screen method), 70
 unselectAll() (vi.framework.components.datatable.SelectTable method), 22
 unselectAll() (vi.widgets.table.SelectTable method), 45
 UnselectAllAction (class in vi.actions.list), 16
 unserialize() (vi.widgets.InternalEdit method), 58
 unserialize() (vi.widgets.internaedit.InternalEdit method), 39
 update() (vi.actions.hierarchy.SelectRootNode method), 9
 update() (vi.actions.tree.SelectRootNode method), 19
 update() (vi.framework.components.datatable.DataTable method), 22
 update() (vi.framework.components.datatable.ViewportDataTable method), 23
 update() (vi.widgets.topbar.Tasks method), 48
 update() (vi.widgets.topbar.UserState method), 48
 updateConf() (in module vi.config), 63
 updateEmptyNotification() (vi.widgets.list.ListWidget method), 40
 updateEmptyNotification() (vi.widgets.ListWidget method), 54

updateUser() (*vi.widgets.topbar.Scripiter method*), 48
 Uploader (*class in vi.widgets.file*), 35
 UserLogoutMsg (*class in vi.widgets*), 57
 UserLogoutMsg (*class in vi.widgets.userlogoutmsg*), 51
 UserPasswordLoginHandler (*class in vi.login*), 66
 UserState (*class in vi.widgets.topbar*), 48

V

vi
 module, 4
 vi.actions
 module, 4
 vi.actions.context
 module, 4
 vi.actions.edit
 module, 5
 vi.actions.file
 module, 6
 vi.actions.hierarchy
 module, 8
 vi.actions.list
 module, 10
 vi.actions.list_order
 module, 17
 vi.actions.tree
 module, 18
 vi.admin
 module, 62
 vi.config
 module, 63
 vi.exception
 module, 64
 vi.framework
 module, 19
 vi.framework.components
 module, 19
 vi.framework.components.actionbar
 module, 20
 vi.framework.components.datatable
 module, 20
 vi.log
 module, 64
 vi.login
 module, 65
 vi.pane
 module, 67
 vi.priorityqueue
 module, 69
 vi.screen
 module, 70
 vi.serversideaction
 module, 70
 vi.sidebarwidgets
 module, 24
 vi.sidebarwidgets.filterselector
 module, 24
 vi.sidebarwidgets.internalpreview
 module, 24
 vi.translations
 module, 25
 vi.translations.de
 module, 25
 vi.translations.en
 module, 25
 vi.utils
 module, 71
 vi.views
 module, 25
 vi.views.edit
 module, 25
 vi.views.hierarchy
 module, 26
 vi.views.list
 module, 26
 vi.views.log
 module, 27
 vi.views.notfound
 module, 27
 vi.views.overview
 module, 27
 vi.views.singleton
 module, 28
 vi.views.tree
 module, 28
 vi.widgets
 module, 29
 vi.widgets.accordion
 module, 29
 vi.widgets.appnavigation
 module, 30
 vi.widgets.code
 module, 31
 vi.widgets.csvexport
 module, 33
 vi.widgets.edit
 module, 33
 vi.widgets.file
 module, 35
 vi.widgets.hierarchy
 module, 37
 vi.widgets.htmleditor
 module, 37
 vi.widgets.internaedit
 module, 38
 vi.widgets.list
 module, 39
 vi.widgets.preview
 module, 41

`vi.widgets.repeatdate`
 module, [42](#)
`vi.widgets.search`
 module, [42](#)
`vi.widgets.sidebar`
 module, [43](#)
`vi.widgets.table`
 module, [43](#)
`vi.widgets.task`
 module, [46](#)
`vi.widgets.tooltip`
 module, [47](#)
`vi.widgets.topbar`
 module, [47](#)
`vi.widgets.tree`
 module, [49](#)
`vi.widgets.userlogoutmsg`
 module, [51](#)
`vi_conf` (in module `vi`), [73](#)
`vi_conf` (in module `vi.config`), [63](#)
`ViewportDataTable` (class in `vi.framework.components.datatable`), [23](#)
`ViewportListWidget` (class in `vi.widgets.list`), [40](#)
`viInitializedEvent` (in module `vi.admin`), [63](#)
`visibilityChanged()` (`vi.widgets.UserLogoutMsg`
 method), [57](#)
`visibilityChanged()`
 (`vi.widgets.userlogoutmsg.UserLogoutMsg`
 method), [51](#)

W

`warn()` (`webworker_scripts.weblog` static method), [76](#)
`weblog` (class in `webworker_scripts`), [76](#)
`webworker_scripts`
 module, [75](#)
`workerFeedback()` (`vi.widgets.code.PythonCode`
 method), [32](#)
`writeRow()` (`webworker_scripts.csvWriter` method), [76](#)
`writeRows()` (`webworker_scripts.csvWriter` method), [76](#)