
ViUR Vi

Release 3.0

Tilman Oestereich, Andreas H. Kelch

Jul 18, 2023

1 About	3
Python Module Index	77
Index	79



Administrativ viusal interface for the ViUR Information System

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()
    onDetach()
    onSelectionChanged(table, selection, *args, **kwargs)
    onClick(sender=None)
    openModule(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(sender=None)
    resetLoadingState()

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

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

```

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

    onClick(sender=None)

    resetLoadingState()

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

    onClick(sender=None)

    resetLoadingState()

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

    onClick(sender=None)

    performReload(sender=None)

    resetLoadingState()

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

    onClick(sender=None)

    resetLoadingState()
```

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(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(sender=None)
    createDir(dialog, dirName)
    onMkDir(req)
    resetLoadingState()

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(sender=None)
    resetLoadingState()

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()
    onDetach()
    onSelectionActivated(table, selection)
    onSelectionChanged(table, selection, *args, **kwargs)
    static isSuitableFor(module, handler, actionName)
    onClick(sender=None)
    editDir(dialog, dirName)
    resetLoadingState()

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

```

```
onClick(sender=None)

disableViUnloadingWarning(*args, **kwargs)

enableViUnloadingWarning(*args, **kwargs)

doDownload(fileData)

resetLoadingState()
```

`vi.actions.hierarchy`

Module Contents

Classes

<code>AddAction</code>	Adds a new node in a hierarchy application.
<code>EditAction</code>	Edits a node in a hierarchy application.
<code>CloneAction</code>	Allows cloning an entry (including its subentries) in a hierarchy application.
<code>DeleteAction</code>	Deletes a node from a hierarchy application.
<code>ReloadAction</code>	Allows adding an entry in a list-module.
<code>ListViewAction</code>	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(sender=None)
```

```
resetLoadingState()
```

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

Bases: `flare.button.Button`

Edits a node in a hierarchy application.

```
onAttach()
```

```
onDetach()
```

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

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

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

```
onClick(sender=None)
```

```
openEditor(key)
```

```
resetLoadingState()
```

```

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

class vi.actions.hierarchy.DeleteAction(*args, **kwargs)
    Bases: flare.button.Button
    Deletes a node from a hierarchy application.
    onAttach()
    onDetach()
    onSelectionChanged(table, selection, *args, **kwargs)
    static isSuitableFor(module, handler, actionName)
    onClick(sender=None)
    doDelete(dialog)
    allDeletedSuccess(success)
    resetLoadingState()

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(sender=None)
    resetLoadingState()

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(sender=None)
    resetLoadingState()

```

`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(sender=None)

    resetLoadingState()

class vi.actions.list.EditAction(*args, **kwargs)
    Bases: flare.button.Button
    Allows editing an entry in a list-module.

    onAttach()

    onDetach()

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

    onSelectionActivated(table, selection)

    static isSuitableFor(module, handler, actionName)

    onClick(sender=None)

    openEditor(key)

    resetLoadingState()

class vi.actions.list.CloneAction(*args, **kwargs)
    Bases: flare.button.Button
    Allows cloning an entry in a list-module.

    onAttach()

    onDetach()

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

    static isSuitableFor(module, handler, actionName)

    onClick(sender=None)

    openEditor(key)

    resetLoadingState()

class vi.actions.list.DeleteAction(*args, **kwargs)
    Bases: flare.button.Button
    Allows deleting an entry in a list-module.

    onAttach()

    onDetach()

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

    static isSuitableFor(module, handler, actionName)

    onClick(sender=None)

    doDelete(dialog)

```

```
allDeletedSuccess(success)

deletedSuccess(req=None, code=None)

deletedFailed(req=None, code=None)

resetLoadingState()

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

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

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

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

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



```

    doCancel(*args, **kwargs)

    doSelectAll(*args, **kwargs)

    doUnselectAll(*args, **kwargs)

    doInvertSelection(*args, **kwargs)

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

    onAttach()

    onDetach()

    onTableChanged(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(event=None)

    resetLoadingState()

class vi.actions.list.TableNextPage(*args, **kwargs)
    Bases: flare.button.Button
    postInit(widget=None)

    onClick(sender=None)

    static isSuitableFor(module, handler, actionName)

    resetLoadingState()

class vi.actions.list.TablePrevPage(*args, **kwargs)
    Bases: flare.button.Button
    postInit(widget=None)

    onClick(sender=None)

    static isSuitableFor(module, handler, actionName)

class vi.actions.list.TableItems(*args, **kwargs)
    Bases: flare.html5.Div
    postInit(widget=None)

    onTableChanged(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(sender=None)
```

```
    onChange(sender=None)
```

```
    setPageAmount()
```

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

```
    resetLoadingState()
```

```
class vi.actions.list.LoadNextBatchAction(*args, **kwargs)
```

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

```
    Load a bunch of pages
```

```
    registerScroll()
```

```
    onScroll(sender=None)
```

```
    onClick(sender=None)
```

```
    onChange(sender=None)
```

```
    loadnextPages(*args, **kwargs)
```

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

```
    resetLoadingState()
```

```
class vi.actions.list.LoadAllAction(*args, **kwargs)
```

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

```
    Allows Loading all Entries in a list
```

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

```
    onClick(sender=None)
```

```
    loadAllRows()
```

```
    resetLoadingState()
```

```
class vi.actions.list.PageFindAction(*args, **kwargs)
```

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

```
    Allows Loading all Entries in a list
```

```
    onKeyPress(event)
```

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

```
    onClick(sender=None)
```

```
    startFind()
```

```
    findText()
```

```

    resetLoadingState()

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

    onAttach()

    onClick(sender=None)

    static isSuitableFor(module, handler, actionName)

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

    static isSuitableFor(module, handler, actionName)

    onClick(sender=None)

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

    onClick(sender=None)

    static isSuitableFor(module, handler, actionName)

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

    static isSuitableFor(module, handler, actionName)

    onClick(sender=None)

    onAttach()

    onDetach()

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

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

    static isSuitableFor(module, handler, actionName)

    onClick(sender=None)

    onAttach()

    onDetach()

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

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

    static isSuitableFor(module, handler, actionName)

    onClick(sender=None)

    onAttach()

    onDetach()

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

```

`vi.actions.list_order`

Module Contents

Classes

<i>ShopMarkAction</i>
<i>ShopMarkPayedAction</i>
<i>ShopMarkSentAction</i>
<i>ShopMarkCanceledAction</i>

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

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

```
    onAttach()
```

```
    onDetach()
```

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

```
    setPayed(order)
```

```
    setPayedSucceeded(response)
```

```
    setPayedFailed(response)
```

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

```
    onClick(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

<code>AddLeafAction</code>	Creates a new leaf (ie. a file) for a tree application
<code>AddNodeOnlyAction</code>	Adds a new node in a hierarchy application.
<code>AddNodeAction</code>	Creates a new node (ie. a directory) for a tree application
<code>EditAction</code>	Edits an entry inside a tree application.
<code>DeleteAction</code>	Allows deleting an entry in a tree-module.
<code>ReloadAction</code>	Allows adding an entry in a list-module.
<code>SelectRootNode</code>	Selector for hierarchy root nodes.

```
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(*sender=None*)

resetLoadingState()

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

Bases: `flare.button.Button`

Adds a new node in a hierarchy application.

static `isSuitableFor(module, handler, actionName)`

onClick(*sender=None*)

resetLoadingState()

```
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(*sender=None*)

resetLoadingState()

```
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()

onDetach()

```
onSelectionActivated(table, selection)  
onSelectionChanged(table, selection, *args, **kwargs)  
static isSuitableFor(module, handler, actionName)  
onClick(sender=None)  
resetLoadingState()
```

```
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()  
    onDetach()  
    onSelectionChanged(table, selection, *args, **kwargs)  
    static isSuitableFor(module, handler, actionName)  
    onClick(sender=None)  
    doDelete(dialog)  
    allDeletedSuccess(success)  
    resetLoadingState()
```

```
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(sender=None)  
    resetLoadingState()
```

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

```
    Bases: flare.html5.Select  
    Selector for hierarchy root nodes.  
    onAttach()  
    onDetach()  
    update()  
    onRootNodeChanged(newNode, *args, **kwargs)  
    onRootNodesAvailable(req)  
    onChange(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(*actions, widget=None, view=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()

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()

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. (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()

setHeader(headers)

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

getTrByIndex(idx)

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

getIndexByTr(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(event)

Determines the row number for the given event

onChange(event)

onMouseDown(event)

onMouseOut(event)

onMouseUp(event)

onKeyDown(event)

onKeyUp(event)

onDbClick(event)

addSelectedRow(row)

Marks a row as selected

removeSelectedRow(row)

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

selectRow(newRow)

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

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

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

focusRow(*row*)

getCurrentSelection()

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

clear()

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

removeRow(*row*)

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

_extraCols()

prepareCol(*row*, *col*)

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

dropTableContent()

Drops content from the table, structure remains unchanged

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

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

selectAll()

Selects all entries of the table.

unselectAll()

Unselects all entries of the table.

invertSelection()

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(*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(*table*, *row*)

Ensure the table scrolls according to the position of its cursor

getRowCount()

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

add(*obj*)

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

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

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

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

remove(*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(*keepModel=False*)

Flushes the whole table.

_renderObject(*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(*recalculate=True*)

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

setShownFields(*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(*table, rows, *args, **kwargs*)

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

onSelectionActivated(*table, rows*)

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

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

Re-emit the event.

getCurrentSelection()

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

setCellRender(*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(*renders*)

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

activateCurrentSelection()

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(*keepModel=False*)

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

rebuildTable(*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(*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(*objList*, *writeToModel=True*)

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

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

Override explanation - removed grid preparation

_renderObject(*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(*args, **kwargs)

reevaluate(*args, **kwargs)

focus()

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

Bases: flare.html5.Div

onClick(event)

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

onAttach()

onDetach()

onStartSearch(searchTxt=None)

setView(btn)

applyFilter(filter, filterID, filterName)

`vi.sidebarwidgets.internalpreview`

Module Contents

Classes

InternalPreview

```
class vi.sidebarwidgets.internalpreview.InternalPreview(module, structure, item, *args, **kwargs)
    Bases: flare.html5.U1
    onCopyKey(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
    initWidget()
        Here we start!

```

vi.views.hierarchy

Module Contents

Classes

*hierarchyHandler**hierarchyHandlerWidget*

```

class vi.views.hierarchy.hierarchyHandler
    Bases: flare.views.view.View
    static canHandle(moduleName, moduleInfo)
class vi.views.hierarchy.hierarchyHandlerWidget
    Bases: flare.views.view.ViewWidget
    initWidget()
        Here we start!

```

vi.views.list

Module Contents

Classes

*listHandler**listHandlerWidget*

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

class vi.views.list.listHandlerWidget
    Bases: flare.views.view.ViewWidget
    initWidget()
        Here we start!
    onViewfocusedChanged(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()
        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()
```

```
    Here we start!
```

```
vi.views.overview
```

Module Contents

Classes

<i>Overview</i>
<i>OverviewWidget</i>

```
class vi.views.overview.Overview
```

```
    Bases: flare.views.view.View
```

```
class vi.views.overview.OverviewWidget
```

```
    Bases: flare.views.view.ViewWidget
```

```
    initWidget()
```

```
        Here we start!
```

```
vi.views.singleton
```

Module Contents

Classes

<i>singletonHandler</i>
<i>singletonHandlerWidget</i>

```
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()
```

```
        Here we start!
```

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

`vi.views.tree`

Module Contents

Classes

<i>treeHandler</i>

<i>treeHandlerWidget</i>

```
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()
        Here we start!
```

`vi.widgets`

Submodules

`vi.widgets.accordion`

Module Contents

Classes

<i>AccordionSegment</i>

<i>Accordion</i>

```
class vi.widgets.accordion.AccordionSegment(ident, title=None)
    Bases: flare.html5.Fieldset
    checkVisibility()
    activate()
    deactivate()
    isActive()
    toggle()
```



```
onClick(event)

addWidget(widget)

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

    clear()

    buildAccordion(order=None)

        Parameters
            sort – None: sorted by Bones, “asc”:ascending, “desc”:descending, dict: {“cate-
                gory”: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
```

```
"""
    <div [name]="item" class="item has-hover">
        <a class="item-link" @click="navigationAction">
            <div class="item-image">
                <flare-icon value="{{icon}}" title="
↪ {{name}}"></flare-icon>
                                </div>

                                <div class="item-content">
                                    <div class="item-headline">{{name}}
↪ </div>
                                </div>
```

(continues on next page)

(continued from previous page)

```

        </a>
        <span [name]="itemArrow" class="item-open is-hidden"
        ↪ " @click="ArrowAction">
            <flare-svg-icon value="icon-arrow-left"></
        ↪ flare-svg-icon>
        </span>
        <span [name]="itemRemove" class="item-pin is-hidden"
        ↪ " @click="RemoveAction">
            <flare-svg-icon value="icon-cancel"></flare-
        ↪ svg-icon>
        </span>
    </div>
    <div [name]="subItem" class="list list--sub">
    </div>
    ""

```

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

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

Handle Click on Navigation Button

RemoveAction(*e=None*)

remove this Nav Element

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

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

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

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

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

appendSubChild(*element*)

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

Bases: `flare.html5.Div`

buildSeperator()

_setValue(*value*)

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

Bases: `flare.html5.Div`

addSeperator()

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

class `vi.widgets.appnavigation.AppNavigation`

Bases: `flare.html5.Nav`

getPreviousNavigationPoint(*view*)

getNavigationPoint(*view*)

```
addNavigationBlock(name)
```

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

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

```
removeNavigationPoint(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()
```

```
    onAttach()
```

```
    onDetach()
```

```
    _getValue()
```

```
    _setValue(val)
```

```
    insertText(text)
```

```
    setCursor(line, char)
```

```
class vi.widgets.code.PythonCode(logger, scripter)
```

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

```
    addToLog(data, type='normal')
```

```
        allowed types: normal, info, warn, error
```

workerFeedback(*e*)

run()

stop()

class `vi.widgets.code.Scripter`(*coder=PythonCode, exampleCode=None, executable=True*)

Bases: `flare.html5.Div`

runClick(*event*)

killClick(*event*)

openHelp(*event*)

startFullscreen(*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(*cursor=None*)

nextChunkComplete(*req*)

exportToFile()

nextChunkFailure(*req, code*)

replaceWithMessage(*message, logClass='success'*)

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

Bases: `flare.popup.Popup`

onExportBtnClick(**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()``onAttach()``onChange(event)``onBoneChange(bone)``showErrorMsg(req=None, code=None)`

Removes all currently visible elements and displays an error message

`reloadData()`

_save(*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()

Removes all visible bones/forms/fieldsets.

closeOrContinue(*sender=None*)**doCloneHierarchy(*sender=None*)****cloneComplete(*req*)****setData(*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(*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(event)
```

```
    onDownloadBtnClick(sender=None)
```

```

class vi.widgets.file.FilePreviewImage(file=None, size=150, *args, **kwargs)
    Bases: flare.html5.Div
    setFile(file)
    download()
    onClick(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(req)
        Internal callback - the actual upload url (retrieved by calling /file/getUploadURL) is known.
    onLoad(*args, **kwargs)
        Internal callback - The state of our upload changed.
    onUploadAdded(req)
    onProgress(event)
        Internal callback - further bytes have been transmitted
    onSuccess(*args, **kwargs)
        Internal callback - The upload succeeded.
    onFailed(errorCode, *args, **kwargs)
    replaceWithMessage(message, isSuccess)

class vi.widgets.file.MultiUploader(files, node, context=None, module='file', *args, **kwargs)
    Bases: flare.html5.Div
    handleFile(file)
    onUploadUrlAvailable(req)
        Internal callback - the actual upload url (retrieved by calling /file/getUploadURL) is known.
    onLoad(*args, **kwargs)
        Internal callback - The state of our upload changed.
    onUploadAdded(req)
    onSuccess(*args, **kwargs)
        Internal callback - The upload succeeded.
    onFailed(errorCode, *args, **kwargs)
    replaceWithMessage(message, isSuccess)
    closeMessage()

class vi.widgets.file.FileLeafWidget
    Bases: vi.widgets.tree.TreeLeafWidget
    EntryIcon()
    setStyle()

```

```
class vi.widgets.file.FileNodeWidget
    Bases: vi.widgets.tree.TreeNodeWidget
    setStyle()

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()
    onStartSearch(searchStr, *args, **kwargs)
    getChildKey(widget)
        Derives a string used to sort the entries on each level
    onDrop(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 distinction!

```
class vi.widgets.hierarchy.HierarchyWidget(*args, **kwargs)
    Bases: vi.widgets.tree.TreeWidget
    A Hierarchy is a Tree without leaf distinction!
    leafWidget
    getActions()
        Returns a list of actions that are being set for the ActionBar. Override this to provide additional actions.
    reloadData()
        Reload the data were displaying.
    reloadListWidget()
    toggleListView()
    setListView(visible=False)
    showListView()
```



```

hideListView()

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

static canHandle(moduleName, moduleInfo)

```

`vi.widgets.internaedit`

Module Contents

Classes

ParsedErrorItem

PassiveErrorItem

InternalEdit

Functions

checkErrors($\rightarrow$ Tuple[bool, List[str]])

```
class vi.widgets.internaedit.ParsedErrorItem(error)
```

Bases: flare.html5.Li

```
style = []
```

```
class vi.widgets.internaedit.PassiveErrorItem(error)
```

Bases: flare.html5.Li

```
style = []
```

```
vi.widgets.internaedit.checkErrors(bone)  $\rightarrow$  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(readOnly=False)
```

```
serializeForPost(validatorCheck=False)
```

```
serializeForDocument()
```

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

Starts serializing and transmitting our values to the server.

unserialize(*data=None*)

Applies the actual data to the bones.

onChange(*event*)

onKeyDown(*event*)

performLogics()

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

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

selectorReturn()

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

tableInitialization(**args, **kwargs*)

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

setAmount(*amount*)

setPage(*page=0*)

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

onRequestingFinished(**args, **kwargs*)

onClick(*event*)

setTableActionBar()

getDefaultEntryActions()

Returns the list of actions available in our actionBar

getActions()

Returns the list of actions available in our actionBar

getAllActions(*view=None*)

Returns the list of actions available in the action bar.

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

Removes all currently visible elements and displays an error message

onNextBatchNeeded()

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

onAttach()

onDetach()

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

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

requestStructure()

receivedStructure(*resp*)

reloadData()

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

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

Applies a new filter.

setContext(*context*)

Applies a new context.

getFilter()

updateEmptyNotification()

onCompletion(*req*)

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

setFields(*fields*)

getFields()

onSelectionActivated(*table, selection*)

activateSelection()

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(**args, **kwargs*)

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

Override explanation

- use ViewPort DataTable with rows parameter

setAmount(*amount*)

setPage(*page=0*)

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

Parameters**page** – sets targetpage**Returns****_setPage**(*page=0*)

render page to table :param page: :return:

onRequestingFinished(**args, **kwargs*)**setTableActionBar**()**static canHandle**(*moduleName, moduleInfo*)**vi.widgets.search****Module Contents****Classes**

Search

class vi.widgets.search.**Search**(**args, **kwargs*)

Bases: flare.html5.Div

doSearch(**args, **kwargs*)**resetSearch**()**onKeyDown**(*event*)**resetLoadingState**()**reevaluate**()**focus**()**vi.widgets.sidebar****Module Contents****Classes**

SideBar

```

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

```

vi.widgets.table

Module Contents

Classes

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

setHeader(headers)

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

getTrByIndex(idx)

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

getIndexByTr(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(event)

Determines the row number for the given event

onChange(*event*)

onMouseDown(*event*)

onMouseOut(*event*)

onMouseUp(*event*)

onKeyDown(*event*)

onKeyUp(*event*)

onDbClick(*event*)

addSelectedRow(*row*)

Marks a row as selected

removeSelectedRow(*row*)

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

selectRow(*newRow*)

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

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

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

focusRow(*row*)

getCurrentSelection()

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

clear()

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

removeRow(*row*)

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

_extraCols()

prepareCol(*row*, *col*)

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

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

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

selectAll()

Selects all entries of the table.

unselectAll()

Unselects all entries of the table.

invertSelection()

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(**args, **kwargs*)

setDataProvider(*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(*table, row*)

Ensure the table scrolls according to the position of its cursor

getRowCount()

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

add(*obj*)

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

extend(*objList*)

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

testIfNextBatchNeededImmediately()

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

remove(*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(*keepModel=False*)

Flushes the whole table.

_renderObject(*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()

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

setShownFields(*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(*event*)

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

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

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

onSelectionActivated(*table, rows*)

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

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

Re-emit the event.

getCurrentSelection()

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

setCellRender(*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(*renders*)

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

activateSelection()

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

vi.widgets.task

Module Contents

Classes

<i>TaskWidget</i>
<i>ServerTaskWidget</i>
<i>TaskSelectWidget</i>

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()

setActiveTask()

onChange(*event*)

invokeTask(**args*, ***kwargs*)

vi.widgets.tooltip**Module Contents****Classes**

<i>ToolTip</i>	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(*event*)

_setDisabled(*disabled*)

_getDisabled()

vi.widgets.topbar**Module Contents****Classes**

<i>TopBarWidget</i>	Provides the top-bar of VI
<i>UserState</i>	
<i>Tasks</i>	
<i>Logout</i>	
<i>Scripter</i>	

class vi.widgets.topbar.TopBarWidget

Bases: flare.html5.Header

Provides the top-bar of VI

invoke()

setTitle(*title=None*)

onClick(*event*)

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

class vi.widgets.topbar.UserState(*args, **kwargs)

Bases: flare.html5.Div

```
    onCurrentUserAvailable(req)

    update()

    static canHandle(action)

    onClick(sender=None)

    openEdit(key)

class vi.widgets.topbar.Tasks(*args, **kwargs)
    Bases: flare.html5.Div
    onTaskListAvailable(req)

    onTaskListFailure()

    onCurrentUserAvailable(req)

    update()

    onClick(event)

    static canHandle(action)

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

    logout()

    static canHandle(action)

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

    updateUser()

    onClick(event=None)

    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()

receivedStructure(*resp*)

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

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

setContext(*context*)

selectorReturn()

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

onKeyDown(*event*)

onKeyUp(*event*)

getActions()

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

clearSelection()

Empties the current selection.

extendSelection(*element*)

Extends the current selection to element.

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

activateSelection(*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(*element*)

_showErrorMsg(*req=None*, *code=None*)

Removes all currently visible elements and displays an error message

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

onAttach()

onDetach()

itemForKey(*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(*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(*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()

Reload the data were displaying.

loadNode(*node*, *skelType=None*, *cursor=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(*req*)

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

onDrop(*event*)

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

onDragOver(*event*)

Allow dropping children on the rootNode

getChildKey(*widget*)

Order by sortindex

static canHandle(*moduleName*, *moduleInfo*)

class vi.widgets.tree.**BrowserLeafWidget**

Bases: flare.viur.widgets.tree.TreeLeafWidget

setStyle()

class vi.widgets.tree.**BrowserNodeWidget**

Bases: flare.viur.widgets.tree.TreeNodeWidget

setStyle()

class vi.widgets.tree.**BreadcrumbNodeWidget**

Bases: flare.viur.widgets.tree.TreeNodeWidget

setStyle()

```
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()

Reload the data were displaying.

rebuildPath()

Rebuild the displayed path-list.

onPathRequestSucceeded(*req*)

Rebuild the displayed path-list according to request data

activateSelection(*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

checkInterval

visibilityChanged(*e*)

stopInterval()

hideMessage()

Make this popup invisible

showMessage()

Show this popup

showLoginWindow(*args, **kwargs)

Return to the login window.

checkForSuspendResume(*args, **kwargs)

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

startPolling(*args, **kwargs)

Start querying the server

onUserTestSuccess(req)

We received a response from the server

onUserTestFail(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>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 distinction!
<i>FileWidget</i>	Base Widget that renders a tree.

class vi.widgets.**Accordion**

Bases: flare.html5.Form

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

clear()

buildAccordion(*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()

setTitle(*title=None*)

onClick(*event*)

setCurrentModulDescr(*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(*callback, multi=True, allow=None*)

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

selectorReturn()

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

tableInitialization(**args, **kwargs*)

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

setAmount(*amount*)

setPage(*page=0*)

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

onRequestingFinished(**args, **kwargs*)

onClick(*event*)

setTableActionBar()

getDefaultEntryActions()

Returns the list of actions available in our actionBar

getActions()

Returns the list of actions available in our actionBar

getAllActions(*view=None*)

Returns the list of actions available in the action bar.

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

Removes all currently visible elements and displays an error message

onNextBatchNeeded()

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

onAttach()

onDetach()

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

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

requestStructure()

receivedStructure(*resp*)

reloadData()

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

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

Applies a new filter.

setContext(*context*)

Applies a new context.

getFilter()

updateEmptyNotification()

onCompletion(*req*)

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

setFields(*fields*)

getFields()

onSelectionActivated(*table*, *selection*)

activateSelection()

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()

onAttach()

onChange(*event*)

onBoneChange(*bone*)

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

Removes all currently visible elements and displays an error message

reloadData()

_save(*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()

Removes all visible bones/forms/fieldsets.

closeOrContinue(*sender=None*)

doCloneHierarchy(*sender=None*)

cloneComplete(*req*)

setData(*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(*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(*event*)

_setDisabled(*disabled*)

_getDisabled()

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

Bases: `flare.html5.Div`

Provides kind of MVC on top of SelectTable.

recalcHeight(**args, **kwargs*)

setDataProvider(*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(*table, row*)

Ensure the table scrolls according to the position of its cursor

getRowCount()

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

add(obj)

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

extend(objList)

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

testIfNextBatchNeededImmediately()

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

remove(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(keepModel=False)

Flushes the whole table.

_renderObject(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()

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

setShownFields(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(event)

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

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

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

onSelectionActivated(table, rows)

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

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

Re-emit the event.

getCurrentSelection()

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

setCellRender(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(renderers)

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

activateSelection()

Emits the selectionActivated event if there's currently a selection

```

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

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

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

    setActiveTask()

    onChange(event)

    invokeTask(*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(readOnly=False)

    serializeForPost(validityCheck=False)

    serializeForDocument()

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

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

    onChange(event)

    onKeyDown(event)

    performLogics()

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

    Bases: flare.html5.Progress

    nextChunk(cursor=None)

    nextChunkComplete(req)

    exportToFile()

    nextChunkFailure(req, code)

    replaceWithMessage(message, logClass='success')

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

    onExportBtnClick(*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()

receivedStructure(*resp*)

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

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

setContext(*context*)

selectorReturn()

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

onKeyDown(*event*)

onKeyUp(*event*)

getActions()

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

clearSelection()

Empties the current selection.

extendSelection(*element*)

Extends the current selection to element.

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

activateSelection(*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(*element*)

_showErrorMsg(*req=None*, *code=None*)

Removes all currently visible elements and displays an error message

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

onAttach()

onDetach()

itemForKey(*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(*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(*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()

Reload the data were displaying.

loadNode(*node*, *skelType=None*, *cursor=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(*req*)

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

onDrop(*event*)

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

onDragOver(*event*)

Allow dropping children on the rootNode

getChildKey(*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()**

Reload the data were displaying.

rebuildPath()

Rebuild the displayed path-list.

onPathRequestSucceeded(*req*)

Rebuild the displayed path-list according to request data

activateSelection(*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**getActions()**

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

```

reloadData()
    Reload the data were displaying.
reloadListWidget()
toggleListView()
setListView(visible=False)
showListView()
hideListView()
onSelectionChanged(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()
onStartSearch(searchStr, *args, **kwargs)
getChildKey(widget)
    Derives a string used to sort the entries on each level
onDrop(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

`AdminScreen`

This is the screen superclass.

Attributes

viInitializedEvent

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

Bases: `vi.screen.Screen`

This is the screen superclass.

It represents a basic screen and its functionality.

onClick(*event*)

reset()

invoke()

Is called to show the screen

getCurrentUser()

getCurrentUserSuccess(*req*)

getCurrentUserFailure(*req, code*)

refresh()

refreshConfig(*req*)

startup()

initializeViews()

initializeConfig()

appendNavList(*NavList, target, parentInfo=()*)

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

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

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

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

checkInitialHash(**args, **kwargs*)

execCall(*path, params=None*)

Performs an execution call.

Parameters

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

stackWidget(*widget, title="", icon=None*)

We dont stack widgets anymore. We use now Popups.


```

removeWidget(widget)
switchFullscreen(fullscreen=True)
isFullscreen()
onError(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(sender=None)
```

```
    openEditor(key)
```

```
class vi.log.logWidget(logList)
```

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

```
    buildDataTable()
```

```
class vi.log.LogButton
```

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

```
    idbdata(event)
```

```
    cleanLog()
```

```
    cleanLogAction(event)
```

```
    renderPopOut()
```

```

onClick(sender=None)

openLog()

    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(type, msg, icon=None, modul=None, action=None, key=None, data=None, date=None, onlyLoad=False)

msgOverlay(logObj)

removeInfo(wrap)

reset()

static canHandle(action)

class vi.log.Log
    Bases: flare.html5.Div
    Provides the "messaging" center displayed at the bottom of VI

    toggleMsgCenter(*args, **kwargs)

    log(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(span)

    reset()

```

vi.login

Module Contents

Classes

<i>LoginInputField</i>	
<i>BaseLoginHandler</i>	
<i>UserPasswordLoginHandler</i>	
<i>GoogleAccountLoginHandler</i>	
<i>OAuthAccountLoginHandler</i>	
<i>LoginScreen</i>	This is the screen superclass.

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

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

class vi.login.UserPasswordLoginHandler(loginScreen, *args, **kwargs)
    Bases: BaseLoginHandler
    cssname = 'userpassword'
    onKeyPress(event)
    onLoginClick(sender=None)
    doLoginSuccess(req)
    doLoginFailure(req, code, *args, **kwargs)
    onVerifyClick(sender=None)
    doVerifySuccess(req)
    doVerifyFailure(*args, **kwargs)
    onSendClick(sender=None)
    reset()
    enable()
    focusLaterIdiot()
    static canHandle(method, secondFactor)

class vi.login.GoogleAccountLoginHandler(loginScreen, *args, **kwargs)
    Bases: BaseLoginHandler
    cssname = 'googleaccount'
    onLoginClick(sender=None)
```

```

    static canHandle(method, secondFactor)

class vi.login.OAuthAccountLoginHandler(loginScreen, *args, **kwargs)
    Bases: BaseLoginHandler
    cssname = 'oauthaccount'
    onLoginClick(sender=None)
    doKeySuccess(req)
    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(logout=False)
        Is called to show the screen
    onLogoutSuccess(*args, **kwargs)
    doShowLogin(req, code, *args, **kwargs)
    insufficientRights()
    doSkipLogin(req)
    onGetAuthMethodsSuccess(req)
    selectHandler(handler=None)
    onGetAuthMethodsFailure(*args, **kwargs)
    redirectNoAdmin()
```

vi.pane

Module Contents

Classes

<i>Pane</i>	Base class for Panes.
<i>GroupPane</i>	This pane groups subpanes; it cannot have direct chil-
	drens

```

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__(*key, value*)

setImage(*loading=False*)

lock()

unlock()

setText(*descr=None, iconURL=None, loading=False*)

onBtnCloseReleased(*args, **kwargs)

addChildPane(*pane*)

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

removeChildPane(*pane*)

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

onDetach()

addWidget(*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()

removeWidget(*widget*)

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

containsWidget(*widget*)

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

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

expand()

collapse()

focus()

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

Bases: [Pane](#)

This pane groups subpanes; it cannot have direct childrens

loadChildren()

DeferredLoadChildren(*delay=1000*)

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

expand()

collapse()

onFocus(*event*)

vi.priorityqueue

Module Contents

Classes

<i>StartupQueue</i>

Attributes

<i>HandlerClassSelector</i>
<i>actionDelegateSelector</i>
<i>initialHashHandler</i>
<i>extendedSearchWidgetSelector</i>
<i>toplevelActionSelector</i>
<i>loginHandlerSelector</i>
<i>startupQueue</i>

```
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()
    setFinalElem(elem)
    insertElem(priority, elem)
    run()
    next()
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()
    unlock()
    invoke()
        Is called to show the screen
    remove()
        Remove the screen from its parent
    setTitle(title=None)
```

`vi.serversideaction`

Module Contents

Classes

ServerSideActionWdg

```
class vi.serversideaction.ServerSideActionWdg(module, handler, actionName, actionData)
    Bases: flare.button.Button
    switchDisabledState(disabled)
    onAttach()
    onDetach()
    onSelectionChanged(table, selection, *args, **kwargs)
    onClick(sender=None)
    apply(sender=None)
```



```

fetchNext()
fetchSucceeded(req)
fetchFailed()
resetLoadingState()

```

vi.utils

Module Contents

Classes

indexeddbConnector

indexeddb

Functions

<i>render_url</i> (url, module[, entry])	Renders a URL that contains {{variables}}, e.g. for previews.
<i>formatString</i> (format, data[, structure, language])	Parses a string given by format and substitutes placeholders using values specified by data.
<i>getImagePreview</i> (data[, cropped, size])	
<i>setPreventUnloading</i> ([mode])	
<i>mergeDict</i> (original, target)	

vi.utils.render_url(url, module, entry=None, ***kwargs*)

Renders a URL that contains {{variables}}, e.g. for previews.

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()

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

Package Contents

Classes

LoginScreen	This is the screen superclass.
AdminScreen	This is the screen superclass.
Application	

Functions

preloadIcons()

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(*logout=False*)

Is called to show the screen

onLogoutSuccess(*args, **kwargs)

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

insufficientRights()

doSkipLogin(*req*)

onGetAuthMethodsSuccess(*req*)

selectHandler(*handler=None*)

onGetAuthMethodsFailure(*args, **kwargs)

redirectNoAdmin()

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

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

This is the screen superclass.

It represents a basic screen and its functionality.

onClick(event)

reset()

invoke()

Is called to show the screen

getCurrentUser()

getCurrentUserSuccess(req)

getCurrentUserFailure(req, code)

refresh()

refreshConfig(req)

startup()

initializeViews()

initializeConfig()

appendNavList(NavList, target, parentInfo=())

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

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

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

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

checkInitialHash(*args, **kwargs)

execCall(path, params=None)

Performs an execution call.

Parameters

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

stackWidget(widget, title="", icon=None)

We dont stack widgets anymore. We use now Popups.

removeWidget(widget)

switchFullscreen(fullscreen=True)

```

    isFullscreen()

    onError(req, code)
vi.conf
class vi.Application
    Bases: flare.html5.Div
    startup(*args, **kwargs)
    getVersionSuccess(req)
    getConfigSuccess(req)
    startupFailure(req, err)
    login(logout=False)
    admin()
    logout()
    setTitle(title=None)
    setPath(path='')
vi.preloadIcons()
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

<i>requestList</i>	
<i>csvWriter</i>	
<i>weblog</i>	
<i>HTTPRequest</i>	Wrapper around XMLHttpRequest
<i>SimpleNetwork</i>	

Functions

<i>request</i> (url[, params, jsonResult, whitelist])	A very simple version of the NetworkService to request synchronous data
---	---

Attributes

<i>log</i>

`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()`

`next()`

`running()`

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

`delimiter = ';'`

`writeRow(row)`

`writeRows(rows)`

`download(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(*args, **kwargs)

Internal callback.

class webworker_scripts.**SimpleNetwork**

Bases: object

genReqStr(*params*)

request(*url, params*)

onCompletion(*text*)

onError(*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](#), 16
- [vi.actions.tree](#), 17
- [vi.admin](#), 59
- [vi.config](#), 61
- [vi.exception](#), 61
- [vi.framework](#), 19
- [vi.framework.components](#), 19
- [vi.framework.components.actionbar](#), 19
- [vi.framework.components.datatable](#), 19
- [vi.log](#), 62
- [vi.login](#), 63
- [vi.pane](#), 65
- [vi.priorityqueue](#), 67
- [vi.screen](#), 68
- [vi.serversideaction](#), 68
- [vi.sidebarwidgets](#), 23
- [vi.sidebarwidgets.filterselector](#), 23
- [vi.sidebarwidgets.internalpreview](#), 24
- [vi.translations](#), 24
- [vi.translations.de](#), 24
- [vi.translations.en](#), 24
- [vi.utils](#), 69
- [vi.views](#), 24
- [vi.views.edit](#), 24
- [vi.views.hierarchy](#), 25
- [vi.views.list](#), 25
- [vi.views.log](#), 26
- [vi.views.notfound](#), 26
- [vi.views.overview](#), 27
- [vi.views.singleton](#), 27
- [vi.views.tree](#), 28
- [vi.widgets](#), 28
- [vi.widgets.accordion](#), 28
- [vi.widgets.appnavigation](#), 29

- [vi.widgets.code](#), 31
- [vi.widgets.csvexport](#), 32
- [vi.widgets.edit](#), 33
- [vi.widgets.file](#), 34
- [vi.widgets.hierarchy](#), 36
- [vi.widgets.internaledit](#), 37
- [vi.widgets.list](#), 38
- [vi.widgets.search](#), 40
- [vi.widgets.sidebar](#), 40
- [vi.widgets.table](#), 41
- [vi.widgets.task](#), 44
- [vi.widgets.tooltip](#), 45
- [vi.widgets.topbar](#), 45
- [vi.widgets.tree](#), 46
- [vi.widgets.userlogoutmsg](#), 49

W

- [webworker_scripts](#), 73

Symbols

- `__editIdx_` (vi.widgets.EditWidget attribute), 52
 - `__editIdx_` (vi.widgets.edit.EditWidget attribute), 33
 - `__setattr__` (vi.pane.Pane method), 66
 - `_attachCodemirror()` (vi.widgets.code.Codemirror method), 31
 - `_deleteFromStore()` (vi.utils.indexeddb method), 70
 - `_deleteObjectStore()` (vi.utils.indexeddb method), 70
 - `_extraCols()` (vi.framework.components.datatable.SelectTable method), 21
 - `_extraCols()` (vi.widgets.table.SelectTable method), 42
 - `_getDisabled()` (vi.widgets.ToolTip method), 53
 - `_getDisabled()` (vi.widgets.tooltip.ToolTip method), 45
 - `_getList()` (vi.utils.indexeddb method), 70
 - `_getListKey()` (vi.utils.indexeddb method), 70
 - `_getValue()` (vi.widgets.code.Codemirror method), 31
 - `_onRequestSucceeded()` (vi.widgets.TreeWidget method), 58
 - `_onRequestSucceeded()` (vi.widgets.tree.TreeWidget method), 48
 - `_processDbUpdate()` (vi.utils.indexeddb method), 70
 - `_processQueue()` (vi.utils.indexeddb method), 70
 - `_registerObjectStore()` (vi.utils.indexeddb method), 70
 - `_renderObject()` (vi.framework.components.datatable.DataTable method), 22
 - `_renderObject()` (vi.framework.components.datatable.ViewportDataTable method), 23
 - `_renderObject()` (vi.widgets.DataTable method), 54
 - `_renderObject()` (vi.widgets.table.DataTable method), 43
 - `_rowForEvent()` (vi.framework.components.datatable.SelectTable method), 20
 - `_rowForEvent()` (vi.widgets.table.SelectTable method), 41
 - `_save()` (vi.widgets.EditWidget method), 53
 - `_save()` (vi.widgets.edit.EditWidget method), 33
 - `_setDisabled()` (vi.widgets.ToolTip method), 53
 - `_setDisabled()` (vi.widgets.tooltip.ToolTip method), 45
 - `_setPage()` (vi.widgets.list.ViewportListWidget method), 40
 - `_setValue()` (vi.widgets.appnavigation.NavigationSeparator method), 30
 - `_setValue()` (vi.widgets.code.Codemirror method), 31
 - `_showErrorMsg()` (vi.widgets.TreeWidget method), 57
 - `_showErrorMsg()` (vi.widgets.tree.TreeWidget method), 47
 - `_updateToStore()` (vi.utils.indexeddb method), 70
 - `_writeToStore()` (vi.utils.indexeddb method), 70
- ## A
- `a` (in module vi), 73
 - `Accordion` (class in vi.widgets), 50
 - `Accordion` (class in vi.widgets.accordion), 29
 - `AccordionSegment` (class in vi.widgets.accordion), 28
 - `ActionBar` (class in vi.framework.components.actionbar), 19
 - `actionDelegateSelector` (in module vi.priorityqueue), 67
 - `activate()` (vi.widgets.accordion.AccordionSegment method), 28
 - `activateCurrentSelection()` (vi.framework.components.datatable.DataTable method), 22
 - `activateSelection()` (vi.widgets.DataTable method), 54
 - `activateSelection()` (vi.widgets.list.ListWidget method), 39
 - `activateSelection()` (vi.widgets.ListWidget method), 52
 - `activateSelection()` (vi.widgets.table.DataTable method), 44
 - `activateSelection()` (vi.widgets.tree.TreeBrowserWidget method), 49
 - `activateSelection()` (vi.widgets.tree.TreeWidget method), 47
 - `activateSelection()` (vi.widgets.TreeBrowserWidget method), 58
 - `activateSelection()` (vi.widgets.TreeWidget method), 57
 - `add()` (vi.framework.components.datatable.DataTable method), 21
 - `add()` (vi.framework.components.datatable.ViewportDataTable method), 21

- method), 22
- add() (vi.widgets.DataTable method), 54
- add() (vi.widgets.table.DataTable method), 43
- AddAction (class in vi.actions.hierarchy), 8
- AddAction (class in vi.actions.list), 10
- addChildPane() (vi.pane.Pane method), 66
- AddLeafAction (class in vi.actions.file), 7
- AddLeafAction (class in vi.actions.tree), 17
- addNavigationBlock()
(vi.widgets.appnavigation.AppNavigation
method), 30
- 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), 17
- AddNodeOnlyAction (class in vi.actions.tree), 17
- addSegment() (vi.widgets.Accordion method), 50
- addSegment() (vi.widgets.accordion.Accordion
method), 29
- addSelectedRow() (vi.framework.components.datatable.SelectTable
method), 20
- addSelectedRow() (vi.widgets.table.SelectTable
method), 42
- addSeperator() (vi.widgets.appnavigation.Navigationblock
method), 30
- addToLog() (vi.widgets.code.PythonCode method), 31
- addWidget() (vi.pane.Pane method), 66
- addWidget() (vi.widgets.accordion.AccordionSegment
method), 29
- admin() (vi.Application method), 73
- AdminScreen (class in vi), 72
- AdminScreen (class in vi.admin), 60
- allDeletedSuccess()
(vi.actions.hierarchy.DeleteAction method), 9
- allDeletedSuccess() (vi.actions.list.DeleteAction
method), 11
- allDeletedSuccess() (vi.actions.tree.DeleteAction
method), 18
- appendNavList() (vi.admin.AdminScreen method), 60
- appendNavList() (vi.AdminScreen method), 72
- appendSubChild() (vi.widgets.appnavigation.NavigationElement
method), 30
- appHierarchy (vi.widgets.edit.EditWidget attribute), 33
- appHierarchy (vi.widgets.EditWidget attribute), 52
- Application (class in vi), 73
- appList (vi.widgets.edit.EditWidget attribute), 33
- appList (vi.widgets.EditWidget attribute), 52
- apply() (vi.serversideaction.ServerSideActionWdg
method), 68
- applyFilter() (vi.sidebarwidgets.filterselector.FilterSelector
method), 23
- AppNavigation (class in vi.widgets.appnavigation), 30
- appSingleton (vi.widgets.edit.EditWidget attribute), 33
- appSingleton (vi.widgets.EditWidget attribute), 52
- appTree (vi.widgets.edit.EditWidget attribute), 33
- appTree (vi.widgets.EditWidget attribute), 52
- ArrowAction() (vi.widgets.appnavigation.NavigationElement
method), 30
- ## B
- BaseLoginHandler (class in vi.login), 64
- BreadcrumbNodeWidget (class in vi.widgets.tree), 48
- BrowserLeafWidget (class in vi.widgets.tree), 48
- BrowserNodeWidget (class in vi.widgets.tree), 48
- buildAccordion() (vi.widgets.Accordion method), 50
- buildAccordion() (vi.widgets.accordion.Accordion
method), 29
- buildDataTable() (vi.log.logWidget method), 62
- buildSeperator() (vi.widgets.appnavigation.NavigationSeperator
method), 30
- ## C
- CancelClose (class in vi.actions.edit), 6
- canHandle() (vi.log.LogButton static method), 63
- canHandle() (vi.login.GoogleAccountLoginHandler
static method), 64
- canHandle() (vi.login.OAuthAccountLoginHandler
static method), 65
- canHandle() (vi.login.UserPasswordLoginHandler
static method), 64
- canHandle() (vi.views.hierarchy.hierarchyHandler
static method), 25
- canHandle() (vi.views.list.listHandler static method), 26
- canHandle() (vi.views.singleton.singletonHandler static
method), 27
- canHandle() (vi.views.tree.treeHandler static method),
28
- canHandle() (vi.widgets.file.FileWidget static method),
36
- canHandle() (vi.widgets.FileWidget static method), 59
- canHandle() (vi.widgets.hierarchy.HierarchyWidget
static method), 37
- canHandle() (vi.widgets.HierarchyWidget static
method), 59
- canHandle() (vi.widgets.list.ListWidget static method),
39
- canHandle() (vi.widgets.list.ViewportListWidget static
method), 40
- canHandle() (vi.widgets.ListWidget static method), 52
- canHandle() (vi.widgets.topbar.Logout static method),
46
- canHandle() (vi.widgets.topbar.Scripiter static method),
46
- canHandle() (vi.widgets.topbar.Tasks static method), 46

- canHandle() (vi.widgets.topbar.UserState static method), 46
 canHandle() (vi.widgets.tree.TreeBrowserWidget static method), 49
 canHandle() (vi.widgets.tree.TreeWidget static method), 48
 canHandle() (vi.widgets.TreeBrowserWidget static method), 58
 canHandle() (vi.widgets.TreeWidget static method), 58
 checkErrors() (in module vi.widgets.internaedit), 37
 checkForSuspendResume() (vi.widgets.UserLogoutMsg method), 55
 checkForSuspendResume() (vi.widgets.userlogoutmsg.UserLogoutMsg method), 49
 checkInitialHash() (vi.admin.AdminScreen method), 60
 checkInitialHash() (vi.AdminScreen method), 72
 checkInterval (vi.widgets.UserLogoutMsg attribute), 55
 checkInterval (vi.widgets.userlogoutmsg.UserLogoutMsg attribute), 49
 checkVisibility() (vi.widgets.accordion.AccordionSegment method), 28
 cleanLog() (vi.log.LogButton method), 62
 cleanLogAction() (vi.log.LogButton method), 62
 clear() (vi.framework.components.datatable.DataTable method), 22
 clear() (vi.framework.components.datatable.SelectTable method), 21
 clear() (vi.framework.components.datatable.ViewportDataTable method), 22
 clear() (vi.widgets.Accordion method), 50
 clear() (vi.widgets.accordion.Accordion method), 29
 clear() (vi.widgets.DataTable method), 54
 clear() (vi.widgets.edit.EditWidget method), 34
 clear() (vi.widgets.EditWidget method), 53
 clear() (vi.widgets.table.DataTable method), 43
 clear() (vi.widgets.table.SelectTable method), 42
 clearSelection() (vi.widgets.tree.TreeWidget method), 47
 clearSelection() (vi.widgets.TreeWidget method), 57
 CloneAction (class in vi.actions.hierarchy), 8
 CloneAction (class in vi.actions.list), 11
 cloneComplete() (vi.widgets.edit.EditWidget method), 34
 cloneComplete() (vi.widgets.EditWidget method), 53
 close() (vi.widgets.SideBar method), 55
 close() (vi.widgets.sidebar.SideBar method), 41
 CloseAction (class in vi.actions.list), 12
 closeMessage() (vi.widgets.file.MultiUploader method), 35
 closeOrContinue() (vi.widgets.edit.EditWidget method), 34
 closeOrContinue() (vi.widgets.EditWidget method), 53
 CodeHelpPopup (class in vi.widgets.code), 31
 Codemirror (class in vi.widgets.code), 31
 CodePopup (class in vi.widgets.code), 31
 collapse() (vi.pane.GroupPane method), 66
 collapse() (vi.pane.Pane method), 66
 CompoundFilter (class in vi.sidebarwidgets.filterselector), 23
 conf (in module vi), 73
 conf (in module vi.config), 61
 connect() (vi.utils.indexeddb method), 70
 connect() (vi.utils.indexeddbConnector method), 69
 containsWidget() (vi.pane.Pane method), 66
 ContextAction (class in vi.actions.context), 4
 createDir() (vi.actions.file.AddNodeAction method), 7
 CreateRecurrentAction (class in vi.actions.list), 15
 cssname (vi.login.GoogleAccountLoginHandler attribute), 64
 cssname (vi.login.OAuthAccountLoginHandler attribute), 65
 cssname (vi.login.UserPasswordLoginHandler attribute), 64
 csvWriter (class in webworker_scripts), 74
- ## D
- d (in module vi), 73
 DataTable (class in vi.framework.components.datatable), 21
 DataTable (class in vi.widgets), 53
 DataTable (class in vi.widgets.table), 42
 db_blocked() (vi.utils.indexeddbConnector method), 70
 db_error() (vi.utils.indexeddbConnector method), 69
 db_onupgradeneeded() (vi.utils.indexeddbConnector method), 70
 db_success() (vi.utils.indexeddb method), 70
 db_success() (vi.utils.indexeddbConnector method), 70
 db_version() (vi.utils.indexeddbConnector method), 70
 dbAction() (vi.utils.indexeddb method), 70
 dbqueue (vi.utils.indexeddb attribute), 70
 dbResult (vi.utils.indexeddbConnector attribute), 69
 dbTransaction (vi.utils.indexeddbConnector attribute), 69
 deactivate() (vi.widgets.accordion.AccordionSegment method), 28
 DeferredLoadChildren() (vi.pane.GroupPane method), 66
 DeleteAction (class in vi.actions.hierarchy), 9
 DeleteAction (class in vi.actions.list), 11
 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), 74

- disable() (*vi.login.BaseLoginHandler* method), 64
 disableViUnloadingWarning()
 (*vi.actions.file.DownloadAction* method), 8
 doApply() (*vi.actions.list.SelectFieldsPopup* method), 12
 doCancel() (*vi.actions.list.SelectFieldsPopup* method), 12
 doCloneHierarchy() (*vi.widgets.edit.EditWidget* method), 34
 doCloneHierarchy() (*vi.widgets.EditWidget* method), 53
 doDelete() (*vi.actions.hierarchy.DeleteAction* method), 9
 doDelete() (*vi.actions.list.DeleteAction* method), 11
 doDelete() (*vi.actions.tree.DeleteAction* method), 18
 doDownload() (*vi.actions.file.DownloadAction* method), 8
 doInvertSelection()
 (*vi.actions.list.SelectFieldsPopup* method), 13
 doLoginFailure() (*vi.login.UserPasswordLoginHandler* method), 64
 doLoginSuccess() (*vi.login.UserPasswordLoginHandler* method), 64
 doMarkPayed() (*vi.actions.list_order.ShopMarkAction* method), 16
 doSave() (*vi.widgets.edit.EditWidget* method), 34
 doSave() (*vi.widgets.EditWidget* method), 53
 doSave() (*vi.widgets.InternalEdit* method), 56
 doSave() (*vi.widgets.internaedit.InternalEdit* method), 37
 doSearch() (*vi.widgets.Search* method), 55
 doSearch() (*vi.widgets.search.Search* method), 40
 doSelectAll() (*vi.actions.list.SelectFieldsPopup* method), 13
 doSetFields() (*vi.actions.list.SelectFieldsPopup* method), 12
 doShowLogin() (*vi.login.LoginScreen* method), 65
 doShowLogin() (*vi.LoginScreen* method), 71
 doKeySuccess() (*vi.login.OAuthAccountLoginHandler* method), 65
 doSkipLogin() (*vi.login.LoginScreen* method), 65
 doSkipLogin() (*vi.LoginScreen* method), 71
 doUnselectAll() (*vi.actions.list.SelectFieldsPopup* method), 13
 doVerifyFailure() (*vi.login.UserPasswordLoginHandler* method), 64
 doVerifySuccess() (*vi.login.UserPasswordLoginHandler* method), 64
 download() (*vi.widgets.file.FilePreviewImage* method), 35
 download() (*webworker_scripts.csvWriter* method), 74
 DownloadAction (class in *vi.actions.file*), 7
 dropTableContent() (*vi.framework.components.datatable.SelectTable* method), 21
- ## 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*), 17
 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*), 52
 EditWidget (class in *vi.widgets.edit*), 33
 enable() (*vi.login.BaseLoginHandler* method), 64
 enable() (*vi.login.UserPasswordLoginHandler* method), 64
 enableViUnloadingWarning()
 (*vi.actions.file.DownloadAction* method), 8
 EntryIcon() (*vi.widgets.file.FileLeafWidget* method), 35
 error() (*webworker_scripts.weblog* static method), 75
 execCall() (*vi.admin.AdminScreen* method), 60
 execCall() (*vi.AdminScreen* method), 72
 ExecuteSingleton (class in *vi.actions.edit*), 5
 expand() (*vi.pane.GroupPane* method), 66
 expand() (*vi.pane.Pane* method), 66
 ExportCsv (class in *vi.widgets*), 56
 ExportCsv (class in *vi.widgets.csvexport*), 32
 ExportCsvAction (class in *vi.actions.list*), 15
 ExportCsvStarter (class in *vi.widgets*), 56
 ExportCsvStarter (class in *vi.widgets.csvexport*), 32
 exportToFile() (*vi.widgets.csvexport.ExportCsv* method), 32
 exportToFile() (*vi.widgets.ExportCsv* method), 56
 extend() (*vi.framework.components.datatable.DataTable* method), 21
 extend() (*vi.framework.components.datatable.ViewportDataTable* method), 22
 extend() (*vi.widgets.DataTable* method), 54
 extend() (*vi.widgets.table.DataTable* method), 43
 extendedSearchWidgetSelector (in module *vi.priorityqueue*), 67
 extendSelection() (*vi.widgets.tree.TreeWidget* method), 47
 extendSelection() (*vi.widgets.TreeWidget* method), 57
- ## F
- fetchFailed() (*vi.serversideaction.ServerSideActionWdg* method), 69
 fetchNext() (*vi.serversideaction.ServerSideActionWdg* method), 68
 fetchSucceeded() (*vi.serversideaction.ServerSideActionWdg* method), 69

- FileImagePopup (class in *vi.widgets.file*), 34
 FileLeafWidget (class in *vi.widgets.file*), 35
 FileNodeWidget (class in *vi.widgets.file*), 35
 FilePreviewImage (class in *vi.widgets.file*), 34
 FileSelectUploader (class in *vi.actions.file*), 6
 FileWidget (class in *vi.widgets*), 59
 FileWidget (class in *vi.widgets.file*), 36
 FilterSelector (class in *vi.sidebarwidgets.filterselector*), 23
 findText() (*vi.actions.list.PageFindAction* method), 14
 focus() (*vi.pane.Pane* method), 66
 focus() (*vi.sidebarwidgets.filterselector.CompoundFilter* method), 23
 focus() (*vi.widgets.Search* method), 55
 focus() (*vi.widgets.search.Search* method), 40
 focusLaterIdiot() (*vi.login.UserPasswordLoginHandler* method), 64
 focusRow() (*vi.framework.components.datatable.SelectTable* method), 21
 focusRow() (*vi.widgets.table.SelectTable* method), 42
 formatString() (in module *vi.utils*), 69
- ## G
- genReqStr() (*webworker_scripts.SimpleNetwork* method), 75
 getActions() (*vi.framework.components.actionbar.ActionBar* method), 19
 getActions() (*vi.widgets.hierarchy.HierarchyWidget* method), 36
 getActions() (*vi.widgets.HierarchyWidget* method), 58
 getActions() (*vi.widgets.list.ListWidget* method), 38
 getActions() (*vi.widgets.ListWidget* method), 51
 getActions() (*vi.widgets.tree.TreeWidget* method), 47
 getActions() (*vi.widgets.TreeWidget* method), 57
 getAllActions() (*vi.widgets.list.ListWidget* method), 38
 getAllActions() (*vi.widgets.ListWidget* method), 51
 getChildKey() (*vi.widgets.file.FileWidget* method), 36
 getChildKey() (*vi.widgets.FileWidget* method), 59
 getChildKey() (*vi.widgets.tree.TreeWidget* method), 48
 getChildKey() (*vi.widgets.TreeWidget* method), 58
 getConf() (in module *vi.config*), 61
 getConfigSuccess() (*vi.Application* method), 73
 getCurrentSelection() (*vi.framework.components.datatable.DataTable* method), 22
 getCurrentSelection() (*vi.framework.components.datatable.SelectTable* method), 21
 getCurrentSelection() (*vi.widgets.DataTable* method), 54
 getCurrentSelection() (*vi.widgets.table.DataTable* method), 44
 getCurrentSelection() (*vi.widgets.table.SelectTable* method), 42
 getCurrentUser() (*vi.admin.AdminScreen* method), 60
 getCurrentUser() (*vi.AdminScreen* method), 72
 getCurrentUserFailure() (*vi.admin.AdminScreen* method), 60
 getCurrentUserFailure() (*vi.AdminScreen* method), 72
 getCurrentUserSuccess() (*vi.admin.AdminScreen* method), 60
 getCurrentUserSuccess() (*vi.AdminScreen* method), 72
 getDefaultEntryActions() (*vi.widgets.list.ListWidget* method), 38
 getDefaultEntryActions() (*vi.widgets.ListWidget* method), 51
 getFields() (*vi.widgets.list.ListWidget* method), 39
 getFields() (*vi.widgets.ListWidget* method), 52
 getFilter() (*vi.widgets.list.ListWidget* method), 39
 getFilter() (*vi.widgets.ListWidget* method), 52
 getImagePreview() (in module *vi.utils*), 69
 getIndexByTr() (*vi.framework.components.datatable.SelectTable* method), 20
 getIndexByTr() (*vi.widgets.table.SelectTable* method), 41
 getIndexedDb() (*vi.utils.indexeddb* method), 70
 getListKeys() (*vi.utils.indexeddb* method), 70
 getNavigationPoint() (*vi.widgets.appnavigation.AppNavigation* method), 30
 getPreviousNavigationPoint() (*vi.widgets.appnavigation.AppNavigation* method), 30
 getRowCount() (*vi.framework.components.datatable.DataTable* method), 21
 getRowCount() (*vi.widgets.DataTable* method), 53
 getRowCount() (*vi.widgets.table.DataTable* method), 43
 getSelectedTask() (*vi.widgets.task.TaskSelectWidget* method), 44
 getSelectedTask() (*vi.widgets.TaskSelectWidget* method), 56
 getTrByIndex() (*vi.framework.components.datatable.SelectTable* method), 20
 getTrByIndex() (*vi.widgets.table.SelectTable* method), 41
 getVersionSuccess() (*vi.Application* method), 73
 getWidget() (*vi.widgets.SideBar* method), 55
 getWidget() (*vi.widgets.sidebar.SideBar* method), 41
 GoogleAccountLoginHandler (class in *vi.login*), 64
 GroupPane (class in *vi.pane*), 66
- ## H
- handleFile() (*vi.widgets.file.MultiUploader* method), 35

- HandlerClassSelector (in module *vi.priorityqueue*), 67
- hideListView() (*vi.widgets.hierarchy.HierarchyWidget* method), 36
- hideListView() (*vi.widgets.HierarchyWidget* method), 59
- hideMessage() (*vi.widgets.UserLogoutMsg* method), 55
- hideMessage() (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), 49
- hierarchyHandler (class in *vi.views.hierarchy*), 25
- hierarchyHandlerWidget (class in *vi.views.hierarchy*), 25
- HierarchyWidget (class in *vi.widgets*), 58
- HierarchyWidget (class in *vi.widgets.hierarchy*), 36
- HttpRequest (class in *webworker_scripts*), 75
- I
- idbdata() (*vi.log.LogButton* method), 62
- idbTableName (in module *vi.log*), 62
- indexeddb (class in *vi.utils*), 70
- indexeddbConnector (class in *vi.utils*), 69
- info() (*webworker_scripts.weblog* static method), 74
- initialHashHandler (in module *vi.priorityqueue*), 67
- initializeConfig() (*vi.admin.AdminScreen* method), 60
- initializeConfig() (*vi.AdminScreen* method), 72
- initializeViews() (*vi.admin.AdminScreen* method), 60
- initializeViews() (*vi.AdminScreen* method), 72
- initWidget() (*vi.views.edit.editHandlerWidget* method), 25
- initWidget() (*vi.views.hierarchy.hierarchyHandlerWidget* method), 25
- initWidget() (*vi.views.list.listHandlerWidget* method), 26
- initWidget() (*vi.views.log.logHandlerWidget* method), 26
- initWidget() (*vi.views.notfound.NotFoundWidget* method), 26
- initWidget() (*vi.views.overview.OverviewWidget* method), 27
- initWidget() (*vi.views.singleton.singletonHandlerWidget* method), 27
- initWidget() (*vi.views.tree.treeHandlerWidget* method), 28
- insertElem() (*vi.priorityqueue.StartupQueue* method), 67
- insertText() (*vi.widgets.code.Codemirror* method), 31
- insufficientRights() (*vi.login.LoginScreen* method), 65
- insufficientRights() (*vi.LoginScreen* method), 71
- InternalEdit (class in *vi.widgets*), 56
- InternalEdit (class in *vi.widgets.internaedit*), 37
- InternalPreview (class in *vi.sidebarwidgets.internalpreview*), 24
- InvalidBoneValueException, 61
- invertSelection() (*vi.framework.components.datatable.SelectTable* method), 21
- invertSelection() (*vi.widgets.table.SelectTable* method), 42
- invoke() (*vi.admin.AdminScreen* method), 60
- invoke() (*vi.AdminScreen* method), 72
- invoke() (*vi.login.LoginScreen* method), 65
- invoke() (*vi.LoginScreen* method), 71
- invoke() (*vi.screen.Screen* method), 68
- invoke() (*vi.widgets.topbar.TopBarWidget* method), 45
- invoke() (*vi.widgets.TopBarWidget* method), 51
- invokeTask() (*vi.widgets.task.TaskSelectWidget* method), 44
- invokeTask() (*vi.widgets.TaskSelectWidget* method), 56
- isActive() (*vi.widgets.accordion.AccordionSegment* method), 28
- isFullscreen() (*vi.admin.AdminScreen* method), 61
- isFullscreen() (*vi.AdminScreen* method), 72
- 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), 9
- isSuitableFor() (*vi.actions.hierarchy.ReloadAction*

- static method*), 9
- `isSuitableFor()` (*vi.actions.list.AddAction static method*), 10
- `isSuitableFor()` (*vi.actions.list.CloneAction static method*), 11
- `isSuitableFor()` (*vi.actions.list.CloseAction static method*), 12
- `isSuitableFor()` (*vi.actions.list.CreateRecurrentAction static method*), 15
- `isSuitableFor()` (*vi.actions.list.DeleteAction static method*), 11
- `isSuitableFor()` (*vi.actions.list.EditAction static method*), 11
- `isSuitableFor()` (*vi.actions.list.ExportCsvAction static method*), 15
- `isSuitableFor()` (*vi.actions.list.ListPreviewAction static method*), 12
- `isSuitableFor()` (*vi.actions.list.ListPreviewInlineAction static method*), 12
- `isSuitableFor()` (*vi.actions.list.ListSelectFilterAction static method*), 15
- `isSuitableFor()` (*vi.actions.list.LoadAllAction static method*), 14
- `isSuitableFor()` (*vi.actions.list.LoadNextBatchAction static method*), 14
- `isSuitableFor()` (*vi.actions.list.PageFindAction static method*), 14
- `isSuitableFor()` (*vi.actions.list.ReloadAction static method*), 13
- `isSuitableFor()` (*vi.actions.list.SelectAction static method*), 12
- `isSuitableFor()` (*vi.actions.list.SelectAllAction static method*), 15
- `isSuitableFor()` (*vi.actions.list.SelectFieldsAction static method*), 13
- `isSuitableFor()` (*vi.actions.list.SelectInvertAction static method*), 15
- `isSuitableFor()` (*vi.actions.list.SetPageRowAmountAction static method*), 14
- `isSuitableFor()` (*vi.actions.list.TableItems static method*), 13
- `isSuitableFor()` (*vi.actions.list.TableNextPage static method*), 13
- `isSuitableFor()` (*vi.actions.list.TablePrevPage static method*), 13
- `isSuitableFor()` (*vi.actions.list.UnSelectAllAction static method*), 15
- `isSuitableFor()` (*vi.actions.list_order.ShopMarkCanceledAction static method*), 16
- `isSuitableFor()` (*vi.actions.list_order.ShopMarkPayedAction static method*), 16
- `isSuitableFor()` (*vi.actions.list_order.ShopMarkSentAction static method*), 16
- `isSuitableFor()` (*vi.actions.tree.AddLeafAction static method*), 17
- `isSuitableFor()` (*vi.actions.tree.AddNodeAction static method*), 17
- `isSuitableFor()` (*vi.actions.tree.AddNodeOnlyAction static method*), 17
- `isSuitableFor()` (*vi.actions.tree.DeleteAction static method*), 18
- `isSuitableFor()` (*vi.actions.tree.EditAction static method*), 18
- `isSuitableFor()` (*vi.actions.tree.ReloadAction static method*), 18
- `isSuitableFor()` (*vi.actions.tree.SelectRootNode static method*), 18
- `itemForKey()` (*vi.widgets.tree.TreeWidget method*), 48
- `itemForKey()` (*vi.widgets.TreeWidget method*), 57
- ## K
- `killClick()` (*vi.widgets.code.Scripter method*), 32
- ## L
- `leafWidget` (*vi.widgets.file.FileWidget attribute*), 36
- `leafWidget` (*vi.widgets.FileWidget attribute*), 59
- `leafWidget` (*vi.widgets.hierarchy.HierarchyWidget attribute*), 36
- `leafWidget` (*vi.widgets.HierarchyWidget attribute*), 58
- `leafWidget` (*vi.widgets.tree.TreeBrowserWidget attribute*), 49
- `leafWidget` (*vi.widgets.tree.TreeWidget attribute*), 47
- `leafWidget` (*vi.widgets.TreeBrowserWidget attribute*), 58
- `leafWidget` (*vi.widgets.TreeWidget attribute*), 57
- `listHandler` (*class in vi.views.list*), 26
- `listHandlerWidget` (*class in vi.views.list*), 26
- `ListPreviewAction` (*class in vi.actions.list*), 12
- `ListPreviewInlineAction` (*class in vi.actions.list*), 12
- `ListSelectFilterAction` (*class in vi.actions.list*), 15
- `ListViewAction` (*class in vi.actions.hierarchy*), 9
- `ListWidget` (*class in vi.widgets*), 51
- `ListWidget` (*class in vi.widgets.list*), 38
- `lngDe` (*in module vi.translations*), 24
- `lngDe` (*in module vi.translations.de*), 24
- `lngEn` (*in module vi.translations*), 24
- `lngEn` (*in module vi.translations.en*), 24
- `LoadAllAction` (*class in vi.actions.list*), 14
- `loadAllRows()` (*vi.actions.list.LoadAllAction method*), 14
- `loadChildren()` (*vi.pane.GroupPane method*), 66
- `LoadNextBatchAction` (*class in vi.actions.list*), 14
- `loadnextPages()` (*vi.actions.list.LoadNextBatchAction method*), 14
- `loadNode()` (*vi.widgets.tree.TreeWidget method*), 48
- `loadNode()` (*vi.widgets.TreeWidget method*), 58
- `lock()` (*vi.login.BaseLoginHandler method*), 64
- `lock()` (*vi.pane.Pane method*), 66

`lock()` (*vi.screen.Screen* method), 68
`Log` (class in *vi.log*), 63
`log` (in module *webworker_scripts*), 75
`log()` (*vi.admin.AdminScreen* method), 60
`log()` (*vi.AdminScreen* method), 72
`log()` (*vi.log.Log* method), 63
`log()` (*vi.log.LogButton* method), 63
`logA` (class in *vi.log*), 62
`LogButton` (class in *vi.log*), 62
`logEntry` (class in *vi.log*), 62
`logHandler` (class in *vi.views.log*), 26
`logHandlerWidget` (class in *vi.views.log*), 26
`login()` (*vi.Application* method), 73
`login()` (*vi.login.BaseLoginHandler* method), 64
`loginHandlerSelector` (in module *vi.priorityqueue*), 67
`LoginInputField` (class in *vi.login*), 63
`LoginScreen` (class in *vi*), 71
`LoginScreen` (class in *vi.login*), 65
`Logout` (class in *vi.widgets.topbar*), 46
`logout()` (*vi.Application* method), 73
`logout()` (*vi.widgets.topbar.Logout* method), 46
`logWidget` (class in *vi.log*), 62

M

`mergeDict()` (in module *vi.utils*), 70
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, 16
 vi.actions.tree, 17
 vi.admin, 59
 vi.config, 61
 vi.exception, 61
 vi.framework, 19
 vi.framework.components, 19
 vi.framework.components.actionbar, 19
 vi.framework.components.datatable, 19
 vi.log, 62
 vi.login, 63
 vi.pane, 65
 vi.priorityqueue, 67
 vi.screen, 68
 vi.serversideaction, 68
 vi.sidebarwidgets, 23
 vi.sidebarwidgets.filterselector, 23
 vi.sidebarwidgets.internalpreview, 24
 vi.translations, 24
 vi.translations.de, 24

vi.translations.en, 24
 vi.utils, 69
 vi.views, 24
 vi.views.edit, 24
 vi.views.hierarchy, 25
 vi.views.list, 25
 vi.views.log, 26
 vi.views.notfound, 26
 vi.views.overview, 27
 vi.views.singleton, 27
 vi.views.tree, 28
 vi.widgets, 28
 vi.widgets.accordion, 28
 vi.widgets.appnavigation, 29
 vi.widgets.code, 31
 vi.widgets.csvexport, 32
 vi.widgets.edit, 33
 vi.widgets.file, 34
 vi.widgets.hierarchy, 36
 vi.widgets.internaledit, 37
 vi.widgets.list, 38
 vi.widgets.search, 40
 vi.widgets.sidebar, 40
 vi.widgets.table, 41
 vi.widgets.task, 44
 vi.widgets.tooltip, 45
 vi.widgets.topbar, 45
 vi.widgets.tree, 46
 vi.widgets.userlogoutmsg, 49
 webworker_scripts, 73
`msgOverlay()` (*vi.log.LogButton* method), 63
`MultiUploader` (class in *vi.widgets.file*), 35

N

`navigationAction()` (*vi.widgets.appnavigation.NavigationElement* method), 30
`Navigationblock` (class in *vi.widgets.appnavigation*), 30
`NavigationElement` (class in *vi.widgets.appnavigation*), 29
`NavigationSeperator` (class in *vi.widgets.appnavigation*), 30
`next()` (*vi.priorityqueue.StartupQueue* method), 67
`next()` (*webworker_scripts.requestList* method), 74
`nextChunk()` (*vi.widgets.csvexport.ExportCsv* method), 32
`nextChunk()` (*vi.widgets.ExportCsv* method), 56
`nextChunkComplete()` (*vi.widgets.csvexport.ExportCsv* method), 32
`nextChunkComplete()` (*vi.widgets.ExportCsv* method), 56
`nextChunkFailure()` (*vi.widgets.csvexport.ExportCsv* method), 32

- nextChunkFailure() (*vi.widgets.ExportCsv method*), 56
- nodeWidget (*vi.widgets.file.FileWidget attribute*), 36
- nodeWidget (*vi.widgets.FileWidget attribute*), 59
- nodeWidget (*vi.widgets.tree.TreeBrowserWidget attribute*), 49
- nodeWidget (*vi.widgets.tree.TreeWidget attribute*), 47
- nodeWidget (*vi.widgets.TreeBrowserWidget attribute*), 58
- nodeWidget (*vi.widgets.TreeWidget attribute*), 56
- NotFound (*class in vi.views.notfound*), 26
- NotFoundWidget (*class in vi.views.notfound*), 26
- ## O
- OAuthAccountLoginHandler (*class in vi.login*), 65
- onActiveNavigationChanged() (*vi.widgets.appnavigation.NavigationElement method*), 30
- 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.list.CloneAction method*), 11
- onAttach() (*vi.actions.list.DeleteAction method*), 11
- onAttach() (*vi.actions.list.EditAction method*), 11
- onAttach() (*vi.actions.list.ListPreviewAction method*), 12
- onAttach() (*vi.actions.list.ListPreviewInlineAction method*), 12
- onAttach() (*vi.actions.list.ListSelectFilterAction method*), 15
- onAttach() (*vi.actions.list.SelectAllAction method*), 15
- onAttach() (*vi.actions.list.SelectFieldsAction method*), 13
- onAttach() (*vi.actions.list.SelectInvertAction method*), 15
- onAttach() (*vi.actions.list.UnSelectAllAction method*), 15
- onAttach() (*vi.actions.list_order.ShopMarkAction method*), 16
- onAttach() (*vi.actions.tree.DeleteAction method*), 18
- onAttach() (*vi.actions.tree.EditAction method*), 17
- onAttach() (*vi.actions.tree.SelectRootNode method*), 18
- onAttach() (*vi.framework.components.datatable.SelectTable method*), 20
- onAttach() (*vi.serversideaction.ServerSideActionWdg method*), 68
- onAttach() (*vi.sidebarwidgets.filterselector.FilterSelector method*), 23
- onAttach() (*vi.widgets.code.Codemirror method*), 31
- onAttach() (*vi.widgets.edit.EditWidget method*), 33
- onAttach() (*vi.widgets.EditWidget method*), 52
- onAttach() (*vi.widgets.list.ListWidget method*), 39
- onAttach() (*vi.widgets.ListWidget method*), 52
- onAttach() (*vi.widgets.SideBar method*), 55
- onAttach() (*vi.widgets.sidebar.SideBar method*), 41
- onAttach() (*vi.widgets.table.SelectTable method*), 41
- onAttach() (*vi.widgets.tree.TreeWidget method*), 48
- onAttach() (*vi.widgets.TreeWidget method*), 57
- onBoneChange() (*vi.widgets.edit.EditWidget method*), 33
- onBoneChange() (*vi.widgets.EditWidget method*), 52
- onBtnCloseReleased() (*vi.pane.Pane method*), 66
- onChange() (*vi.actions.file.FileSelectUploader method*), 6
- onChange() (*vi.actions.list.ListPreviewAction method*), 12
- onChange() (*vi.actions.list.LoadNextBatchAction method*), 14
- onChange() (*vi.actions.list.SetPageRowAmountAction method*), 14
- onChange() (*vi.actions.tree.SelectRootNode method*), 18
- onChange() (*vi.framework.components.datatable.SelectTable method*), 20
- onChange() (*vi.widgets.edit.EditWidget method*), 33
- onChange() (*vi.widgets.EditWidget method*), 52
- onChange() (*vi.widgets.InternalEdit method*), 56
- onChange() (*vi.widgets.internaedit.InternalEdit method*), 38
- onChange() (*vi.widgets.table.SelectTable method*), 41
- onChange() (*vi.widgets.task.TaskSelectWidget method*), 44
- onChange() (*vi.widgets.TaskSelectWidget method*), 56
- 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*), 9
- `onClick()` (*vi.actions.hierarchy.ReloadAction method*), 9
- `onClick()` (*vi.actions.list.AddAction method*), 10
- `onClick()` (*vi.actions.list.CloneAction method*), 11
- `onClick()` (*vi.actions.list.CloseAction method*), 12
- `onClick()` (*vi.actions.list.CreateRecurrentAction method*), 15
- `onClick()` (*vi.actions.list.DeleteAction method*), 11
- `onClick()` (*vi.actions.list.EditAction method*), 11
- `onClick()` (*vi.actions.list.ExportCsvAction method*), 15
- `onClick()` (*vi.actions.list.ListPreviewAction method*), 12
- `onClick()` (*vi.actions.list.ListPreviewInlineAction method*), 12
- `onClick()` (*vi.actions.list.ListSelectFilterAction method*), 15
- `onClick()` (*vi.actions.list.LoadAllAction method*), 14
- `onClick()` (*vi.actions.list.LoadNextBatchAction method*), 14
- `onClick()` (*vi.actions.list.PageFindAction method*), 14
- `onClick()` (*vi.actions.list.ReloadAction method*), 13
- `onClick()` (*vi.actions.list.SelectAction method*), 12
- `onClick()` (*vi.actions.list.SelectAllAction method*), 15
- `onClick()` (*vi.actions.list.SelectFieldsAction method*), 13
- `onClick()` (*vi.actions.list.SelectInvertAction method*), 15
- `onClick()` (*vi.actions.list.SetPageRowAmountAction method*), 14
- `onClick()` (*vi.actions.list.TableNextPage method*), 13
- `onClick()` (*vi.actions.list.TablePrevPage method*), 13
- `onClick()` (*vi.actions.list.UnSelectAllAction method*), 15
- `onClick()` (*vi.actions.list_order.ShopMarkAction method*), 16
- `onClick()` (*vi.actions.tree.AddLeafAction method*), 17
- `onClick()` (*vi.actions.tree.AddNodeAction method*), 17
- `onClick()` (*vi.actions.tree.AddNodeOnlyAction method*), 17
- `onClick()` (*vi.actions.tree.DeleteAction method*), 18
- `onClick()` (*vi.actions.tree.EditAction method*), 18
- `onClick()` (*vi.actions.tree.ReloadAction method*), 18
- `onClick()` (*vi.admin.AdminScreen method*), 60
- `onClick()` (*vi.AdminScreen method*), 72
- `onClick()` (*vi.log.logA method*), 62
- `onClick()` (*vi.log.LogButton method*), 62
- `onClick()` (*vi.login.BaseLoginHandler method*), 64
- `onClick()` (*vi.pane.GroupPane method*), 66
- `onClick()` (*vi.pane.Pane method*), 66
- `onClick()` (*vi.serversideaction.ServerSideActionWdg method*), 68
- `onClick()` (*vi.sidebarwidgets.filterselector.FilterSelector method*), 23
- `onClick()` (*vi.widgets.accordion.AccordionSegment method*), 28
- `onClick()` (*vi.widgets.file.FileImagePopup method*), 34
- `onClick()` (*vi.widgets.file.FilePreviewImage method*), 35
- `onClick()` (*vi.widgets.list.ListWidget method*), 38
- `onClick()` (*vi.widgets.ListWidget method*), 51
- `onClick()` (*vi.widgets.ToolTip method*), 53
- `onClick()` (*vi.widgets.tooltip.ToolTip method*), 45
- `onClick()` (*vi.widgets.topbar.Logout method*), 46
- `onClick()` (*vi.widgets.topbar.Scripiter method*), 46
- `onClick()` (*vi.widgets.topbar.Tasks method*), 46
- `onClick()` (*vi.widgets.topbar.TopBarWidget method*), 45
- `onClick()` (*vi.widgets.topbar.UserState method*), 46
- `onClick()` (*vi.widgets.TopBarWidget method*), 51
- `onCompletion()` (*vi.widgets.list.ListWidget method*), 39
- `onCompletion()` (*vi.widgets.ListWidget method*), 52
- `onCompletion()` (*webworker_scripts.SimpleNetwork method*), 75
- `onCopyKey()` (*vi.sidebarwidgets.internalpreview.InternalPreview method*), 24
- `onCurrentUserAvailable()` (*vi.widgets.topbar.Scripiter method*), 46
- `onCurrentUserAvailable()` (*vi.widgets.topbar.Tasks method*), 46
- `onCurrentUserAvailable()` (*vi.widgets.topbar.UserState method*), 45
- `onCursorMoved()` (*vi.framework.components.datatable.DataTable method*), 21
- `onCursorMoved()` (*vi.widgets.DataTable method*), 53
- `onCursorMoved()` (*vi.widgets.table.DataTable method*), 43
- `onDataChanged()` (*vi.widgets.list.ListWidget method*), 39
- `onDataChanged()` (*vi.widgets.ListWidget method*), 52
- `onDataChanged()` (*vi.widgets.tree.TreeWidget method*), 47
- `onDataChanged()` (*vi.widgets.TreeWidget method*), 57
- `onDbClick()` (*vi.framework.components.datatable.SelectTable method*), 20
- `onDbClick()` (*vi.widgets.table.SelectTable method*), 42
- `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.list.CloneAction method*), 11
- `onDetach()` (*vi.actions.list.DeleteAction method*), 11

- onDetach() (vi.actions.list.EditAction method), 11
- onDetach() (vi.actions.list.ListPreviewAction method), 12
- onDetach() (vi.actions.list.ListPreviewInlineAction method), 12
- onDetach() (vi.actions.list.SelectAllAction method), 15
- onDetach() (vi.actions.list.SelectFieldsAction method), 13
- onDetach() (vi.actions.list.SelectInvertAction method), 15
- onDetach() (vi.actions.list.UnSelectAllAction method), 15
- onDetach() (vi.actions.list_order.ShopMarkAction method), 16
- onDetach() (vi.actions.tree.DeleteAction method), 18
- onDetach() (vi.actions.tree.EditAction method), 17
- onDetach() (vi.actions.tree.SelectRootNode method), 18
- onDetach() (vi.pane.Pane method), 66
- onDetach() (vi.serversideaction.ServerSideActionWdg method), 68
- onDetach() (vi.sidebarwidgets.filtersselector.FilterSelector method), 23
- onDetach() (vi.widgets.code.Codemirror method), 31
- onDetach() (vi.widgets.edit.EditWidget method), 33
- onDetach() (vi.widgets.EditWidget method), 52
- onDetach() (vi.widgets.list.ListWidget method), 39
- onDetach() (vi.widgets.ListWidget method), 52
- onDetach() (vi.widgets.SideBar method), 55
- onDetach() (vi.widgets.sidebar.SideBar method), 41
- onDetach() (vi.widgets.tree.TreeWidget method), 48
- onDetach() (vi.widgets.TreeWidget method), 57
- onDownloadBtnClick() (vi.widgets.file.FileImagePopup method), 34
- onDragOver() (vi.widgets.tree.TreeWidget method), 48
- onDragOver() (vi.widgets.TreeWidget method), 58
- onDrop() (vi.widgets.file.FileWidget method), 36
- onDrop() (vi.widgets.FileWidget method), 59
- onDrop() (vi.widgets.tree.TreeWidget method), 48
- onDrop() (vi.widgets.TreeWidget method), 58
- onError() (vi.admin.AdminScreen method), 61
- onError() (vi.AdminScreen method), 73
- onError() (webworker_scripts.SimpleNetwork method), 75
- onExportBtnClick() (vi.widgets.csvexport.ExportCsvStarter method), 32
- onExportBtnClick() (vi.widgets.ExportCsvStarter method), 56
- onFailed() (vi.widgets.file.MultiUploader method), 35
- onFailed() (vi.widgets.file.Uploader method), 35
- onFilterChanged() (vi.sidebarwidgets.filtersselector.CompoundFilter method), 23
- onFocus() (vi.pane.GroupPane method), 66
- onGetAuthMethodsFailure() (vi.login.LoginScreen method), 65
- onGetAuthMethodsFailure() (vi.LoginScreen method), 71
- onGetAuthMethodsSuccess() (vi.login.LoginScreen method), 65
- onGetAuthMethodsSuccess() (vi.LoginScreen method), 71
- onHasSubItemsChanged() (vi.widgets.appnavigation.NavigationElement method), 30
- onKeyDown() (vi.framework.components.datatable.SelectTable method), 20
- onKeyDown() (vi.widgets.InternalEdit method), 56
- onKeyDown() (vi.widgets.internaedit.InternalEdit method), 38
- onKeyDown() (vi.widgets.Search method), 55
- onKeyDown() (vi.widgets.search.Search method), 40
- onKeyDown() (vi.widgets.table.SelectTable method), 42
- onKeyDown() (vi.widgets.tree.TreeWidget method), 47
- onKeyDown() (vi.widgets.TreeWidget method), 57
- onKeyPress() (vi.actions.list.PageFindAction method), 14
- onKeyPress() (vi.login.LoginInputField method), 64
- onKeyPress() (vi.login.UserPasswordLoginHandler method), 64
- onKeyUp() (vi.framework.components.datatable.SelectTable method), 20
- onKeyUp() (vi.widgets.table.SelectTable method), 42
- onKeyUp() (vi.widgets.tree.TreeWidget method), 47
- onKeyUp() (vi.widgets.TreeWidget method), 57
- onLoad() (vi.widgets.file.MultiUploader method), 35
- onLoad() (vi.widgets.file.Uploader method), 35
- onLoginClick() (vi.login.GoogleAccountLoginHandler method), 64
- onLoginClick() (vi.login.OAuthAccountLoginHandler method), 65
- onLoginClick() (vi.login.UserPasswordLoginHandler method), 64
- onLogoutSuccess() (vi.login.LoginScreen method), 65
- onLogoutSuccess() (vi.LoginScreen method), 71
- onMkDir() (vi.actions.file.AddNodeAction method), 7
- onMouseDown() (vi.framework.components.datatable.SelectTable method), 20
- onMouseDown() (vi.widgets.table.SelectTable method), 42
- onMouseOut() (vi.framework.components.datatable.SelectTable method), 20
- onMouseOut() (vi.widgets.table.SelectTable method), 42
- onMouseUp() (vi.framework.components.datatable.SelectTable method), 20
- onMouseUp() (vi.widgets.table.SelectTable method), 42
- onNextBatchNeeded() (vi.widgets.list.ListWidget method), 39

<code>onNextBatchNeeded()</code> (<i>vi.widgets.ListWidget</i> method), 51	<code>onSelectionChanged()</code> (<i>vi.actions.hierarchy.CloneAction</i> method), 9
<code>onPathRequestSucceeded()</code> (<i>vi.widgets.tree.TreeBrowserWidget</i> method), 49	<code>onSelectionChanged()</code> (<i>vi.actions.hierarchy.DeleteAction</i> method), 9
<code>onPathRequestSucceeded()</code> (<i>vi.widgets.TreeBrowserWidget</i> method), 58	<code>onSelectionChanged()</code> (<i>vi.actions.hierarchy.EditAction</i> method), 8
<code>onProgress()</code> (<i>vi.widgets.file.Uploader</i> method), 35	<code>onSelectionChanged()</code> (<i>vi.actions.list.CloneAction</i> method), 11
<code>onReadyStateChange()</code> (<i>web-worker_scripts.HTTPRequest</i> method), 75	<code>onSelectionChanged()</code> (<i>vi.actions.list.DeleteAction</i> method), 11
<code>onRequestingFinished()</code> (<i>vi.widgets.list.ListWidget</i> method), 38	<code>onSelectionChanged()</code> (<i>vi.actions.list.EditAction</i> method), 11
<code>onRequestingFinished()</code> (<i>vi.widgets.list.ViewportListWidget</i> method), 40	<code>onSelectionChanged()</code> (<i>vi.actions.list.ListPreviewAction</i> method), 12
<code>onRequestingFinished()</code> (<i>vi.widgets.ListWidget</i> method), 51	<code>onSelectionChanged()</code> (<i>vi.actions.list.ListPreviewInlineAction</i> method), 12
<code>onRootNodeChanged()</code> (<i>vi.actions.tree.SelectRootNode</i> method), 18	<code>onSelectionChanged()</code> (<i>vi.actions.list_order.ShopMarkAction</i> method), 16
<code>onRootNodesAvailable()</code> (<i>vi.actions.tree.SelectRootNode</i> method), 18	<code>onSelectionChanged()</code> (<i>vi.actions.tree.DeleteAction</i> method), 18
<code>onScroll()</code> (<i>vi.actions.list.LoadNextBatchAction</i> method), 14	<code>onSelectionChanged()</code> (<i>vi.actions.tree.EditAction</i> method), 18
<code>onScroll()</code> (<i>vi.widgets.DataTable</i> method), 54	<code>onSelectionChanged()</code> (<i>vi.framework.components.datatable.DataTable</i> method), 22
<code>onScroll()</code> (<i>vi.widgets.table.DataTable</i> method), 43	<code>onSelectionChanged()</code> (<i>vi.serversideaction.ServerSideActionWdg</i> method), 68
<code>onSelectionActivated()</code> (<i>vi.actions.file.EditAction</i> method), 7	<code>onSelectionChanged()</code> (<i>vi.widgets.DataTable</i> method), 54
<code>onSelectionActivated()</code> (<i>vi.actions.hierarchy.EditAction</i> method), 8	<code>onSelectionChanged()</code> (<i>vi.widgets.hierarchy.HierarchyWidget</i> method), 37
<code>onSelectionActivated()</code> (<i>vi.actions.list.EditAction</i> method), 11	<code>onSelectionChanged()</code> (<i>vi.widgets.HierarchyWidget</i> method), 59
<code>onSelectionActivated()</code> (<i>vi.actions.tree.EditAction</i> method), 17	<code>onSelectionChanged()</code> (<i>vi.widgets.table.DataTable</i> method), 43
<code>onSelectionActivated()</code> (<i>vi.framework.components.datatable.DataTable</i> method), 22	<code>onSendClick()</code> (<i>vi.login.UserPasswordLoginHandler</i> method), 64
<code>onSelectionActivated()</code> (<i>vi.widgets.DataTable</i> method), 54	<code>onSetDefaultRootNode()</code> (<i>vi.widgets.tree.TreeWidget</i> method), 48
<code>onSelectionActivated()</code> (<i>vi.widgets.list.ListWidget</i> method), 39	<code>onSetDefaultRootNode()</code> (<i>vi.widgets.TreeWidget</i> method), 57
<code>onSelectionActivated()</code> (<i>vi.widgets.ListWidget</i> method), 52	<code>onStartSearch()</code> (<i>vi.sidebarwidgets.filtersselector.FilterSelector</i> method), 23
<code>onSelectionActivated()</code> (<i>vi.widgets.table.DataTable</i> method), 43	<code>onStartSearch()</code> (<i>vi.widgets.file.FileWidget</i> method), 36
<code>onSelectionChanged()</code> (<i>vi.actions.context.ContextAction</i> method), 5	<code>onStartSearch()</code> (<i>vi.widgets.FileWidget</i> method), 59
<code>onSelectionChanged()</code> (<i>vi.actions.file.DownloadAction</i> method), 7	<code>onSuccess()</code> (<i>vi.widgets.file.MultiUploader</i> method), 35
<code>onSelectionChanged()</code> (<i>vi.actions.file.EditAction</i> method), 7	

- onSuccess() (vi.widgets.file.Uploader method), 35
 onTableChanged() (vi.actions.list.SelectAllAction method), 15
 onTableChanged() (vi.actions.list.SelectFieldsAction method), 13
 onTableChanged() (vi.actions.list.SelectInvertAction method), 15
 onTableChanged() (vi.actions.list.TableItems method), 13
 onTableChanged() (vi.actions.list.UnSelectAllAction method), 15
 onTableChanged() (vi.framework.components.datatable.DataTable method), 22
 onTableChanged() (vi.widgets.DataTable method), 54
 onTableChanged() (vi.widgets.table.DataTable method), 43
 onTaskListAvailable() (vi.widgets.topbar.Tasks method), 46
 onTaskListFailure() (vi.widgets.topbar.Tasks method), 46
 onUploadAdded() (vi.widgets.file.MultiUploader method), 35
 onUploadAdded() (vi.widgets.file.Uploader method), 35
 onUploadUrlAvailable() (vi.widgets.file.MultiUploader method), 35
 onUploadUrlAvailable() (vi.widgets.file.Uploader method), 35
 onUserTestFail() (vi.widgets.UserLogoutMsg method), 55
 onUserTestFail() (vi.widgets.userlogoutmsg.UserLogoutMsg method), 50
 onUserTestSuccess() (vi.widgets.UserLogoutMsg method), 55
 onUserTestSuccess() (vi.widgets.userlogoutmsg.UserLogoutMsg method), 50
 onVerifyClick() (vi.login.UserPasswordLoginHandler method), 64
 onViewfocusedChanged() (vi.views.list.listHandlerWidget method), 26
 onViewfocusedChanged() (vi.views.singleton.singletonHandlerWidget method), 27
 openEdit() (vi.widgets.topbar.UserState method), 46
 openEditor() (vi.actions.hierarchy.CloneAction method), 9
 openEditor() (vi.actions.hierarchy.EditAction method), 8
 openEditor() (vi.actions.list.CloneAction method), 11
 openEditor() (vi.actions.list.EditAction method), 11
 openEditor() (vi.log.logA method), 62
 openHelp() (vi.widgets.code.Scripter method), 32
 openLog() (vi.log.LogButton method), 63
 openModule() (vi.actions.context.ContextAction method), 5
 openNewMainView() (vi.admin.AdminScreen method), 60
 openNewMainView() (vi.AdminScreen method), 72
 openNewPopup() (vi.admin.AdminScreen method), 60
 openNewPopup() (vi.AdminScreen method), 72
 openView() (vi.admin.AdminScreen method), 60
 openView() (vi.AdminScreen method), 72
 Overview (class in vi.views.overview), 27
 OverviewWidget (class in vi.views.overview), 27
- ## P
- PageFindAction (class in vi.actions.list), 14
 Pane (class in vi.pane), 65
 parseAnswer() (vi.login.BaseLoginHandler method), 64
 ParsedErrorItem (class in vi.widgets.internaedit), 37
 parseHashParameters() (in module vi.widgets.edit), 33
 PassiveErrorItem (class in vi.widgets.internaedit), 37
 performLogics() (vi.widgets.InternalEdit method), 56
 performLogics() (vi.widgets.internaedit.InternalEdit method), 38
 performReload() (vi.actions.edit.Refresh method), 6
 pollInterval (vi.widgets.UserLogoutMsg attribute), 55
 pollInterval (vi.widgets.userlogoutmsg.UserLogoutMsg attribute), 49
 postInit() (vi.actions.list.TableItems method), 13
 postInit() (vi.actions.list.TableNextPage method), 13
 postInit() (vi.actions.list.TablePrevPage method), 13
 preloadIcons() (in module vi), 73
 prepareCol() (vi.framework.components.datatable.SelectTable method), 21
 prepareCol() (vi.widgets.table.SelectTable method), 42
 PythonCode (class in vi.widgets.code), 31
- ## Q
- queue (vi.utils.indexeddb attribute), 70
- ## R
- rebuildCB() (vi.actions.list.ListPreviewAction method), 12
 rebuildChildrenClassInfo() (vi.pane.Pane method), 66
 rebuildPath() (vi.widgets.tree.TreeBrowserWidget method), 49
 rebuildPath() (vi.widgets.TreeBrowserWidget method), 58
 rebuildTable() (vi.framework.components.datatable.DataTable method), 22
 rebuildTable() (vi.framework.components.datatable.ViewportDataTable method), 22
 rebuildTable() (vi.widgets.DataTable method), 54

rebuildTable() (*vi.widgets.table.DataTable* method), 43
 recalcHeight() (*vi.widgets.DataTable* method), 53
 recalcHeight() (*vi.widgets.table.DataTable* method), 43
 receivedStructure() (*vi.widgets.list.ListWidget* method), 39
 receivedStructure() (*vi.widgets.ListWidget* method), 52
 receivedStructure() (*vi.widgets.tree.TreeWidget* method), 47
 receivedStructure() (*vi.widgets.TreeWidget* method), 57
 redirectNoAdmin() (*vi.login.LoginScreen* method), 65
 redirectNoAdmin() (*vi.LoginScreen* method), 71
 reevaluate() (*vi.sidebarwidgets.filtersselector.CompoundFilter* method), 23
 reevaluate() (*vi.widgets.Search* method), 55
 reevaluate() (*vi.widgets.search.Search* method), 40
 Refresh (class in *vi.actions.edit*), 6
 refresh() (*vi.admin.AdminScreen* method), 60
 refresh() (*vi.AdminScreen* method), 72
 refreshConfig() (*vi.admin.AdminScreen* method), 60
 refreshConfig() (*vi.AdminScreen* method), 72
 registerScroll() (*vi.actions.list.LoadNextBatchAction* method), 14
 ReloadAction (class in *vi.actions.hierarchy*), 9
 ReloadAction (class in *vi.actions.list*), 13
 ReloadAction (class in *vi.actions.tree*), 18
 reloadData() (*vi.widgets.edit.EditWidget* method), 33
 reloadData() (*vi.widgets.EditWidget* method), 53
 reloadData() (*vi.widgets.hierarchy.HierarchyWidget* method), 36
 reloadData() (*vi.widgets.HierarchyWidget* method), 58
 reloadData() (*vi.widgets.list.ListWidget* method), 39
 reloadData() (*vi.widgets.ListWidget* method), 52
 reloadData() (*vi.widgets.tree.TreeBrowserWidget* method), 49
 reloadData() (*vi.widgets.tree.TreeWidget* method), 48
 reloadData() (*vi.widgets.TreeBrowserWidget* method), 58
 reloadData() (*vi.widgets.TreeWidget* method), 57
 reloadListWidget() (*vi.widgets.hierarchy.HierarchyWidget* method), 36
 reloadListWidget() (*vi.widgets.HierarchyWidget* method), 59
 remove() (*vi.framework.components.datatable.DataTable* method), 21
 remove() (*vi.screen.Screen* method), 68
 remove() (*vi.widgets.DataTable* method), 54
 remove() (*vi.widgets.table.DataTable* method), 43
 RemoveAction() (*vi.widgets.appnavigation.NavigationElement* method), 30
 removeChildPane() (*vi.pane.Pane* method), 66
 removeInfo() (*vi.log.LogButton* method), 63
 removeNavigationPoint() (*vi.widgets.appnavigation.AppNavigation* method), 31
 removeNewCls() (*vi.log.Log* method), 63
 removeRow() (*vi.framework.components.datatable.SelectTable* method), 21
 removeRow() (*vi.widgets.table.SelectTable* method), 42
 removeSelectedRow() (*vi.framework.components.datatable.SelectTable* method), 20
 removeSelectedRow() (*vi.widgets.table.SelectTable* method), 42
 removeWidget() (*vi.admin.AdminScreen* method), 60
 removeWidget() (*vi.AdminScreen* method), 72
 removeWidget() (*vi.pane.Pane* method), 66
 render_url() (in module *vi.utils*), 69
 renderPopOut() (*vi.log.LogButton* method), 62
 renderStructure() (*vi.widgets.InternalEdit* method), 56
 renderStructure() (*vi.widgets.internaedit.InternalEdit* method), 37
 replaceWithMessage() (*vi.widgets.csvexport.ExportCsv* method), 32
 replaceWithMessage() (*vi.widgets.ExportCsv* method), 56
 replaceWithMessage() (*vi.widgets.file.MultiUploader* method), 35
 replaceWithMessage() (*vi.widgets.file.Uploader* method), 35
 request() (in module *webworker_scripts*), 74
 request() (*webworker_scripts.SimpleNetwork* method), 75
 requestChildren() (*vi.widgets.tree.TreeWidget* method), 47
 requestChildren() (*vi.widgets.TreeWidget* method), 57
 requestData() (*webworker_scripts.requestList* method), 74
 requestList (class in *webworker_scripts*), 74
 requestStructure() (*vi.widgets.list.ListWidget* method), 39
 requestStructure() (*vi.widgets.ListWidget* method), 52
 requestStructure() (*vi.widgets.tree.TreeWidget* method), 47
 requestStructure() (*vi.widgets.TreeWidget* method), 57
 reset() (*vi.admin.AdminScreen* method), 60
 reset() (*vi.AdminScreen* method), 72
 reset() (*vi.log.Log* method), 63
 reset() (*vi.log.LogButton* method), 63
 reset() (*vi.login.BaseLoginHandler* method), 64

- reset() (*vi.login.UserPasswordLoginHandler* method), 64
 reset() (*vi.priorityqueue.StartupQueue* method), 67
 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), 9
 resetLoadingState() (*vi.actions.hierarchy.ReloadAction* method), 9
 resetLoadingState() (*vi.actions.list.AddAction* method), 11
 resetLoadingState() (*vi.actions.list.CloneAction* method), 11
 resetLoadingState() (*vi.actions.list.DeleteAction* method), 12
 resetLoadingState() (*vi.actions.list.EditAction* method), 11
 resetLoadingState() (*vi.actions.list.LoadAllAction* method), 14
 resetLoadingState() (*vi.actions.list.LoadNextBatchAction* method), 14
 resetLoadingState() (*vi.actions.list.PageFindAction* method), 14
 resetLoadingState() (*vi.actions.list.ReloadAction* method), 13
 resetLoadingState() (*vi.actions.list.SetPageRowAmountAction* method), 14
 resetLoadingState() (*vi.actions.list.TableNextPage* method), 13
 resetLoadingState() (*vi.actions.tree.AddLeafAction* method), 17
 resetLoadingState() (*vi.actions.tree.AddNodeAction* method), 17
 resetLoadingState() (*vi.actions.tree.AddNodeOnlyAction* method), 17
 resetLoadingState() (*vi.actions.tree.DeleteAction* method), 18
 resetLoadingState() (*vi.actions.tree.EditAction* method), 18
 resetLoadingState() (*vi.actions.tree.ReloadAction* method), 18
 resetLoadingState() (*vi.framework.components.actionbar.ActionBar* method), 19
 resetLoadingState() (*vi.serversideaction.ServerSideActionWdg* method), 69
 resetLoadingState() (*vi.widgets.Search* method), 55
 resetLoadingState() (*vi.widgets.search.Search* method), 40
 resetSearch() (*vi.widgets.Search* method), 55
 resetSearch() (*vi.widgets.search.Search* method), 40
 run() (*vi.priorityqueue.StartupQueue* method), 67
 run() (*vi.widgets.code.PythonCode* method), 32
 runClick() (*vi.widgets.code.Scripiter* method), 32
 running() (*webworker_scripts.requestList* method), 74
- ## S
- s (in module *vi*), 73
 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*), 73
 scinv (in module *vi*), 73
 Screen (class in *vi.screen*), 68
 Scripiter (class in *vi.widgets.code*), 32
 Scripiter (class in *vi.widgets.topbar*), 46
 Search (class in *vi.widgets*), 54
 Search (class in *vi.widgets.search*), 40
 searchWidget() (*vi.widgets.file.FileWidget* method), 36
 searchWidget() (*vi.widgets.FileWidget* method), 59
 SelectAction (class in *vi.actions.list*), 12
 selectAll() (*vi.framework.components.datatable.SelectTable* method), 21
 selectAll() (*vi.widgets.table.SelectTable* method), 42
 SelectAllAction (class in *vi.actions.list*), 15

- SelectFieldsAction (class in *vi.actions.list*), 13
 SelectFieldsPopup (class in *vi.actions.list*), 12
 selectHandler() (*vi.login.LoginScreen* method), 65
 selectHandler() (*vi.LoginScreen* method), 71
 SelectInvertAction (class in *vi.actions.list*), 15
 selectorReturn() (*vi.widgets.list.ListWidget* method), 38
 selectorReturn() (*vi.widgets.ListWidget* method), 51
 selectorReturn() (*vi.widgets.tree.TreeWidget* method), 47
 selectorReturn() (*vi.widgets.TreeWidget* method), 57
 SelectRootNode (class in *vi.actions.tree*), 18
 selectRow() (*vi.framework.components.datatable.SelectTable* method), 20
 selectRow() (*vi.widgets.table.SelectTable* method), 42
 SelectTable (class in *vi.framework.components.datatable*), 19
 SelectTable (class in *vi.widgets.table*), 41
 seperatorAction() (*vi.widgets.appnavigation.NavigationButtons* method), 30
 serializeForDocument() (*vi.widgets.InternalEdit* method), 56
 serializeForDocument() (*vi.widgets.internaedit.InternalEdit* method), 37
 serializeForPost() (*vi.widgets.InternalEdit* method), 56
 serializeForPost() (*vi.widgets.internaedit.InternalEdit* method), 37
 ServerSideActionWdg (class in *vi.serversideaction*), 68
 ServerTaskWidget (class in *vi.widgets.task*), 44
 setActions() (*vi.framework.components.actionbar.ActionBar* method), 19
 setActiveTask() (*vi.widgets.task.TaskSelectWidget* method), 44
 setActiveTask() (*vi.widgets.TaskSelectWidget* method), 56
 setAmount() (*vi.widgets.list.ListWidget* method), 38
 setAmount() (*vi.widgets.list.ViewportListWidget* method), 39
 setAmount() (*vi.widgets.ListWidget* method), 51
 setCell() (*vi.framework.components.datatable.SelectTable* method), 21
 setCell() (*vi.widgets.table.SelectTable* method), 42
 setCellRender() (*vi.framework.components.datatable.DataTable* method), 22
 setCellRender() (*vi.widgets.DataTable* method), 54
 setCellRender() (*vi.widgets.table.DataTable* method), 44
 setCellRenders() (*vi.framework.components.datatable.DataTable* method), 22
 setCellRenders() (*vi.widgets.DataTable* method), 54
 setCellRenders() (*vi.widgets.table.DataTable* method), 44
 setContext() (*vi.widgets.list.ListWidget* method), 39
 setContext() (*vi.widgets.ListWidget* method), 52
 setContext() (*vi.widgets.tree.TreeWidget* method), 47
 setContext() (*vi.widgets.TreeWidget* method), 57
 setCurrentModulDescr() (*vi.widgets.topbar.TopBarWidget* method), 45
 setCurrentModulDescr() (*vi.widgets.TopBarWidget* method), 51
 setCursor() (*vi.widgets.code.Codemirror* method), 31
 setCursorPosition() (*vi.framework.components.datatable.SelectTable* method), 20
 setCursorPosition() (*vi.widgets.table.SelectTable* method), 42
 setData() (*vi.widgets.edit.EditWidget* method), 34
 setData() (*vi.widgets.EditWidget* method), 53
 setDataProvider() (*vi.framework.components.datatable.DataTable* method), 21
 setDataProvider() (*vi.widgets.DataTable* method), 53
 setDataProvider() (*vi.widgets.table.DataTable* method), 43
 setFields() (*vi.widgets.list.ListWidget* method), 39
 setFields() (*vi.widgets.ListWidget* method), 52
 setFile() (*vi.widgets.file.FilePreviewImage* method), 35
 setFilter() (*vi.widgets.list.ListWidget* method), 39
 setFilter() (*vi.widgets.ListWidget* method), 52
 setFinalElem() (*vi.priorityqueue.StartupQueue* method), 67
 setHeader() (*vi.framework.components.datatable.SelectTable* method), 20
 setHeader() (*vi.widgets.table.SelectTable* method), 41
 setImage() (*vi.pane.Pane* method), 66
 setListView() (*vi.widgets.hierarchy.HierarchyWidget* method), 36
 setListView() (*vi.widgets.HierarchyWidget* method), 59
 setPage() (*vi.widgets.list.ListWidget* method), 38
 setPage() (*vi.widgets.list.ViewportListWidget* method), 39
 setPage() (*vi.widgets.ListWidget* method), 51
 setPageAmount() (*vi.actions.list.SetPageRowAmountAction* method), 14
 SetPageRowAmountAction (class in *vi.actions.list*), 13
 setPath() (*vi.Application* method), 73
 setPayed() (*vi.actions.list_order.ShopMarkAction* method), 16
 setPayedFailed() (*vi.actions.list_order.ShopMarkAction* method), 16
 setPayedSucceeded() (*vi.actions.list_order.ShopMarkAction* method), 16
 setPreventUnloading() (in module *vi.utils*), 69
 setRootNode() (*vi.widgets.tree.TreeWidget* method), 48

- setRootNode() (*vi.widgets.TreeWidget* method), 57
 setSelector() (*vi.widgets.list.ListWidget* method), 38
 setSelector() (*vi.widgets.ListWidget* method), 51
 setSelector() (*vi.widgets.tree.TreeWidget* method), 47
 setSelector() (*vi.widgets.TreeWidget* method), 57
 setShownFields() (*vi.framework.components.datatable.DataTable* method), 22
 setShownFields() (*vi.widgets.DataTable* method), 54
 setShownFields() (*vi.widgets.table.DataTable* method), 43
 setStyle() (*vi.widgets.file.FileLeafWidget* method), 35
 setStyle() (*vi.widgets.file.FileNodeWidget* method), 36
 setStyle() (*vi.widgets.tree.BreadcrumbNodeWidget* method), 48
 setStyle() (*vi.widgets.tree.BrowserLeafWidget* method), 48
 setStyle() (*vi.widgets.tree.BrowserNodeWidget* method), 48
 setTableActionBar() (*vi.widgets.list.ListWidget* method), 38
 setTableActionBar() (*vi.widgets.list.ViewportListWidget* method), 40
 setTableActionBar() (*vi.widgets.ListWidget* method), 51
 setText() (*vi.pane.Pane* method), 66
 setTitle() (*vi.Application* method), 73
 setTitle() (*vi.screen.Screen* method), 68
 setTitle() (*vi.widgets.topbar.TopBarWidget* method), 45
 setTitle() (*vi.widgets.TopBarWidget* method), 51
 setView() (*vi.sidebarwidgets.filtersselector.FilterSelector* method), 23
 setWidget() (*vi.widgets.SideBar* method), 55
 setWidget() (*vi.widgets.sidebar.SideBar* method), 41
 ShopMarkAction (*class in vi.actions.list_order*), 16
 ShopMarkCanceledAction (*class in vi.actions.list_order*), 16
 ShopMarkPayedAction (*class in vi.actions.list_order*), 16
 ShopMarkSentAction (*class in vi.actions.list_order*), 16
 showErrorMsg() (*vi.widgets.edit.EditWidget* method), 33
 showErrorMsg() (*vi.widgets.EditWidget* method), 53
 showErrorMsg() (*vi.widgets.list.ListWidget* method), 38
 showErrorMsg() (*vi.widgets.ListWidget* method), 51
 showListView() (*vi.widgets.hierarchy.HierarchyWidget* method), 36
 showListView() (*vi.widgets.HierarchyWidget* method), 59
 showLoginWindow() (*vi.widgets.UserLogoutMsg* method), 55
 showLoginWindow() (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), 49
 showMessage() (*vi.widgets.UserLogoutMsg* method), 55
 showMessage() (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), 49
 SideBar (*class in vi.widgets*), 55
 SideBar (*class in vi.widgets.sidebar*), 40
 SimpleNetwork (*class in webworker_scripts*), 75
 SingletonHandler (*class in vi.views.singleton*), 27
 singletonHandlerWidget (*class in vi.views.singleton*), 27
 stackWidget() (*vi.admin.AdminScreen* method), 60
 stackWidget() (*vi.AdminScreen* method), 72
 start() (*in module vi*), 73
 startFind() (*vi.actions.list.PageFindAction* method), 14
 startFullscreen() (*vi.widgets.code.Scripiter* method), 32
 startPolling() (*vi.widgets.UserLogoutMsg* method), 55
 startPolling() (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), 50
 startup() (*vi.admin.AdminScreen* method), 60
 startup() (*vi.AdminScreen* method), 72
 startup() (*vi.Application* method), 73
 startupFailure() (*vi.Application* method), 73
 StartupQueue (*class in vi.priorityqueue*), 67
 startupQueue (*in module vi.priorityqueue*), 67
 stop() (*vi.widgets.code.PythonCode* method), 32
 stopInterval() (*vi.widgets.UserLogoutMsg* method), 55
 stopInterval() (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), 49
 style (*vi.widgets.internaledit.ParsedErrorItem* attribute), 37
 style (*vi.widgets.internaledit.PassiveErrorItem* attribute), 37
 switchDisabledState() (*vi.serversideaction.ServerSideActionWdg* method), 68
 switchFullscreen() (*vi.admin.AdminScreen* method), 61
 switchFullscreen() (*vi.AdminScreen* method), 72
- ## T
- tableInitialization() (*vi.widgets.list.ListWidget* method), 38
 tableInitialization() (*vi.widgets.list.ViewportListWidget* method), 39
 tableInitialization() (*vi.widgets.ListWidget* method), 51
 TableItems (*class in vi.actions.list*), 13
 TableNextPage (*class in vi.actions.list*), 13
 TablePrevPage (*class in vi.actions.list*), 13
 Tasks (*class in vi.widgets.topbar*), 46
 TaskSelectWidget (*class in vi.widgets*), 56
 TaskSelectWidget (*class in vi.widgets.task*), 44

- TaskWidget (class in *vi.widgets*), 55
 TaskWidget (class in *vi.widgets.task*), 44
 testIfNextBatchNeededImmediately() (vi.widgets.DataTable method), 54
 testIfNextBatchNeededImmediately() (vi.widgets.table.DataTable method), 43
 toggle() (vi.widgets.accordion.AccordionSegment method), 28
 toggleIntPrev() (vi.actions.list.ListPreviewInlineAction method), 12
 toggleListView() (vi.widgets.hierarchy.HierarchyWidget method), 36
 toggleListView() (vi.widgets.HierarchyWidget method), 59
 toggleMsgCenter() (vi.log.Log method), 63
 ToolTip (class in *vi.widgets*), 53
 ToolTip (class in *vi.widgets.tooltip*), 45
 TopBarWidget (class in *vi.widgets*), 51
 TopBarWidget (class in *vi.widgets.topbar*), 45
 toplevelActionSelector (in module *vi.priorityqueue*), 67
 tpl (vi.widgets.appnavigation.NavigationElement attribute), 29
 TreeBrowserWidget (class in *vi.widgets*), 58
 TreeBrowserWidget (class in *vi.widgets.tree*), 48
 treeHandler (class in *vi.views.tree*), 28
 treeHandlerWidget (class in *vi.views.tree*), 28
 TreeWidget (class in *vi.widgets*), 56
 TreeWidget (class in *vi.widgets.tree*), 47
- ## U
- unlock() (vi.login.BaseLoginHandler method), 64
 unlock() (vi.pane.Pane method), 66
 unlock() (vi.screen.Screen method), 68
 unselectAll() (vi.framework.components.datatable.SelectTable method), 21
 unselectAll() (vi.widgets.table.SelectTable method), 42
 UnselectAllAction (class in *vi.actions.list*), 15
 unserialize() (vi.widgets.InternalEdit method), 56
 unserialize() (vi.widgets.internaledit.InternalEdit method), 37
 update() (vi.actions.tree.SelectRootNode method), 18
 update() (vi.framework.components.datatable.DataTable method), 21
 update() (vi.framework.components.datatable.ViewportDataTable method), 23
 update() (vi.widgets.topbar.Tasks method), 46
 update() (vi.widgets.topbar.UserState method), 46
 updateConf() (in module *vi.config*), 61
 updateEmptyNotification() (vi.widgets.list.ListWidget method), 39
 updateEmptyNotification() (vi.widgets.ListWidget method), 52
 updateUser() (vi.widgets.topbar.Scripiter method), 46
 Uploader (class in *vi.widgets.file*), 35
 UserLogoutMsg (class in *vi.widgets*), 55
 UserLogoutMsg (class in *vi.widgets.userlogoutmsg*), 49
 UserPasswordLoginHandler (class in *vi.login*), 64
 UserState (class in *vi.widgets.topbar*), 45
- ## 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, 16
 vi.actions.tree module, 17
 vi.admin module, 59
 vi.config module, 61
 vi.exception module, 61
 vi.framework module, 19
 vi.framework.components module, 19
 vi.framework.components.actionbar module, 19
 vi.framework.components.datatable module, 19
 vi.log module, 62
 vi.login module, 63
 vi.pane module, 65
 vi.priorityqueue module, 67
 vi.screen module, 68
 vi.serversideaction module, 68
 vi.sidebarwidgets module, 23

vi.sidebarwidgets.filterselector
 module, 23
 vi.sidebarwidgets.internalpreview
 module, 24
 vi.translations
 module, 24
 vi.translations.de
 module, 24
 vi.translations.en
 module, 24
 vi.utils
 module, 69
 vi.views
 module, 24
 vi.views.edit
 module, 24
 vi.views.hierarchy
 module, 25
 vi.views.list
 module, 25
 vi.views.log
 module, 26
 vi.views.notfound
 module, 26
 vi.views.overview
 module, 27
 vi.views.singleton
 module, 27
 vi.views.tree
 module, 28
 vi.widgets
 module, 28
 vi.widgets.accordion
 module, 28
 vi.widgets.appnavigation
 module, 29
 vi.widgets.code
 module, 31
 vi.widgets.csvexport
 module, 32
 vi.widgets.edit
 module, 33
 vi.widgets.file
 module, 34
 vi.widgets.hierarchy
 module, 36
 vi.widgets.internaedit
 module, 37
 vi.widgets.list
 module, 38
 vi.widgets.search
 module, 40
 vi.widgets.sidebar
 module, 40
 vi.widgets.table
 module, 41
 vi.widgets.task
 module, 44
 vi.widgets.tooltip
 module, 45
 vi.widgets.topbar
 module, 45
 vi.widgets.tree
 module, 46
 vi.widgets.userlogoutmsg
 module, 49
 vi_conf (in module vi), 71
 vi_conf (in module vi.config), 61
 ViewportDataTable (class in vi.framework.components.datatable), 22
 ViewportListWidget (class in vi.widgets.list), 39
 viInitializedEvent (in module vi.admin), 61
 visibilityChanged() (vi.widgets.UserLogoutMsg method), 55
 visibilityChanged() (vi.widgets.userlogoutmsg.UserLogoutMsg method), 49

W

warn() (webworker_scripts.weblog static method), 74
 weblog (class in webworker_scripts), 74
 webworker_scripts
 module, 73
 workerFeedback() (vi.widgets.code.PythonCode method), 31
 writeRow() (webworker_scripts.csvWriter method), 74
 writeRows() (webworker_scripts.csvWriter method), 74