

---

**ViUR Vi**  
*Release 3.0*

**Tilman Oestereich, Andreas H. Kelch**

**Aug 06, 2021**



<b>1 About</b>	<b>3</b>
<b>Python Module Index</b>	<b>71</b>
<b>Index</b>	<b>73</b>





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 Acionbars

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 documentation<sup>1</sup>.

### 1.4.1 vi

#### Subpackages

`vi.actions`

#### Submodules

`vi.actions.context`

#### Module Contents

#### Classes

---

<sup>1</sup> Created with sphinx-autoapi



---

*ContextAction*

---

```

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

    onAttach(self)
    onDetach(self)
    onSelectionChanged(self, table, selection)
    onClick(self, sender=None)
    openModule(self, data, title=None)
    static isSuitableFor(module, handler, actionName)

```

vi.actions.edit

**Module Contents****Classes**

---

*SaveContinue*

---

---

*SaveSingleton*

---

---

*ExecuteSingleton*

---

---

*SaveClose*

---

---

*Refresh*

---

---

*CancelClose*

---

```

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

    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    resetLoadingState(self)

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

    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    resetLoadingState(self)

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

```

```
    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    resetLoadingState(self)
class vi.actions.edit.SaveClose(*args, **kwargs)
    Bases: flare.button.Button
    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    resetLoadingState(self)
class vi.actions.edit.Refresh(*args, **kwargs)
    Bases: flare.button.Button
    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    performReload(self, sender=None)
    resetLoadingState(self)
class vi.actions.edit.CancelClose(*args, **kwargs)
    Bases: flare.button.Button
    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    resetLoadingState(self)
```

`vi.actions.file`

## Module Contents

### Classes

---

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

---

```
class vi.actions.file.FileSelectUploader(*args, **kwargs)
    Bases: flare.html5.Input
    Small wrapper around <input type="file">. Creates the element; executes the click (=opens the file dialog); runs
    the callback if a file has been selected and removes itself from its parent.
    onChange(self, event)
class vi.actions.file.AddNodeAction(*args, **kwargs)
    Bases: flare.button.Button
    Adds a new directory to a tree.simple application.
    static isSuitableFor(module, handler, actionName)
```

```

onClick(self, sender=None)
createDir(self, dialog, dirName)
onMkDir(self, req)
resetLoadingState(self)
class vi.actions.file.AddLeafAction(*args, **kwargs)
    Bases: flare.button.Button

    Allows uploading of files using the file dialog.

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

    Provides editing in a tree.simple application. If a directory is selected, it opens a dialog for renaming that directory, otherwise the full editWidget is used.

onAttach(self)
onDetach(self)
onSelectionActivated(self, table, selection)
onSelectionChanged(self, table, selection)
static isSuitableFor(module, handler, actionName)
onClick(self, sender=None)
editDir(self, dialog, dirName)
resetLoadingState(self)
class vi.actions.file.DownloadAction(*args, **kwargs)
    Bases: flare.button.Button

    Allows downloading files from the server.

onAttach(self)
onDetach(self)
onSelectionChanged(self, table, selection)
static isSuitableFor(module, handler, actionName)
onClick(self, sender=None)
disableViUnloadingWarning(self, *args, **kwargs)
enableViUnloadingWarning(self, *args, **kwargs)
doDownload(self, fileData)
resetLoadingState(self)

```

**vi.actions.hierarchy****Module Contents****Classes**

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

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

Bases: `flare.button.Button`

Adds a new node in a hierarchy application.

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

`onClick(self, sender=None)`

`resetLoadingState(self)`

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

Bases: `flare.button.Button`

Edits a node in a hierarchy application.

`onAttach(self)`

`onDetach(self)`

`onSelectionChanged(self, table, selection)`

`onSelectionActivated(self, table, selection)`

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

`onClick(self, sender=None)`

`openEditor(self, key)`

`resetLoadingState(self)`

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

Bases: `flare.button.Button`

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

`onAttach(self)`

`onDetach(self)`

`onSelectionChanged(self, table, selection)`

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

`onClick(self, sender=None)`

`openEditor(self, key)`

```

    resetLoadingState(self)
class vi.actions.hierarchy.DeleteAction(*args, **kwargs)
    Bases: flare.button.Button

    Deletes a node from a hierarchy application.

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

    Allows adding an entry in a list-module.

    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    resetLoadingState(self)
class vi.actions.hierarchy.SelectRootNode(module, handler, actionName, *args, **kwargs)
    Bases: flare.html5.Select

    Selector for hierarchy root nodes.

    onAttach(self)
    onDetach(self)
    update(self)
    onRootNodeChanged(self, newNode)
    onRootNodesAvailable(self, req)
    onChange(self, event)
    static isSuitableFor(module, handler, actionName)
class vi.actions.hierarchy.ListViewAction(*args, **kwargs)
    Bases: flare.button.Button

    Allows adding an entry in a list-module.

    static isSuitableFor(module, handler, actionName)
    onClick(self, sender=None)
    resetLoadingState(self)

```

vi.actions.list

## Module Contents

### Classes

---

<i>AddAction</i>	Allows adding an entry in a list-module.
<i>EditAction</i>	Allows editing an entry in a list-module.
<i>CloneAction</i>	Allows cloning an entry in a list-module.
<i>DeleteAction</i>	Allows deleting an entry in a list-module.
<i>ListPreviewAction</i>	
<hr/>	
<i>ListPreviewInlineAction</i>	
<hr/>	
<i>CloseAction</i>	
<hr/>	
<i>SelectAction</i>	
<hr/>	
<i>SelectFieldsPopup</i>	
<hr/>	
<i>SelectFieldsAction</i>	
<hr/>	
<i>ReloadAction</i>	Allows Reloading
<i>TableNextPage</i>	
<hr/>	
<i>TablePrevPage</i>	
<hr/>	
<i>TableItems</i>	
<hr/>	
<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>	
<hr/>	
<i>CreateRecurrentAction</i>	
<hr/>	
<i>ExportCsvAction</i>	
<hr/>	
<i>SelectAllAction</i>	
<hr/>	
<i>UnSelectAllAction</i>	
<hr/>	
<i>SelectInvertAction</i>	
<hr/>	

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

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

```
    Allows adding an entry in a list-module.
```

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

```

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

```

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

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

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

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

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

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



```

    onTableChanged(self, table, count)
    static isSuitableFor(module, handler, actionName)
class vi.actions.list.ReloadAction(*args, **kwargs)
    Bases: flare.button.Button
    Allows Reloading
    static isSuitableFor(module, handler, actionName)
    onClick(self, event=None)
    resetLoadingState(self)
class vi.actions.list.TableNextPage(*args, **kwargs)
    Bases: flare.button.Button
    postInit(self, widget=None)
    onClick(self, sender=None)
    static isSuitableFor(module, handler, actionName)
    resetLoadingState(self)
class vi.actions.list.TablePrevPage(*args, **kwargs)
    Bases: flare.button.Button
    postInit(self, widget=None)
    onClick(self, sender=None)
    static isSuitableFor(module, handler, actionName)
class vi.actions.list.TableItems(*args, **kwargs)
    Bases: flare.html5.Div
    postInit(self, widget=None)
    onTableChanged(self, table, rowCount)
    static isSuitableFor(module, handler, actionName)
class vi.actions.list.SetPageRowAmountAction(*args, **kwargs)
    Bases: flare.html5.Div
    Load a bunch of pages
    onClick(self, sender=None)
    onChange(self, sender=None)
    setPageAmount(self)
    static isSuitableFor(module, handler, actionName)
    resetLoadingState(self)
class vi.actions.list.LoadNextBatchAction(*args, **kwargs)
    Bases: flare.html5.Div
    Load a bunch of pages
    registerScroll(self)
    onScroll(self, sender=None)
    onClick(self, sender=None)

```

```
onChange(self, sender=None)  
loadnextPages(self, *args, **kwargs)  
static isSuitableFor(module, handler, actionName)  
resetLoadingState(self)  
class vi.actions.list.LoadAllAction(*args, **kwargs)  
    Bases: flare.button.Button  
    Allows Loading all Entries in a list  
    static isSuitableFor(module, handler, actionName)  
    onClick(self, sender=None)  
    loadAllRows(self)  
    resetLoadingState(self)  
class vi.actions.list.PageFindAction(*args, **kwargs)  
    Bases: flare.html5.Div  
    Allows Loading all Entries in a list  
    onKeyPress(self, event)  
    static isSuitableFor(module, handler, actionName)  
    onClick(self, sender=None)  
    startFind(self)  
    findText(self)  
    resetLoadingState(self)  
class vi.actions.list.ListSelectFilterAction(*args, **kwargs)  
    Bases: flare.button.Button  
    onAttach(self)  
    onClick(self, sender=None)  
    static isSuitableFor(module, handler, actionName)  
class vi.actions.list.CreateRecurrentAction(*args, **kwargs)  
    Bases: flare.button.Button  
    static isSuitableFor(module, handler, actionName)  
    onClick(self, sender=None)  
class vi.actions.list.ExportCsvAction(*args, **kwargs)  
    Bases: flare.button.Button  
    onClick(self, sender=None)  
    static isSuitableFor(module, handler, actionName)  
class vi.actions.list.SelectAllAction(*args, **kwargs)  
    Bases: flare.button.Button  
    static isSuitableFor(module, handler, actionName)  
    onClick(self, sender=None)  
    onAttach(self)
```

```

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

```

vi.actions.list\_order

## Module Contents

### Classes

---

*ShopMarkAction*

---

*ShopMarkPayedAction*

---

*ShopMarkSentAction*

---

*ShopMarkCanceledAction*

---

```

class vi.actions.list_order.ShopMarkAction(action, title, cls="", txtQuestion=None, txtSuccess=None,
                                           txtFailure=None, *args, **kwargs)
    Bases: flare.button.Button
    onAttach(self)
    onDetach(self)
    onSelectionChanged(self, table, selection)
    setPayed(self, order)
    setPayedSucceeded(self, response)
    setPayedFailed(self, response)

```

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

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

**class** vi.actions.list\_order.**ShopMarkPayedAction**(\*args, \*\*kwargs)

Bases: *ShopMarkAction*

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

**class** vi.actions.list\_order.**ShopMarkSentAction**(\*args, \*\*kwargs)

Bases: *ShopMarkAction*

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

**class** vi.actions.list\_order.**ShopMarkCanceledAction**(\*args, \*\*kwargs)

Bases: *ShopMarkAction*

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

vi.actions.tree

## Module Contents

### Classes

---

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

---

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

Bases: flare.button.Button

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

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

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

**resetLoadingState**(*self*)

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

Bases: flare.button.Button

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

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

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

**resetLoadingState**(*self*)

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

Bases: flare.button.Button

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

**onAttach**(*self*)

```

onDetach(self)
onSelectionActivated(self, table, selection)
onSelectionChanged(self, table, selection)
static isSuitableFor(module, handler, actionName)
onClick(self, sender=None)
resetLoadingState(self)
class vi.actions.tree.DeleteAction(*args, **kwargs)
    Bases: flare.button.Button

    Allows deleting an entry in a tree-module. The type (node or leaf) of the entry is determined dynamically.
onAttach(self)
onDetach(self)
onSelectionChanged(self, table, selection)
static isSuitableFor(module, handler, actionName)
onClick(self, sender=None)
doDelete(self, dialog)
allDeletedSuccess(self, success)
resetLoadingState(self)
class vi.actions.tree.ReloadAction(*args, **kwargs)
    Bases: flare.button.Button

    Allows adding an entry in a list-module.
static isSuitableFor(module, handler, actionName)
onClick(self, sender=None)
resetLoadingState(self)
class vi.actions.tree.SelectRootNode(module, handler, actionName, *args, **kwargs)
    Bases: flare.html5.Select

    Allows selecting a different rootNode in Tree applications
onAttach(self)
onDetach(self)
update(self)
onRootNodeChanged(self, newNode)
onRootNodesAvailable(self, req)
onChange(self, event)
static isSuitableFor(module, handler, actionName)

```

`vi.framework`

## Subpackages

`vi.framework.components`

## Submodules

`vi.framework.components.actionbar`

## Module Contents

### Classes

---

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

---

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

Bases: `flare.html5.Div`

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

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

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

**getActions**(*self*)

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

**resetLoadingState**(*self*)

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

`vi.framework.components.datatable`

## Module Contents

### Classes

---

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

---

`DataTable`

`ViewportDataTable`

---

---

```
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**(*self*)

**setHeader**(*self, headers*)

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

**getTrByIndex**(*self, idx*)

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

**getIndexByTr**(*self, tr*)

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

**\_rowForEvent**(*self, event*)

Determines the row number for the given event

**onChange**(*self, event*)

**onMouseDown**(*self, event*)

**onMouseOut**(*self, event*)

**onMouseUp**(*self, event*)

**onKeyDown**(*self, event*)

**onKeyUp**(*self, event*)

**onDbClick**(*self, event*)

**addSelectedRow**(*self, row*)

Marks a row as selected

**removeSelectedRow**(*self, row*)

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

**selectRow**(*self, newRow*)

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

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

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

**focusRow**(*self, row*)

**getCurrentSelection**(*self*)

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

**clear**(*self*)

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

**removeRow**(*self*, *row*)

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

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

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

**dropTableContent**(*self*)

Drops content from the table, structure remains unchanged

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

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

**selectAll**(*self*)

Selects all entries of the table.

**unselectAll**(*self*)

Unselects all entries of the table.

**invertSelection**(*self*)

Inverts the current selection on the whole table currently displayed.

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

Bases: flare.html5.Div

**setDataProvider**(*self*, *obj*)

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

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

Ensure the table scrolls according to the position of its cursor

**getRowCount**(*self*)

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

**add**(*self*, *obj*)

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

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

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

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

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

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

Flushes the whole table.

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

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



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

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

**setShownFields**(*self*, *fields*)

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

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

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

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

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

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

Re-emit the event.

**getCurrentSelection**(*self*)

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

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

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

**setCellRenderers**(*self*, *renders*)

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

**activateCurrentSelection**(*self*)

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

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

Bases: `DataTable`

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

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

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

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

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

**add**(*self*, *obj*)

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

Override explanation - `_renderObject` call is always prepared

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

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

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

Override explanation - removed grid preparation

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

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

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

`vi.sidebarwidgets`

## Submodules

`vi.sidebarwidgets.filterselector`

## Module Contents

### Classes

---

*CompoundFilter*

---

*FilterSelector*

---

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

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

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

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

```
    focus(self)
```

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

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

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

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

```
    onAttach(self)
```

```
    onDetach(self)
```

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

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

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

`vi.sidebarwidgets.internalpreview`

## Module Contents

### Classes

---

*InternalPreview*

---

```
class vi.sidebarwidgets.internalpreview.InternalPreview(module, structure, item, *args, **kwargs)
```

```
    Bases: flare.html5.Ul
```

`onCopyKey(self, btn)`

`vi.translations`

### Submodules

`vi.translations.de`

### Module Contents

`vi.translations.de.lngDe`

`vi.translations.en`

### Module Contents

`vi.translations.en.lngEn`

### Package Contents

`vi.translations.lngDe`

`vi.translations.lngEn`

`vi.views`

### Submodules

`vi.views.edit`

### Module Contents

### Classes

---

*editHandler*

---

*editHandlerWidget*

---

**class** `vi.views.edit.editHandler`

Bases: `flare.views.view.View`

**class** `vi.views.edit.editHandlerWidget`

Bases: `flare.views.view.ViewWidget`

**initWidget**(*self*)

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**(*self*)

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**(*self*)

Here we start!

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

---

`vi.views.log`

## Module Contents

### Classes

---

*logHandler*

---

*logHandlerWidget*

---

**class** `vi.views.log.logHandler`

Bases: `flare.views.view.View`

**class** `vi.views.log.logHandlerWidget`

Bases: `flare.views.view.ViewWidget`

**initWidget**(*self*)

Here we start!

`vi.views.notfound`

## Module Contents

### Classes

---

*NotFound*

---

*NotFoundWidget*

---

**class** `vi.views.notfound.NotFound`

Bases: `flare.views.view.View`

**class** `vi.views.notfound.NotFoundWidget`

Bases: `flare.views.view.ViewWidget`

**initWidget**(*self*)

Here we start!

`vi.views.overview`

## Module Contents

### Classes

*Overview*

---

*OverviewWidget*

---

```
class vi.views.overview.Overview
    Bases: flare.views.view.View
class vi.views.overview.OverviewWidget
    Bases: flare.views.view.ViewWidget
    initWidget(self)
        Here we start!
```

vi.views.singleton

## Module Contents

### Classes

---

*singletonHandler*

---

*singletonHandlerWidget*

---

```
class vi.views.singleton.singletonHandler
    Bases: flare.views.view.View
    static canHandle(moduleName, moduleInfo)
class vi.views.singleton.singletonHandlerWidget
    Bases: flare.views.view.ViewWidget
    initWidget(self)
        Here we start!
    onViewfocusedChanged(self, viewname, *args, **kwargs)
```

vi.views.tree

## Module Contents

### Classes

---

*treeHandler*

---

*treeHandlerWidget*

---

```
class vi.views.tree.treeHandler
```

Bases: flare.views.view.View

**static canHandle**(*moduleName, moduleInfo*)

**class** vi.views.tree.**treeHandlerWidget**

Bases: flare.views.view.ViewWidget

**initWidget**(*self*)

Here we start!

vi.widgets

Submodules

vi.widgets.accordion

Module Contents

Classes

---

*AccordionSegment*

---

*Accordion*

---

**class** vi.widgets.accordion.**AccordionSegment**(*ident, title=None*)

Bases: flare.html5.Fieldset

**checkVisibility**(*self*)

**activate**(*self*)

**deactivate**(*self*)

**isActive**(*self*)

**toggle**(*self*)

**onClick**(*self, event*)

**addWidget**(*self, widget*)

**class** vi.widgets.accordion.**Accordion**

Bases: flare.html5.Form

**\_segments** = []

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

**clearSegments**(*self*)

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

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

**Returns**

**vi.widgets.appnavigation****Module Contents****Classes***NavigationElement**NavigationSeperator**Navigationblock**AppNavigation*

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

Bases: flare.html5.Div

**tpl** = **Multiline-String**

```

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

```

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

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

Handle Click on Navigation Button



**RemoveAction**(*self*, *e=None*)  
remove this Nav Element

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

**onActiveNavigationChanged**(*self*, *e*, *wdg*)  
What should happen if the State from the surrounding Navigation gets an update

**onHasSubItemsChanged**(*self*, *e*, *wdg*)  
If subChild is added, show itemArrow, hide if no subitem present

**appendSubChild**(*self*, *element*)

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

Bases: flare.html5.Div

**buildSeperator**(*self*)

**\_setValue**(*self*, *value*)

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

Bases: flare.html5.Div

**addSeperator**(*self*)

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

**class** vi.widgets.appnavigation.**AppNavigation**

Bases: flare.html5.Nav

**getPreviousNavigationPoint**(*self*, *view*)

**getNavigationPoint**(*self*, *view*)

**addNavigationBlock**(*self*, *name*)

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

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

**removeNavigationPoint**(*self*, *view*)

vi.widgets.csvexport

## Module Contents

### Classes

---

*ExportCsv*

---

*ExportCsvStarter*

---

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

Bases: flare.html5.Progress

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

**nextChunkComplete**(*self*, *req*)

```
exportToFile(self)
nextChunkFailure(self, req, code)
replaceWithMessage(self, message, logClass='success')
```

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

vi.widgets.edit

## Module Contents

### Classes

---

*ParsedErrorItem*

---

*PassiveErrorItem*

---

*EditWidget*

---

### Functions

---

<i>parseHashParameters</i> (src, prefix="")	Converts a flat dictionary containing dotted properties into a multi-dimensional one.
---	---

---

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

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

```
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!',
    *args, **kwargs)
    Bases: flare.html5.Div
    appList = list
    appHierarchy = hierarchy
```

**appTree** = tree  
**appSingleton** = singleton  
**\_\_editIdx\_\_** = 0  
**onDetach**(*self*)  
**onAttach**(*self*)  
**performLogics**(*self*)  
**onChange**(*self*, *event*)  
**onBoneChange**(*self*, *bone*)  
**showErrorMsg**(*self*, *req=None*, *code=None*)  
 Removes all currently visible elements and displays an error message  
**reloadData**(*self*)  
**save**(*self*, *data*)  
 Creates the actual NetworkService request used to transmit our data. If data is None, it fetches a clean add/edit form.  
     **Parameters data** (*dict* or *None*) – The values to transmit or None to fetch a new, clean add/edit form.  
**clear**(*self*)  
 Removes all visible bones/forms/fieldsets.  
**closeOrContinue**(*self*, *sender=None*)  
**doCloneHierarchy**(*self*, *sender=None*)  
**cloneComplete**(*self*, *request*)  
**formatReadFromClientErrorSeverity**(*self*, *error*)  
**setData**(*self*, *request=None*, *data=None*, *ignoreMissing=False*, *askHierarchyCloning=True*)  
 Rebuilds the UI according to the skeleton received from server  
     **Parameters**  
     • **request** (*NetworkService*) – A finished NetworkService request  
     • **data** (*dict*) – The data received  
**unserialize**(*self*, *data=None*, *errors=()*)  
 Applies the actual data to the bones.  
**serializeForPost**(*self*, *validityCheck=False*)  
**serializeForDocument**(*self*)  
**doSave**(*self*, *closeOnSuccess=False*, *\*args*, *\*\*kwargs*)  
 Starts serializing and transmitting our values to the server.

`vi.widgets.file`

Module Contents

Classes

---

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

---

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

Bases: `flare.popup.Popup`

**onClick**(*self*, *event*)

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

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

Bases: `flare.html5.Div`

**setFile**(*self*, *file*)

**download**(*self*)

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

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

Bases: `flare.html5.Div`

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

**onUploadUrlAvailable**(*self*, *req*)

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

**onSkeyAvailable**(*self*, *req*)

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

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

Internal callback - The state of our upload changed.

**onUploadAdded**(*self*, *req*)

**onProgress**(*self*, *event*)

Internal callback - further bytes have been transmitted

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

Internal callback - The upload succeeded.

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

```

replaceWithMessage(self, message, isSuccess)

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

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

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

class vi.widgets.file.FileWidget(module, rootNode=None, selectMode=None, node=None, context=None,
    *args, **kwargs)
    Bases: vi.widgets.tree.TreeBrowserWidget
    Base Widget that renders a tree.
    leafWidget
    nodeWidget
    searchWidget(self)
    onStartSearch(self, searchStr, *args, **kwargs)
    getChildKey(self, widget)
        Derives a string used to sort the entries on each level
    onDrop(self, event)
        We got a drop event. Make that item a direct child of our rootNode
    static canHandle(module, moduleInfo)

```

`vi.widgets.hierarchy`

## Module Contents

### Classes

---

*HierarchyWidget*

A Hierarchy is a Tree without leaf distiction!

---

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

Bases: `vi.widgets.tree.TreeWidget`

A Hierarchy is a Tree without leaf distiction!

**leafWidget**

**reloadData**(*self*)

Reload the data were displaying.

**reloadListWidget**(*self*)

**toggleListView**(*self*)

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

**showListView**(*self*)

**hideListView**(*self*)

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

**static canHandle**(*moduleName*, *moduleInfo*)

`vi.widgets.htmleditor`

## Module Contents

### Classes

---

*TextInsertImageAction*

---

*HtmlEditor*

---

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

Bases: `flare.button.Button`

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

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

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

**resetLoadingState**(*self*)

**class** `vi.widgets.htmleditor.HtmlEditor(*args, **kwargs)`

Bases: `flare.html5.Textarea`

```

initSources = False
_attachSummernote(self, retry=0)
onAttach(self)
onDetach(self)
onEditorChange(self, e, *args, **kwargs)
_getValue(self)
_setValue(self, val)
enable(self)
disable(self)

```

`vi.widgets.internaledit`

## Module Contents

### Classes

---

*ParsedErrorItem*

---

*PassiveErrorItem*

---

*InternalEdit*

---

### Functions

---

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

---

```

class vi.widgets.internaledit.ParsedErrorItem(error)

```

```

    Bases: flare.html5.Li

```

```

    style = []

```

```

class vi.widgets.internaledit.PassiveErrorItem(error)

```

```

    Bases: flare.html5.Li

```

```

    style = []

```

```

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

```

```

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

```

```

    Bases: flare.html5.Div

```

```

    renderStructure(self, readOnly=False)

```

```

    serializeForPost(self, validityCheck=False)

```

**serializeForDocument**(*self*)

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

Starts serializing and transmitting our values to the server.

**unserialize**(*self*, *data=None*)

Applies the actual data to the bones.

**onChange**(*self*, *event*)

**onKeyDown**(*self*, *event*)

**performLogics**(*self*)

vi.widgets.list

## Module Contents

### Classes

---

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

---

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

Bases: flare.html5.Div

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

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

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

**selectorReturn**(*self*)

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

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

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

**setAmount**(*self*, *amount*)

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

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

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

**onClick**(*self*, *event*)

**setTableActionBar**(*self*)

**getDefaultEntryActions**(*self*)

Returns the list of actions available in our actionBar

**getActions**(*self*)

Returns the list of actions available in our actionBar

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

Returns the list of actions available in the action bar.



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

Removes all currently visible elements and displays an error message

**onNextBatchNeeded**(*self*)

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

**onAttach**(*self*)

**onDetach**(*self*)

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

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

**requestStructure**(*self*)

**receivedStructure**(*self*, *resp*)

**reloadData**(*self*)

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

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

Applies a new filter.

**setContext**(*self*, *context*)

Applies a new context.

**getFilter**(*self*)

**updateEmptyNotification**(*self*)

**onCompletion**(*self*, *req*)

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

**setFields**(*self*, *fields*)

**getFields**(*self*)

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

**activateSelection**(*self*)

**static canHandle**(*moduleName*, *moduleInfo*)

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

Bases: `ListWidget`

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

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

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

#### Override explanation

- use ViewPort DataTable with rows parameter

**setAmount**(*self*, *amount*)

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

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

**Parameters** `page` – sets targetpage

**Returns**

```
_setPage(self, page=0)  
    render page to table :param page: :return:  
onRequestingFinished(self, *args, **kwargs)  
setTableActionBar(self)  
static canHandle(moduleName, moduleInfo)
```

**vi.widgets.preview**

## Module Contents

### Classes

---

*Preview*

---

```
class vi.widgets.preview.Preview(urls, entry, modul, *args, **kwargs)  
    Bases: flare.html5.Div  
    onChange(self, event)  
    setUrl(self, url)  
    doClose(self, *args, **kwargs)
```

**vi.widgets.repeatdate**

## Module Contents

### Classes

---

*RepeatDatePopup*

---

```
class vi.widgets.repeatdate.RepeatDatePopup(module, key)  
    Bases: flare.html5.Div  
    __editIdx__ = 0  
    reloadData(self)  
    save(self, data)  
    setData(self, request=None, data=None, ignoreMissing=False)  
        Rebuilds the UI according to the skeleton received from server  
        Parameters  
        • request (NetworkService) – A finished NetworkService request  
        • data (dict) – The data received  
    clear(self)  
        Removes all visible bones/forms/fieldsets.
```

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

Removes all currently visible elements and displays an error message

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

**vi.widgets.search**

## Module Contents

### Classes

---

*Search*

---

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

Bases: `flare.html5.Div`

**doSearch**(*self*, \*args, \*\*kwargs)

**resetSearch**(*self*)

**onKeyDown**(*self*, *event*)

**resetLoadingState**(*self*)

**reevaluate**(*self*)

**focus**(*self*)

**vi.widgets.sidebar**

## Module Contents

### Classes

---

*SideBar*

---

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

Bases: `flare.html5.Div`

**onAttach**(*self*)

**onDetach**(*self*)

**setWidget**(*self*, *widget*)

**getWidget**(*self*)

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

## vi.widgets.table

### Module Contents

#### Classes

---

<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. (If the user holds ctrl and clicks a row). The selection might contain no, one or multiple rows. Its also called if the cursor moves. Its called if the user simply double clicks a row. So its possible to receive a selectionActivated event without an selectionChanged Event.
- **selectionActivated:** called if a selection is activated, ie. a row is double-clicked or **Return** is pressed.
- **cursorMoved:** called when the currently active row changes. The user can select the current row with a single click or by moving the cursor up and down using the arrow keys.

**onAttach**(self)

**setHeader**(self, headers)

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

**getTrByIndex**(self, idx)

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

**getIndexByTr**(self, tr)

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

**\_rowForEvent**(self, event)

Determines the row number for the given event

**onChange**(self, event)

**onMouseDown**(self, event)

**onMouseOut**(self, event)

**onMouseUp**(self, event)

**onKeyDown**(self, event)

**onKeyUp**(self, event)

**onDbClick**(self, event)

**addSelectedRow**(self, row)

Marks a row as selected

**removeSelectedRow**(*self*, *row*)

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

**selectRow**(*self*, *newRow*)

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

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

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

**focusRow**(*self*, *row*)**getCurrentSelection**(*self*)

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

**clear**(*self*)

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

**removeRow**(*self*, *row*)

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

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

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

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

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

**selectAll**(*self*)

Selects all entries of the table.

**unselectAll**(*self*)

Unselects all entries of the table.

**invertSelection**(*self*)

Inverts the current selection on the whole table currently displayed.

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

Bases: flare.html5.Div

Provides kind of MVC on top of SelectTable.

**recalcHeight**(*self*, \**args*, \*\**kwargs*)**setDataProvider**(*self*, *obj*)

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

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

Ensure the table scrolls according to the position of its cursor

**getRowCount**(*self*)

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

**add**(*self*, *obj*)

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

**extend**(*self*, *objList*)

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

**testIfNextBatchNeededImmediately**(*self*)

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

**remove**(*self, objOrIndex*)

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

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

Flushes the whole table.

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

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

**rebuildTable**(*self*)

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

**setShownFields**(*self, fields*)

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

**onScroll**(*self, event*)

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

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

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

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

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

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

Re-emit the event.

**getCurrentSelection**(*self*)

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

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

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

**setCellRenders**(*self, renders*)

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

**activateSelection**(*self*)

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

**vi.widgets.task**

**Module Contents**

**Classes**

---

*TaskWidget*

---

continues on next page

Table 36 – continued from previous page

---

*ServerTaskWidget*

---

*TaskSelectWidget*

---

**class** `vi.widgets.task.TaskWidget`(*title*)Bases: `flare.popup.Popup`**class** `vi.widgets.task.ServerTaskWidget`(*title, key*)Bases: *TaskWidget***class** `vi.widgets.task.TaskSelectWidget`Bases: *TaskWidget***getSelectedTask**(*self*)**setActiveTask**(*self*)**onChange**(*self, event*)**invokeTask**(*self, \*args, \*\*kwargs*)**vi.widgets.tooltip**

## Module Contents

### Classes

---

*ToolTip*

---

Small utility class for providing tooltips

---

**class** `vi.widgets.tooltip.ToolTip`(*shortText="", longText="", \*args, \*\*kwargs*)Bases: `flare.html5.Div`

Small utility class for providing tooltips

**onClick**(*self, event*)**\_setDisabled**(*self, disabled*)**\_getDisabled**(*self*)**vi.widgets.topbar**

## Module Contents

### Classes

---

*TopBarWidget*

---

Provides the top-bar of VI

---

*UserState*

---

*Tasks*

---

continues on next page

---

Table 38 – continued from previous page

*Logout*


---

```

class vi.widgets.topbar.TopBarWidget
    Bases: flare.html5.Header

    Provides the top-bar of VI

    invoke(self)

    setTitle(self, title=None)

    onClick(self, event)

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

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

    onCurrentUserAvailable(self, req)

    update(self)

    static canHandle(action)

    onClick(self, sender=None)

    openEdit(self, key)

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

    onTaskListAvailable(self, req)

    onTaskListFailure(self)

    onCurrentUserAvailable(self, req)

    update(self)

    onClick(self, event)

    static canHandle(action)

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

    onClick(self, event)

    logout(self)

    static canHandle(action)

```

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

Base Widget that renders a tree.

continues on next page



Table 39 – continued from previous page

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

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

Bases: `flare.html5.Div`

Base Widget that renders a tree.

**nodeWidget**

**leafWidget**

**requestStructure**(*self*)

**receivedStructure**(*self*, *resp*)

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

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

**selectorReturn**(*self*)

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

**onKeyDown**(*self*, *event*)

**onKeyUp**(*self*, *event*)

**getActions**(*self*)

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

**clearSelection**(*self*)

Empties the current selection.

**extendSelection**(*self*, *element*)

Extends the current selection to element.

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

**activateSelection**(*self*, *element*)

Activates the current selection or element.

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

**requestChildren**(*self*, *element*)

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

Removes all currently visible elements and displays an error message

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

**onAttach**(*self*)

**onDetach**(*self*)

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

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

**onSetDefaultRootNode**(*self, req*)

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

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

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

**reloadData**(*self*)

Reload the data were displaying.

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

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

**onRequestSucceeded**(*self, req*)

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

**onDrop**(*self, event*)

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

**onDragOver**(*self, event*)

Allow dropping children on the rootNode

**getChildKey**(*self, widget*)

Order by sortindex

**static canHandle**(*moduleName, moduleInfo*)

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

Bases: flare.forms.widgets.tree.TreeLeafWidget

**setStyle**(*self*)

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

Bases: flare.forms.widgets.tree.TreeNodeWidget

**setStyle**(*self*)

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

Bases: flare.forms.widgets.tree.TreeNodeWidget

**setStyle**(*self*)

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

Bases: *TreeWidget*

Base Widget that renders a tree.

**leafWidget**

**nodeWidget**

**reloadData**(*self*)

Reload the data were displaying.

**rebuildPath**(*self*)

Rebuild the displayed path-list.

**onPathRequestSucceeded**(*self, req*)

Rebuild the displayed path-list according to request data

**activateSelection**(*self, element*)

Activates the current selection or element.

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

**static canHandle**(*module, moduleInfo*)

`vi.widgets.userlogoutmsg`

## Module Contents

### Classes

---

*UserLogoutMsg*

---

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

Bases: `flare.popup.Popup`

**pollInterval** = 120

**checkIntervall**

**visibilityChanged**(*self, e*)

**stopInterval**(*self*)

**hideMessage**(*self*)

Make this popup invisible

**showMessage**(*self*)

Show this popup

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

Return to the login window.

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

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

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

Start querying the server

**onUserTestSuccess**(*self, req*)

We received a response from the server

**onUserTestFail**(*self, text, ns*)

Error retrieving the current user response from the server

## Package Contents

### Classes

---

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

---

#### **class** vi.widgets.**Accordion**

Bases: flare.html5.Form

**\_segments** = []

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

**clearSegments**(self)

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

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

**Returns**

#### **class** vi.widgets.**TopBarWidget**

Bases: flare.html5.Header

Provides the top-bar of VI

**invoke**(*self*)

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

**onClick**(*self*, *event*)

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

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

Bases: `flare.html5.Div`

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

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

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

**selectorReturn**(*self*)

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

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

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

**setAmount**(*self*, *amount*)

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

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

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

**onClick**(*self*, *event*)

**setTableActionBar**(*self*)

**getDefaultEntryActions**(*self*)

Returns the list of actions available in our actionBar

**getActions**(*self*)

Returns the list of actions available in our actionBar

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

Returns the list of actions available in the action bar.

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

Removes all currently visible elements and displays an error message

**onNextBatchNeeded**(*self*)

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

**onAttach**(*self*)

**onDetach**(*self*)

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

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

**requestStructure**(*self*)

**receivedStructure**(*self*, *resp*)

**reloadData**(*self*)

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

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

Applies a new filter.

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

**getFilter**(*self*)

**updateEmptyNotification**(*self*)

**onCompletion**(*self, req*)

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

**setFields**(*self, fields*)

**getFields**(*self*)

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

**activateSelection**(*self*)

**static canHandle**(*moduleName, moduleInfo*)

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

Bases: flare.html5.Div

**appList** = list

**appHierarchy** = hierarchy

**appTree** = tree

**appSingleton** = singleton

**\_\_editIdx** = 0

**onDetach**(*self*)

**onAttach**(*self*)

**performLogics**(*self*)

**onChange**(*self, event*)

**onBoneChange**(*self, bone*)

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

Removes all currently visible elements and displays an error message

**reloadData**(*self*)

**save**(*self, data*)

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

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

**clear**(*self*)

Removes all visible bones/forms/fieldsets.

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

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

**cloneComplete**(*self, request*)

**formatReadFromClientErrorSeverity**(*self, error*)

**setData**(*self*, *request=None*, *data=None*, *ignoreMissing=False*, *askHierarchyCloning=True*)  
Rebuilds the UI according to the skeleton received from server

**Parameters**

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

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

**serializeForPost**(*self*, *validityCheck=False*)

**serializeForDocument**(*self*)

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

**class** `vi.widgets.ToolTip`(*shortText=""*, *longText=""*, *\*args*, *\*\*kwargs*)  
Bases: `flare.html5.Div`

Small utility class for providing tooltips

**onClick**(*self*, *event*)

**\_setDisabled**(*self*, *disabled*)

**\_getDisabled**(*self*)

**class** `vi.widgets.DataTable`(*\_loadOnDisplay=False*, *\*args*, *\*\*kwargs*)  
Bases: `flare.html5.Div`

Provides kind of MVC on top of SelectTable.

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

**setDataProvider**(*self*, *obj*)

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

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

Ensure the table scrolls according to the position of its cursor

**getRowCount**(*self*)

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

**add**(*self*, *obj*)

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

**extend**(*self*, *objList*)

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

**testIfNextBatchNeededImmediately**(*self*)

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

**remove**(*self*, *objOrIndex*)

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

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

Flushes the whole table.

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

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

**rebuildTable**(*self*)

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

**setShownFields**(*self, fields*)

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

**onScroll**(*self, event*)

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

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

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

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

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

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

Re-emit the event.

**getCurrentSelection**(*self*)

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

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

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

**setCellRenderers**(*self, renders*)

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

**activateSelection**(*self*)

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

**class** `vi.widgets.Search`(\*args, \*\*kwargs)

Bases: `flare.html5.Div`

**doSearch**(*self, \*args, \*\*kwargs*)

**resetSearch**(*self*)

**onKeyDown**(*self, event*)

**resetLoadingState**(*self*)

**reevaluate**(*self*)

**focus**(*self*)

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

Bases: `flare.html5.Div`

**onAttach**(*self*)

**onDetach**(*self*)

**setWidget**(*self, widget*)

**getWidget**(*self*)

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



```

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

    pollInterval = 120

    checkIntervall

    visibilityChanged(self, e)

    stopInterval(self)

    hideMessage(self)
        Make this popup invisible

    showMessage(self)
        Show this popup

    showLoginWindow(self, *args, **kwargs)
        Return to the login window.

    checkForSuspendResume(self, *args, **kwargs)
        Test if at least self.pollIntervall seconds have passed and query the server if

    startPolling(self, *args, **kwargs)
        Start querying the server

    onUserTestSuccess(self, req)
        We received a response from the server

    onUserTestFail(self, text, ns)
        Error retrieving the current user response from the server

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

class vi.widgets.TaskSelectWidget
    Bases: TaskWidget

    getSelectedTask(self)

    setActiveTask(self)

    onChange(self, event)

    invokeTask(self, *args, **kwargs)

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

    Bases: flare.html5.Div

    renderStructure(self, readOnly=False)

    serializeForPost(self, validityCheck=False)

    serializeForDocument(self)

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

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

    onChange(self, event)

    onKeyDown(self, event)

```

```
performLogics(self)  
class vi.widgets.HtmlEditor(*args, **kwargs)  
    Bases: flare.html5.Textarea  
    initSources = False  
    _attachSummernote(self, retry=0)  
    onAttach(self)  
    onDetach(self)  
    onEditorChange(self, e, *args, **kwargs)  
    _getValue(self)  
    _setValue(self, val)  
    enable(self)  
    disable(self)  
class vi.widgets.ExportCsv(widget, selection, encoding=None, language=None, separator=None,  
    lineSeparator=None, *args, **kwargs)  
    Bases: flare.html5.Progress  
    nextChunk(self, cursor=None)  
    nextChunkComplete(self, req)  
    exportToFile(self)  
    nextChunkFailure(self, req, code)  
    replaceWithMessage(self, message, logClass='success')  
class vi.widgets.ExportCsvStarter(widget, *args, **kwargs)  
    Bases: flare.popup.Popup  
    onExportBtnClick(self, *args, **kwargs)  
class vi.widgets.TreeWidget(module, rootNode=None, node=None, context=None, *args, **kwargs)  
    Bases: flare.html5.Div  
    Base Widget that renders a tree.  
    nodeWidget  
    leafWidget  
    requestStructure(self)  
    receivedStructure(self, resp)  
    setSelector(self, callback, multi=True, allow=None)  
        Configures the widget as selector for a relationalBone and shows it.  
    selectorReturn(self)  
        Returns the current selection to the callback configured with setSelector.  
    onKeyDown(self, event)  
    onKeyUp(self, event)  
    getActions(self)  
        Returns a list of actions that are being set for the ActionBar. Override this to provide additional actions.
```

**clearSelection**(*self*)

Empties the current selection.

**extendSelection**(*self*, *element*)

Extends the current selection to element.

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

**activateSelection**(*self*, *element*)

Activates the current selection or element.

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

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

Removes all currently visible elements and displays an error message

**onDataChanged**(*self*, *module*, *\*\*kwargs*)**onAttach**(*self*)**onDetach**(*self*)**itemForKey**(*self*, *key*, *elem=None*)

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

**onSetDefaultRootNode**(*self*, *req*)

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

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

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

**reloadData**(*self*)

Reload the data were displaying.

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

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

**onRequestSucceeded**(*self*, *req*)

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

**onDrop**(*self*, *event*)

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

**onDragOver**(*self*, *event*)

Allow dropping children on the rootNode

**getChildKey**(*self*, *widget*)

Order by sortindex

**static canHandle**(*moduleName*, *moduleInfo*)

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

Bases: [TreeWidget](#)

Base Widget that renders a tree.

**leafWidget**

**nodeWidget**

**reloadData**(*self*)

Reload the data were displaying.

**rebuildPath**(*self*)

Rebuild the displayed path-list.

**onPathRequestSucceeded**(*self, req*)

Rebuild the displayed path-list according to request data

**activateSelection**(*self, element*)

Activates the current selection or element.

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

**static canHandle**(*module, moduleInfo*)

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

Bases: `vi.widgets.tree.TreeWidget`

A Hierarchy is a Tree without leaf distinction!

**leafWidget**

**reloadData**(*self*)

Reload the data were displaying.

**reloadListWidget**(*self*)

**toggleListView**(*self*)

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

**showListView**(*self*)

**hideListView**(*self*)

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

**static canHandle**(*moduleName, moduleInfo*)

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

Bases: `vi.widgets.tree.TreeBrowserWidget`

Base Widget that renders a tree.

**leafWidget**

**nodeWidget**

**searchWidget**(*self*)

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

**getChildKey**(*self, widget*)

Derives a string used to sort the entries on each level

**onDrop**(*self, event*)

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

**static canHandle**(*module, moduleInfo*)

## Submodules

`vi.admin`

## Module Contents

### Classes

---

*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**(*self*, *event*)

**reset**(*self*)

**invoke**(*self*)

Is called to show the screen

**getCurrentUser**(*self*)

**getCurrentUserSuccess**(*self*, *req*)

**getCurrentUserFailure**(*self*, *req*, *code*)

**startup**(*self*)

**initializeViews**(*self*)

**initializeConfig**(*self*)

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

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

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

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

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

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

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

Performs an execution call.

### Parameters

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

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

We dont stack widgets anymore. We use now Popups.

**removeWidget**(*self, widget*)

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

**isFullscreen**(*self*)

**onError**(*self, req, code*)

`vi.admin.viInitializedEvent`

`vi.config`

### Module Contents

#### Functions

---

*updateConf*(*\_conf*)

---

*getConf*()

---

#### Attributes

---

*vi\_conf*

---

*conf*

---

`vi.config.vi_conf`

`vi.config.updateConf(_conf)`

`vi.config.getConf()`

`vi.config.conf`

**vi.exception****Module Contents****exception** `vi.exception.InvalidBoneValueException`Bases: `ValueError`

Inappropriate argument value (of correct type).

**vi.log****Module Contents****Classes**

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

**Attributes**

<code>iddbTableName</code>
----------------------------

`vi.log.iddbTableName = vi_log3`**class** `vi.log.logEntry(logObj=None)`Bases: `flare.html5.Span`

PopOut Elements

**class** `vi.log.logA(logObj=None)`Bases: `flare.html5.A`

click handler for loglist

**onClick**(*self*, *sender=None*)**openEditor**(*self*, *key*)**class** `vi.log.logWidget(logList)`Bases: `flare.html5.Div`**buildDataTable**(*self*)**class** `vi.log.LogButton`Bases: `flare.html5.Div`**idbdata**(*self*, *event*)

```
cleanLog(self)  
cleanLogAction(self, event)  
renderPopOut(self)  
onClick(self, sender=None)  
openLog(self)
```

```
    apane = Pane( translate("Log"), closeable=True, iconClasses=[ "apptype_list"], collapseable=True  
    )  
    wg = logWidget(self.logsList )  
    apane.addWidget(wg)  
    conf["mainWindow"].addPane(apane) conf["mainWindow"].focusPane(apane)  
log(self, type, msg, icon=None, modul=None, action=None, key=None, data=None, date=None,  
    onlyLoad=False)  
msgOverlay(self, logObj)  
removeInfo(self, wrap)  
reset(self)  
static canHandle(action)
```

```
class vi.log.Log
```

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

```
    Provides the "messaging" center displayed at the bottom of VI
```

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

```
    log(self, type, msg, icon=None, date=None)
```

```
        Adds a message to the log :param type: The type of the message. :type type: "success", "error", "warning",  
        "info", "progress" :param msg: The message to append :type msg: str
```

```
    removeNewCls(self, span)
```

```
    reset(self)
```

```
vi.login
```

## Module Contents

### Classes

---

*LoginInputField*

---

*BaseLoginHandler*

---

*UserPasswordLoginHandler*

---

continues on next page



Table 48 – continued from previous page

<i>GoogleAccountLoginHandler</i>	
<i>LoginScreen</i>	This is the screen superclass.
<pre> <b>class</b> vi.login.LoginInputField(notifier, *args, **kwargs)     Bases: flare.html5.Input         <b>onKeyPress</b>(self, event) <b>class</b> vi.login.BaseLoginHandler(loginScreen, *args, **kwargs)     Bases: flare.html5.Li         <b>onClick</b>(self, event)         <b>enable</b>(self)         <b>disable</b>(self)         <b>lock</b>(self)         <b>unlock</b>(self)         <b>login</b>(self)         <b>reset</b>(self)         <b>parseAnswer</b>(self, req) <b>class</b> vi.login.UserPasswordLoginHandler(loginScreen, *args, **kwargs)     Bases: <i>BaseLoginHandler</i>         <b>cssname</b> = userpassword         <b>onKeyPress</b>(self, event)         <b>onLoginClick</b>(self, sender=None)         <b>doLoginSuccess</b>(self, req)         <b>doLoginFailure</b>(self, req, code, *args, **kwargs)         <b>onVerifyClick</b>(self, sender=None)         <b>doVerifySuccess</b>(self, req)         <b>doVerifyFailure</b>(self, *args, **kwargs)         <b>onSendClick</b>(self, sender=None)         <b>reset</b>(self)         <b>enable</b>(self)         <b>focusLaterIdiot</b>(self)         <b>static canHandle</b>(method, secondFactor) <b>class</b> vi.login.GoogleAccountLoginHandler(loginScreen, *args, **kwargs)     Bases: <i>BaseLoginHandler</i>         <b>cssname</b> = googleaccount         <b>onLoginClick</b>(self, sender=None)         <b>static canHandle</b>(method, secondFactor) </pre>	

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

Bases: `vi.screen.Screen`

This is the screen superclass.

It represents a basic screen and its functionality.

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

Is called to show the screen

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

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

**insufficientRights**(`self`)

**doSkipLogin**(`self, req`)

**onGetAuthMethodsSuccess**(`self, req`)

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

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

**redirectNoAdmin**(`self`)

`vi.pane`

## Module Contents

### Classes

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

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

Bases: `flare.html5.Div`

Base class for Panes.

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

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

**\_\_setattr\_\_**(`self, key, value`)

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

**lock**(`self`)

**unlock**(`self`)

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

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

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

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

**removeChildPane**(*self*, *pane*)

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

**onDetach**(*self*)

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

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

**rebuildChildrenClassInfo**(*self*)

**removeWidget**(*self*, *widget*)

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

**containsWidget**(*self*, *widget*)

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

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

**expand**(*self*)

**collapse**(*self*)

**focus**(*self*)

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

Bases: `Pane`

This pane groups subpanes; it cannot have direct childrens

**loadChildren**(*self*)

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

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

**expand**(*self*)

**collapse**(*self*)

**onFocus**(*self*, *event*)

`vi.priorityqueue`

## Module Contents

### Classes

---

`StartupQueue`

---

## Attributes

---

*startupQueue*

---

*HandlerClassSelector*

---

*actionDelegateSelector*

---

*initialHashHandler*

---

*extendedSearchWidgetSelector*

---

*toplevelActionSelector*

---

*loginHandlerSelector*

---

*protocolWrapperClassSelector*

---

*protocolWrapperInstanceSelector*

---

**class** `vi.priorityqueue.StartupQueue`

Bases: `object`

**reset**(*self*)

**setFinalElem**(*self*, *elem*)

**insertElem**(*self*, *priority*, *elem*)

**run**(*self*)

**next**(*self*)

`vi.priorityqueue.startupQueue`

`vi.priorityqueue.HandlerClassSelector`

`vi.priorityqueue.actionDelegateSelector`

`vi.priorityqueue.initialHashHandler`

`vi.priorityqueue.extendedSearchWidgetSelector`

`vi.priorityqueue.toplevelActionSelector`

`vi.priorityqueue.loginHandlerSelector`

`vi.priorityqueue.protocolWrapperClassSelector`

`vi.priorityqueue.protocolWrapperInstanceSelector`

vi.screen

## Module Contents

### Classes

---

*Screen*

This is the screen superclass.

---

```

class vi.screen.Screen(*args, **kwargs)
    Bases: flare.html5.Div

    This is the screen superclass.

    It represents a basic screen and its functionality.

    lock(self)

    unlock(self)

    invoke(self)
        Is called to show the screen

    remove(self)
        Remove the screen from its parent

    setTitle(self, title=None)

```

vi.serversideaction

## Module Contents

### Classes

---

*ServerSideActionWdg*

---

```

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

    switchDisabledState(self, disabled)

    onAttach(self)

    onDetach(self)

    onSelectionChanged(self, table, selection)

    onClick(self, sender=None)

    fetchNext(self)

    fetchSucceeded(self, req)

    fetchFailed(self, req, code)

    resetLoadingState(self)

```

vi.utils

## Module Contents

### Classes

---

*indexeddbConnector*

---

*indexeddb*

---

### Functions

---

*formatString*(format, data, structure=None, lan- Parses a string given by format and substitutes place-  
guage=None) holders using values specified by data.

---

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

---

*setPreventUnloading*(mode=True)

---

*mergeDict*(original, target)

---

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

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

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

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

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

**dbResult**

**dbTransaction**

**connect**(self)

**db\_error**(self, event)

**db\_blocked**(self, events)

**db\_version**(self, event)

**db\_onupgradeneeded**(self, event)

**db\_success**(self, event)

**class** vi.utils.**indexeddb**(dbName, dbVersion=None)

**queue** = []

**dbqueue** = []

**connect**(self)

**getList**(self, name)

```

    _getList(self, event)
    getListKeys(self, name)
    _getListKey(self, event)
    db_success(self, event)
    dbAction(self, action, name, key=None, obj=None)
    _processDbUpdate(self, event)
    _processQueue(self, event)
    _writeToStore(self, item, dbResult, dbTransaction)
    _deleteFromStore(self, item, dbResult, dbTransaction)
    _updateToStore(self, item, dbResult, dbTransaction)
    _deleteObjectStore(self, item, dbResult, dbTransaction)
    _registerObjectStore(self, item, dbResult, dbTransaction)

```

```
vi.utils.mergeDict(original, target)
```

## Package Contents

### Classes

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

### Functions

<i>start()</i>
----------------

### Attributes

<i>vi_conf</i>
<i>conf</i>
<i>s</i>
<i>a</i>
<i>d</i>

continues on next page

Table 58 – continued from previous page

---

*sc*

---

*scinv*

---

*s*

---

**vi.vi\_conf****class** *vi.LoginScreen*(\*args, \*\*kwargs)Bases: *vi.screen.Screen*

This is the screen superclass.

It represents a basic screen and its functionality.

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

Is called to show the screen

**onLogoutSuccess**(*self*, \*args, \*\*kwargs)**doShowLogin**(*self*, *req*, *code*, \*args, \*\*kwargs)**insufficientRights**(*self*)**doSkipLogin**(*self*, *req*)**onGetAuthMethodsSuccess**(*self*, *req*)**selectHandler**(*self*, *handler=None*)**onGetAuthMethodsFailure**(*self*, \*args, \*\*kwargs)**redirectNoAdmin**(*self*)**class** *vi.AdminScreen*(\*args, \*\*kwargs)Bases: *vi.screen.Screen*

This is the screen superclass.

It represents a basic screen and its functionality.

**onClick**(*self*, *event*)**reset**(*self*)**invoke**(*self*)

Is called to show the screen

**getCurrentUser**(*self*)**getCurrentUserSuccess**(*self*, *req*)**getCurrentUserFailure**(*self*, *req*, *code*)**startup**(*self*)**initializeViews**(*self*)**initializeConfig**(*self*)**appendNavList**(*self*, *NavList*, *target*, *parentInfo=()*)**openView**(*self*, *name*, *icon*, *viewName*, *moduleName*, *actionName*, *data*, *focusView=True*, *append=False*, *target='mainNav'*)



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

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

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

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

**execCall**(*self, path, params=None*)  
Performs an execution call.

#### Parameters

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

**stackWidget**(*self, widget, disableOtherWidgets=True*)  
We dont stack widgets anymore. We use now Popups.

**removeWidget**(*self, widget*)

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

**isFullscreen**(*self*)

**onError**(*self, req, code*)

**vi.conf**

**class vi.Application**

Bases: flare.html5.Div

**startup**(*self, \*args, \*\*kwargs*)

**getVersionSuccess**(*self, req*)

**getConfigSuccess**(*self, req*)

**startupFailure**(*self, req, err*)

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

**admin**(*self*)

**logout**(*self*)

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

**setPath**(*self, path=""*)

**vi.start()**

**vi.s**

**vi.a**

**vi.d**

**vi.sc**

**vi.scinv**

**vi.s**



## 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, 15
- vi.actions.tree, 16
- vi.admin, 57
- vi.config, 58
- vi.exception, 59
- vi.framework, 18
- vi.framework.components, 18
- vi.framework.components.actionbar, 18
- vi.framework.components.datatable, 18
- vi.log, 59
- vi.login, 60
- vi.pane, 62
- vi.priorityqueue, 63
- vi.screen, 65
- vi.serversideaction, 65
- vi.sidebarwidgets, 22
- vi.sidebarwidgets.filterselector, 22
- vi.sidebarwidgets.internalpreview, 22
- vi.translations, 23
- vi.translations.de, 23
- vi.translations.en, 23
- vi.utils, 66
- vi.views, 23
- vi.views.edit, 23
- vi.views.hierarchy, 24
- vi.views.list, 24
- vi.views.log, 25
- vi.views.notfound, 25
- vi.views.overview, 25
- vi.views.singleton, 26
- vi.views.tree, 26
- vi.widgets, 27
- vi.widgets.accordion, 27
- vi.widgets.appnavigation, 28
- vi.widgets.csvexport, 29
- vi.widgets.edit, 30
- vi.widgets.file, 32
- vi.widgets.hierarchy, 34
- vi.widgets.htmleditor, 34
- vi.widgets.internaedit, 35
- vi.widgets.list, 36
- vi.widgets.preview, 38
- vi.widgets.repeatdate, 38
- vi.widgets.search, 39
- vi.widgets.sidebar, 39
- vi.widgets.table, 40
- vi.widgets.task, 42
- vi.widgets.tooltip, 43
- vi.widgets.topbar, 43
- vi.widgets.tree, 44
- vi.widgets.userlogoutmsg, 47



## Symbols

- `__editIdx_` (*vi.widgets.EditWidget* attribute), 50
  - `__editIdx_` (*vi.widgets.edit.EditWidget* attribute), 31
  - `__editIdx_` (*vi.widgets.repeatdate.RepeatDatePopup* attribute), 38
  - `__setattr__` (*vi.pane.Pane* method), 62
  - `_attachSummernote` (*vi.widgets.HtmlEditor* method), 54
  - `_attachSummernote` (*vi.widgets.htmleditor.HtmlEditor* method), 35
  - `_deleteFromStore` (*vi.utils.indexeddb* method), 67
  - `_deleteObjectStore` (*vi.utils.indexeddb* method), 67
  - `_extraCols` (*vi.framework.components.datatable.SelectTable* method), 20
  - `_extraCols` (*vi.widgets.table.SelectTable* method), 41
  - `_getDisabled` (*vi.widgets.ToolTip* method), 51
  - `_getDisabled` (*vi.widgets.tooltip.ToolTip* method), 43
  - `_getList` (*vi.utils.indexeddb* method), 67
  - `_getListKey` (*vi.utils.indexeddb* method), 67
  - `_getValue` (*vi.widgets.HtmlEditor* method), 54
  - `_getValue` (*vi.widgets.htmleditor.HtmlEditor* method), 35
  - `_processDbUpdate` (*vi.utils.indexeddb* method), 67
  - `_processQueue` (*vi.utils.indexeddb* method), 67
  - `_registerObjectStore` (*vi.utils.indexeddb* method), 67
  - `_renderObject` (*vi.framework.components.datatable.DataTable* method), 20
  - `_renderObject` (*vi.framework.components.datatable.ViewportDataTable* method), 21
  - `_renderObject` (*vi.widgets.DataTable* method), 51
  - `_renderObject` (*vi.widgets.table.DataTable* method), 42
  - `_rowForEvent` (*vi.framework.components.datatable.SelectTable* method), 19
  - `_rowForEvent` (*vi.widgets.table.SelectTable* method), 40
  - `_segments` (*vi.widgets.Accordion* attribute), 48
  - `_segments` (*vi.widgets.accordion.Accordion* attribute), 27
  - `_setDisabled` (*vi.widgets.ToolTip* method), 51
  - `_setDisabled` (*vi.widgets.tooltip.ToolTip* method), 43
  - `_setPage` (*vi.widgets.list.ViewportListWidget* method), 37
  - `_setValue` (*vi.widgets.HtmlEditor* method), 54
  - `_setValue` (*vi.widgets.appnavigation.NavigationSeperator* method), 29
  - `_setValue` (*vi.widgets.htmleditor.HtmlEditor* method), 35
  - `_updateToStore` (*vi.utils.indexeddb* method), 67
  - `_writeToStore` (*vi.utils.indexeddb* method), 67
- ## A
- `a` (in module *vi*), 69
  - `Accordion` (class in *vi.widgets*), 48
  - `Accordion` (class in *vi.widgets.accordion*), 27
  - `AccordionSegment` (class in *vi.widgets.accordion*), 27
  - `ActionBar` (class in *vi.framework.components.actionbar*), 18
  - `actionDelegateSelector` (in module *vi.priorityqueue*), 64
  - `activate` (*vi.widgets.accordion.AccordionSegment* method), 27
  - `activateCurrentSelection` (*vi.framework.components.datatable.DataTable* method), 21
  - `activateSelection` (*vi.widgets.DataTable* method), 52
  - `activateSelection` (*vi.widgets.list.ListWidget* method), 37
  - `activateSelection` (*vi.widgets.ListWidget* method), 50
  - `activateSelection` (*vi.widgets.table.DataTable* method), 42
  - `activateSelection` (*vi.widgets.tree.TreeBrowserWidget* method), 46
  - `activateSelection` (*vi.widgets.tree.TreeWidget* method), 45
  - `activateSelection` (*vi.widgets.TreeBrowserWidget* method), 56
  - `activateSelection` (*vi.widgets.TreeWidget* method), 55

- add() (*vi.framework.components.datatable.DataTable method*), 20
- add() (*vi.framework.components.datatable.ViewportDataTable method*), 21
- add() (*vi.widgets.DataTable method*), 51
- add() (*vi.widgets.table.DataTable method*), 41
- AddAction (*class in vi.actions.hierarchy*), 8
- AddAction (*class in vi.actions.list*), 10
- addChildPane() (*vi.pane.Pane method*), 62
- AddLeafAction (*class in vi.actions.file*), 7
- AddLeafAction (*class in vi.actions.tree*), 16
- addNavigationBlock() (*vi.widgets.appnavigation.AppNavigation method*), 29
- addNavigationPoint() (*vi.widgets.appnavigation.AppNavigation method*), 29
- addNavigationPointAfter() (*vi.widgets.appnavigation.AppNavigation method*), 29
- AddNodeAction (*class in vi.actions.file*), 6
- AddNodeAction (*class in vi.actions.tree*), 16
- addSegment() (*vi.widgets.Accordion method*), 48
- addSegment() (*vi.widgets.accordion.Accordion method*), 27
- addSelectedRow() (*vi.framework.components.datatable.SelectTable method*), 19
- addSelectedRow() (*vi.widgets.table.SelectTable method*), 40
- addSeperator() (*vi.widgets.appnavigation.Navigationblock method*), 29
- addWidget() (*vi.pane.Pane method*), 63
- addWidget() (*vi.widgets.accordion.AccordionSegment method*), 27
- admin() (*vi.Application method*), 69
- AdminScreen (*class in vi*), 68
- AdminScreen (*class in vi.admin*), 57
- allDeletedSuccess() (*vi.actions.hierarchy.DeleteAction method*), 9
- allDeletedSuccess() (*vi.actions.list.DeleteAction method*), 11
- allDeletedSuccess() (*vi.actions.tree.DeleteAction method*), 17
- appendNavList() (*vi.admin.AdminScreen method*), 57
- appendNavList() (*vi.AdminScreen method*), 68
- appendSubChild() (*vi.widgets.appnavigation.NavigationElement method*), 29
- appHierarchy (*vi.widgets.edit.EditWidget attribute*), 30
- appHierarchy (*vi.widgets.EditWidget attribute*), 50
- Application (*class in vi*), 69
- appList (*vi.widgets.edit.EditWidget attribute*), 30
- appList (*vi.widgets.EditWidget attribute*), 50
- applyFilter() (*vi.sidebarwidgets.filtersselector.FilterSelector method*), 22
- AppNavigation (*class in vi.widgets.appnavigation*), 29
- appSingleton (*vi.widgets.edit.EditWidget attribute*), 31
- appSingleton (*vi.widgets.EditWidget attribute*), 50
- appTree (*vi.widgets.edit.EditWidget attribute*), 30
- appTree (*vi.widgets.EditWidget attribute*), 50
- ArrowAction() (*vi.widgets.appnavigation.NavigationElement method*), 29
- ## B
- BaseLoginHandler (*class in vi.login*), 61
- BreadcrumbNodeWidget (*class in vi.widgets.tree*), 46
- BrowserLeafWidget (*class in vi.widgets.tree*), 46
- BrowserNodeWidget (*class in vi.widgets.tree*), 46
- buildAccordion() (*vi.widgets.Accordion method*), 48
- buildAccordion() (*vi.widgets.accordion.Accordion method*), 27
- buildDataTable() (*vi.log.logWidget method*), 59
- buildSeperator() (*vi.widgets.appnavigation.NavigationSeperator method*), 29
- ## C
- CancelClose (*class in vi.actions.edit*), 6
- canHandle() (*vi.log.LogButton static method*), 60
- canHandle() (*vi.login.GoogleAccountLoginHandler static method*), 61
- canHandle() (*vi.login.UserPasswordLoginHandler static method*), 61
- canHandle() (*vi.views.hierarchy.hierarchyHandler static method*), 24
- canHandle() (*vi.views.list.listHandler static method*), 24
- canHandle() (*vi.views.singleton.singletonHandler static method*), 26
- canHandle() (*vi.views.tree.treeHandler static method*), 27
- canHandle() (*vi.widgets.file.FileWidget static method*), 33
- canHandle() (*vi.widgets.FileWidget static method*), 56
- canHandle() (*vi.widgets.hierarchy.HierarchyWidget static method*), 34
- canHandle() (*vi.widgets.HierarchyWidget static method*), 56
- canHandle() (*vi.widgets.list.ListWidget static method*), 37
- canHandle() (*vi.widgets.list.ViewportListWidget static method*), 38
- canHandle() (*vi.widgets.ListWidget static method*), 50
- canHandle() (*vi.widgets.topbar.Logout static method*), 44
- canHandle() (*vi.widgets.topbar.Tasks static method*), 44
- canHandle() (*vi.widgets.topbar.UserState static method*), 44
- canHandle() (*vi.widgets.tree.TreeBrowserWidget static method*), 47

- canHandle() (vi.widgets.tree.TreeWidget static method), 46  
 canHandle() (vi.widgets.TreeBrowserWidget static method), 56  
 canHandle() (vi.widgets.TreeWidget static method), 55  
 checkErrors() (in module vi.widgets.internaledit), 35  
 checkForSuspendResume() (vi.widgets.UserLogoutMsg method), 53  
 checkForSuspendResume() (vi.widgets.userlogoutmsg.UserLogoutMsg method), 47  
 checkInitialHash() (vi.admin.AdminScreen method), 57  
 checkInitialHash() (vi.AdminScreen method), 69  
 checkIntervall (vi.widgets.UserLogoutMsg attribute), 53  
 checkIntervall (vi.widgets.userlogoutmsg.UserLogoutMsg attribute), 47  
 checkVisibility() (vi.widgets.accordion.AccordionSegment method), 27  
 cleanLog() (vi.log.LogButton method), 59  
 cleanLogAction() (vi.log.LogButton method), 60  
 clear() (vi.framework.components.datatable.DataTable method), 20  
 clear() (vi.framework.components.datatable.SelectTable method), 20  
 clear() (vi.framework.components.datatable.ViewportDataTable method), 21  
 clear() (vi.widgets.DataTable method), 51  
 clear() (vi.widgets.edit.EditWidget method), 31  
 clear() (vi.widgets.EditWidget method), 50  
 clear() (vi.widgets.repeatdate.RepeatDatePopup method), 38  
 clear() (vi.widgets.table.DataTable method), 42  
 clear() (vi.widgets.table.SelectTable method), 41  
 clearSegments() (vi.widgets.Accordion method), 48  
 clearSegments() (vi.widgets.accordion.Accordion method), 27  
 clearSelection() (vi.widgets.tree.TreeWidget method), 45  
 clearSelection() (vi.widgets.TreeWidget method), 54  
 CloneAction (class in vi.actions.hierarchy), 8  
 CloneAction (class in vi.actions.list), 11  
 cloneComplete() (vi.widgets.edit.EditWidget method), 31  
 cloneComplete() (vi.widgets.EditWidget method), 50  
 close() (vi.widgets.SideBar method), 52  
 close() (vi.widgets.sidebar.SideBar method), 39  
 CloseAction (class in vi.actions.list), 12  
 closeMessage() (vi.widgets.file.MultiUploader method), 33  
 closeOrContinue() (vi.widgets.edit.EditWidget method), 31  
 closeOrContinue() (vi.widgets.EditWidget method), 50  
 collapse() (vi.pane.GroupPane method), 63  
 collapse() (vi.pane.Pane method), 63  
 CompoundFilter (class in vi.sidebarwidgets.filtersselector), 22  
 conf (in module vi), 69  
 conf (in module vi.config), 58  
 connect() (vi.utils.indexeddb method), 66  
 connect() (vi.utils.indexeddbConnector method), 66  
 containsWidget() (vi.pane.Pane method), 63  
 ContextAction (class in vi.actions.context), 5  
 createDir() (vi.actions.file.AddNodeAction method), 7  
 CreateRecurrentAction (class in vi.actions.list), 14  
 cssname (vi.login.GoogleAccountLoginHandler attribute), 61  
 cssname (vi.login.UserPasswordLoginHandler attribute), 61
- ## D
- DataTable (class in vi.framework.components.datatable), 20  
 DataTable (class in vi.widgets), 51  
 DataTable (class in vi.widgets.table), 41  
 db\_blocked() (vi.utils.indexeddbConnector method), 66  
 db\_error() (vi.utils.indexeddbConnector method), 66  
 db\_onupgradeneeded() (vi.utils.indexeddbConnector method), 66  
 db\_success() (vi.utils.indexeddb method), 67  
 db\_success() (vi.utils.indexeddbConnector method), 66  
 db\_version() (vi.utils.indexeddbConnector method), 66  
 dbAction() (vi.utils.indexeddb method), 67  
 dbqueue (vi.utils.indexeddb attribute), 66  
 dbResult (vi.utils.indexeddbConnector attribute), 66  
 dbTransaction (vi.utils.indexeddbConnector attribute), 66  
 deactivate() (vi.widgets.accordion.AccordionSegment method), 27  
 DeferredLoadChildren() (vi.pane.GroupPane method), 63  
 DeleteAction (class in vi.actions.hierarchy), 9  
 DeleteAction (class in vi.actions.list), 11  
 DeleteAction (class in vi.actions.tree), 17  
 deletedFailed() (vi.actions.list.DeleteAction method), 11  
 deletedSuccess() (vi.actions.list.DeleteAction method), 11  
 disable() (vi.login.BaseLoginHandler method), 61  
 disable() (vi.widgets.HtmlEditor method), 54  
 disable() (vi.widgets.htmleditor.HtmlEditor method), 35  
 disableViUnloadingWarning() (vi.actions.file.DownloadAction method), 7



- doApply() (*vi.actions.list.SelectFieldsPopup* method), 12
- doCancel() (*vi.actions.list.SelectFieldsPopup* method), 12
- doCloneHierarchy() (*vi.widgets.edit.EditWidget* method), 31
- doCloneHierarchy() (*vi.widgets.EditWidget* method), 50
- doClose() (*vi.widgets.preview.Preview* method), 38
- doDelete() (*vi.actions.hierarchy.DeleteAction* method), 9
- doDelete() (*vi.actions.list.DeleteAction* method), 11
- doDelete() (*vi.actions.tree.DeleteAction* method), 17
- doDownload() (*vi.actions.file.DownloadAction* method), 7
- doInvertSelection() (*vi.actions.list.SelectFieldsPopup* method), 12
- doLoginFailure() (*vi.login.UserPasswordLoginHandler* method), 61
- doLoginSuccess() (*vi.login.UserPasswordLoginHandler* method), 61
- doMarkPayed() (*vi.actions.list\_order.ShopMarkAction* method), 15
- doSave() (*vi.widgets.edit.EditWidget* method), 31
- doSave() (*vi.widgets.EditWidget* method), 51
- doSave() (*vi.widgets.InternalEdit* method), 53
- doSave() (*vi.widgets.internaledit.InternalEdit* method), 36
- doSave() (*vi.widgets.repeatdate.RepeatDatePopup* method), 39
- doSearch() (*vi.widgets.Search* method), 52
- doSearch() (*vi.widgets.search.Search* method), 39
- doSelectAll() (*vi.actions.list.SelectFieldsPopup* method), 12
- doSetFields() (*vi.actions.list.SelectFieldsPopup* method), 12
- doShowLogin() (*vi.login.LoginScreen* method), 62
- doShowLogin() (*vi.LoginScreen* method), 68
- doSkipLogin() (*vi.login.LoginScreen* method), 62
- doSkipLogin() (*vi.LoginScreen* method), 68
- doUnselectAll() (*vi.actions.list.SelectFieldsPopup* method), 12
- doVerifyFailure() (*vi.login.UserPasswordLoginHandler* method), 61
- doVerifySuccess() (*vi.login.UserPasswordLoginHandler* method), 61
- download() (*vi.widgets.file.FilePreviewImage* method), 32
- DownloadAction (*class in vi.actions.file*), 7
- dropTableContent() (*vi.framework.components.datatable.SelectTable* method), 20
- 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*), 16
- editDir() (*vi.actions.file.EditAction* method), 7
- editHandler (*class in vi.views.edit*), 23
- editHandlerWidget (*class in vi.views.edit*), 23
- EditWidget (*class in vi.widgets*), 50
- EditWidget (*class in vi.widgets.edit*), 30
- enable() (*vi.login.BaseLoginHandler* method), 61
- enable() (*vi.login.UserPasswordLoginHandler* method), 61
- enable() (*vi.widgets.HtmlEditor* method), 54
- enable() (*vi.widgets.htmleditor.HtmlEditor* method), 35
- enableViUnloadingWarning() (*vi.actions.file.DownloadAction* method), 7
- EntryIcon() (*vi.widgets.file.FileLeafWidget* method), 33
- execCall() (*vi.admin.AdminScreen* method), 57
- execCall() (*vi.AdminScreen* method), 69
- ExecuteSingleton (*class in vi.actions.edit*), 5
- expand() (*vi.pane.GroupPane* method), 63
- expand() (*vi.pane.Pane* method), 63
- ExportCsv (*class in vi.widgets*), 54
- ExportCsv (*class in vi.widgets.csvexport*), 29
- ExportCsvAction (*class in vi.actions.list*), 14
- ExportCsvStarter (*class in vi.widgets*), 54
- ExportCsvStarter (*class in vi.widgets.csvexport*), 30
- exportToFile() (*vi.widgets.csvexport.ExportCsv* method), 29
- exportToFile() (*vi.widgets.ExportCsv* method), 54
- extend() (*vi.framework.components.datatable.DataTable* method), 20
- extend() (*vi.framework.components.datatable.ViewportDataTable* method), 21
- extend() (*vi.widgets.DataTable* method), 51
- extend() (*vi.widgets.table.DataTable* method), 41
- extendedSearchWidgetSelector (*in module vi.priorityqueue*), 64
- extendSelection() (*vi.widgets.tree.TreeWidget* method), 45
- extendSelection() (*vi.widgets.TreeWidget* method), 55
- F**
- fetchFailed() (*vi.serversideaction.ServerSideActionWdg* method), 65
- fetchNext() (*vi.serversideaction.ServerSideActionWdg* method), 65
- fetchSucceeded() (*vi.serversideaction.ServerSideActionWdg* method), 65
- FileImagePopup (*class in vi.widgets.file*), 32
- FileLeafWidget (*class in vi.widgets.file*), 33



- FileNodeWidget (class in *vi.widgets.file*), 33  
 FilePreviewImage (class in *vi.widgets.file*), 32  
 FileSelectUploader (class in *vi.actions.file*), 6  
 FileWidget (class in *vi.widgets*), 56  
 FileWidget (class in *vi.widgets.file*), 33  
 FilterSelector (class in *vi.sidebarwidgets.filterselector*), 22  
 findText() (*vi.actions.list.PageFindAction* method), 14  
 focus() (*vi.pane.Pane* method), 63  
 focus() (*vi.sidebarwidgets.filterselector.CompoundFilter* method), 22  
 focus() (*vi.widgets.Search* method), 52  
 focus() (*vi.widgets.search.Search* method), 39  
 focusLaterIdiot() (*vi.login.UserPasswordLoginHandler* method), 61  
 focusRow() (*vi.framework.components.datatable.SelectTable* method), 19  
 focusRow() (*vi.widgets.table.SelectTable* method), 41  
 formatReadFromClientErrorSeverity() (*vi.widgets.edit.EditWidget* method), 31  
 formatReadFromClientErrorSeverity() (*vi.widgets.EditWidget* method), 50  
 formatString() (in module *vi.utils*), 66
- ## G
- getActions() (*vi.framework.components.actionbar.ActionBar* method), 18  
 getActions() (*vi.widgets.list.ListWidget* method), 36  
 getActions() (*vi.widgets.ListWidget* method), 49  
 getActions() (*vi.widgets.tree.TreeWidget* method), 45  
 getActions() (*vi.widgets.TreeWidget* method), 54  
 getAllActions() (*vi.widgets.list.ListWidget* method), 36  
 getAllActions() (*vi.widgets.ListWidget* method), 49  
 getChildKey() (*vi.widgets.file.FileWidget* method), 33  
 getChildKey() (*vi.widgets.FileWidget* method), 56  
 getChildKey() (*vi.widgets.tree.TreeWidget* method), 46  
 getChildKey() (*vi.widgets.TreeWidget* method), 55  
 getConf() (in module *vi.config*), 58  
 getConfigSuccess() (*vi.Application* method), 69  
 getCurrentSelection() (*vi.framework.components.datatable.DataTable* method), 21  
 getCurrentSelection() (*vi.framework.components.datatable.SelectTable* method), 19  
 getCurrentSelection() (*vi.widgets.DataTable* method), 52  
 getCurrentSelection() (*vi.widgets.table.DataTable* method), 42  
 getCurrentSelection() (*vi.widgets.table.SelectTable* method), 41  
 getCurrentUser() (*vi.admin.AdminScreen* method), 57  
 getCurrentUser() (*vi.AdminScreen* method), 68  
 getCurrentUserFailure() (*vi.admin.AdminScreen* method), 57  
 getCurrentUserFailure() (*vi.AdminScreen* method), 68  
 getCurrentUserSuccess() (*vi.admin.AdminScreen* method), 57  
 getCurrentUserSuccess() (*vi.AdminScreen* method), 68  
 getDefaultEntryActions() (*vi.widgets.list.ListWidget* method), 36  
 getDefaultEntryActions() (*vi.widgets.ListWidget* method), 49  
 getFields() (*vi.widgets.list.ListWidget* method), 37  
 getFields() (*vi.widgets.ListWidget* method), 50  
 getFilter() (*vi.widgets.list.ListWidget* method), 37  
 getFilter() (*vi.widgets.ListWidget* method), 50  
 getImagePreview() (in module *vi.utils*), 66  
 getIndexByTr() (*vi.framework.components.datatable.SelectTable* method), 19  
 getIndexByTr() (*vi.widgets.table.SelectTable* method), 40  
 getList() (*vi.utils.indexeddb* method), 66  
 getListKeys() (*vi.utils.indexeddb* method), 67  
 getNavigationPoint() (*vi.widgets.appnavigation.AppNavigation* method), 29  
 getPreviousNavigationPoint() (*vi.widgets.appnavigation.AppNavigation* method), 29  
 getRowCount() (*vi.framework.components.datatable.DataTable* method), 20  
 getRowCount() (*vi.widgets.DataTable* method), 51  
 getRowCount() (*vi.widgets.table.DataTable* method), 41  
 getSelectedTask() (*vi.widgets.task.TaskSelectWidget* method), 43  
 getSelectedTask() (*vi.widgets.TaskSelectWidget* method), 53  
 getTrByIndex() (*vi.framework.components.datatable.SelectTable* method), 19  
 getTrByIndex() (*vi.widgets.table.SelectTable* method), 40  
 getVersionSuccess() (*vi.Application* method), 69  
 getWidget() (*vi.widgets.SideBar* method), 52  
 getWidget() (*vi.widgets.sidebar.SideBar* method), 39  
 GoogleAccountLoginHandler (class in *vi.login*), 61  
 GroupPane (class in *vi.pane*), 63
- ## H
- handleFile() (*vi.widgets.file.MultiUploader* method), 33  
 HandlerClassSelector (in module *vi.priorityqueue*), 64  
 hideListView() (*vi.widgets.hierarchy.HierarchyWidget* method), 34

- hideListView() (*vi.widgets.HierarchyWidget* method), 56
- hideMessage() (*vi.widgets.UserLogoutMsg* method), 53
- hideMessage() (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), 47
- hierarchyHandler (*class in vi.views.hierarchy*), 24
- hierarchyHandlerWidget (*class in vi.views.hierarchy*), 24
- HierarchyWidget (*class in vi.widgets*), 56
- HierarchyWidget (*class in vi.widgets.hierarchy*), 34
- HtmlEditor (*class in vi.widgets*), 54
- HtmlEditor (*class in vi.widgets.htmleditor*), 34
- I
- idbdata() (*vi.log.LogButton* method), 59
- idbTableName (*in module vi.log*), 59
- indexeddb (*class in vi.utils*), 66
- indexeddbConnector (*class in vi.utils*), 66
- initialHashHandler (*in module vi.priorityqueue*), 64
- initializeConfig() (*vi.admin.AdminScreen* method), 57
- initializeConfig() (*vi.AdminScreen* method), 68
- initializeViews() (*vi.admin.AdminScreen* method), 57
- initializeViews() (*vi.AdminScreen* method), 68
- initSources (*vi.widgets.HtmlEditor* attribute), 54
- initSources (*vi.widgets.htmleditor.HtmlEditor* attribute), 34
- initWidget() (*vi.views.edit.editHandlerWidget* method), 23
- initWidget() (*vi.views.hierarchy.hierarchyHandlerWidget* method), 24
- initWidget() (*vi.views.list.listHandlerWidget* method), 24
- initWidget() (*vi.views.log.logHandlerWidget* method), 25
- initWidget() (*vi.views.notfound.NotFoundWidget* method), 25
- initWidget() (*vi.views.overview.OverviewWidget* method), 26
- initWidget() (*vi.views.singleton.singletonHandlerWidget* method), 26
- initWidget() (*vi.views.tree.treeHandlerWidget* method), 27
- insertElem() (*vi.priorityqueue.StartupQueue* method), 64
- insufficientRights() (*vi.login.LoginScreen* method), 62
- insufficientRights() (*vi.LoginScreen* method), 68
- InternalEdit (*class in vi.widgets*), 53
- InternalEdit (*class in vi.widgets.internaledit*), 35
- InternalPreview (*class in vi.sidebarwidgets.internalpreview*), 22
- InvalidBoneValueException, 59
- invertSelection() (*vi.framework.components.datatable.SelectTable* method), 20
- invertSelection() (*vi.widgets.table.SelectTable* method), 41
- invoke() (*vi.admin.AdminScreen* method), 57
- invoke() (*vi.AdminScreen* method), 68
- invoke() (*vi.login.LoginScreen* method), 62
- invoke() (*vi.LoginScreen* method), 68
- invoke() (*vi.screen.Screen* method), 65
- invoke() (*vi.widgets.topbar.TopBarWidget* method), 44
- invoke() (*vi.widgets.TopBarWidget* method), 48
- invokeTask() (*vi.widgets.task.TaskSelectWidget* method), 43
- invokeTask() (*vi.widgets.TaskSelectWidget* method), 53
- isActive() (*vi.widgets.accordion.AccordionSegment* method), 27
- isFullscreen() (*vi.admin.AdminScreen* method), 58
- isFullscreen() (*vi.AdminScreen* method), 69
- 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), 6
- 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), 8
- 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.hierarchy.SelectRootNode* 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), 14
- isSuitableFor() (*vi.actions.list.DeleteAction* static method), 11
- isSuitableFor() (*vi.actions.list.EditAction* static method), 11
- isSuitableFor() (*vi.actions.list.ExportCsvAction* static method), 14
- isSuitableFor() (*vi.actions.list.ListPreviewAction* static method), 12
- isSuitableFor() (*vi.actions.list.ListPreviewInlineAction* static method), 12
- isSuitableFor() (*vi.actions.list.ListSelectFilterAction* static method), 14
- 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), 14
- isSuitableFor() (*vi.actions.list.SelectFieldsAction* static method), 13
- isSuitableFor() (*vi.actions.list.SelectInvertAction* static method), 15
- isSuitableFor() (*vi.actions.list.SetPageRowAmountAction* static method), 13
- 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), 16
- isSuitableFor() (*vi.actions.tree.AddNodeAction* static method), 16
- isSuitableFor() (*vi.actions.tree.DeleteAction* static method), 17
- isSuitableFor() (*vi.actions.tree.EditAction* static method), 17
- isSuitableFor() (*vi.actions.tree.ReloadAction* static method), 17
- isSuitableFor() (*vi.actions.tree.SelectRootNode* static method), 17
- isSuitableFor() (*vi.widgets.htmleditor.TextInsertImageAction* static method), 34
- itemForKey() (*vi.widgets.tree.TreeWidget* method), 45
- itemForKey() (*vi.widgets.TreeWidget* method), 55
- ## L
- leafWidget (*vi.widgets.file.FileWidget* attribute), 33
- leafWidget (*vi.widgets.FileWidget* attribute), 56
- leafWidget (*vi.widgets.hierarchy.HierarchyWidget* attribute), 34
- leafWidget (*vi.widgets.HierarchyWidget* attribute), 56
- leafWidget (*vi.widgets.tree.TreeBrowserWidget* attribute), 46
- leafWidget (*vi.widgets.tree.TreeWidget* attribute), 45
- leafWidget (*vi.widgets.TreeBrowserWidget* attribute), 55
- leafWidget (*vi.widgets.TreeWidget* attribute), 54
- listHandler (*class in vi.views.list*), 24
- listHandlerWidget (*class in vi.views.list*), 24
- ListPreviewAction (*class in vi.actions.list*), 11
- ListPreviewInlineAction (*class in vi.actions.list*), 12
- ListSelectFilterAction (*class in vi.actions.list*), 14
- ListViewAction (*class in vi.actions.hierarchy*), 9
- ListWidget (*class in vi.widgets*), 49
- ListWidget (*class in vi.widgets.list*), 36
- lngDe (*in module vi.translations*), 23
- lngDe (*in module vi.translations.de*), 23
- lngEn (*in module vi.translations*), 23
- lngEn (*in module vi.translations.en*), 23
- LoadAllAction (*class in vi.actions.list*), 14
- loadAllRows() (*vi.actions.list.LoadAllAction* method), 14
- loadChildren() (*vi.pane.GroupPane* method), 63
- LoadNextBatchAction (*class in vi.actions.list*), 13
- loadnextPages() (*vi.actions.list.LoadNextBatchAction* method), 14
- loadNode() (*vi.widgets.tree.TreeWidget* method), 46
- loadNode() (*vi.widgets.TreeWidget* method), 55
- lock() (*vi.login.BaseLoginHandler* method), 61
- lock() (*vi.pane.Pane* method), 62
- lock() (*vi.screen.Screen* method), 65
- Log (*class in vi.log*), 60
- log() (*vi.admin.AdminScreen* method), 57
- log() (*vi.AdminScreen* method), 69

log() (*vi.log.Log method*), 60  
 log() (*vi.log.LogButton method*), 60  
 logA (*class in vi.log*), 59  
 LogButton (*class in vi.log*), 59  
 logEntry (*class in vi.log*), 59  
 logHandler (*class in vi.views.log*), 25  
 logHandlerWidget (*class in vi.views.log*), 25  
 login() (*vi.Application method*), 69  
 login() (*vi.login.BaseLoginHandler method*), 61  
 loginHandlerSelector (*in module vi.priorityqueue*), 64  
 LoginInputField (*class in vi.login*), 61  
 LoginScreen (*class in vi*), 68  
 LoginScreen (*class in vi.login*), 61  
 Logout (*class in vi.widgets.topbar*), 44  
 logout() (*vi.Application method*), 69  
 logout() (*vi.widgets.topbar.Logout method*), 44  
 logWidget (*class in vi.log*), 59

## M

mergeDict() (*in module vi.utils*), 67  
 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, 15  
   vi.actions.tree, 16  
   vi.admin, 57  
   vi.config, 58  
   vi.exception, 59  
   vi.framework, 18  
   vi.framework.components, 18  
   vi.framework.components.actionbar, 18  
   vi.framework.components.datatable, 18  
   vi.log, 59  
   vi.login, 60  
   vi.pane, 62  
   vi.priorityqueue, 63  
   vi.screen, 65  
   vi.serversideaction, 65  
   vi.sidebarwidgets, 22  
   vi.sidebarwidgets.filterselector, 22  
   vi.sidebarwidgets.internalpreview, 22  
   vi.translations, 23  
   vi.translations.de, 23  
   vi.translations.en, 23  
   vi.utils, 66  
   vi.views, 23  
   vi.views.edit, 23  
   vi.views.hierarchy, 24

  vi.views.list, 24  
   vi.views.log, 25  
   vi.views.notfound, 25  
   vi.views.overview, 25  
   vi.views.singleton, 26  
   vi.views.tree, 26  
   vi.widgets, 27  
   vi.widgets.accordion, 27  
   vi.widgets.appnavigation, 28  
   vi.widgets.csvexport, 29  
   vi.widgets.edit, 30  
   vi.widgets.file, 32  
   vi.widgets.hierarchy, 34  
   vi.widgets.htmleditor, 34  
   vi.widgets.internaedit, 35  
   vi.widgets.list, 36  
   vi.widgets.preview, 38  
   vi.widgets.repeatdate, 38  
   vi.widgets.search, 39  
   vi.widgets.sidebar, 39  
   vi.widgets.table, 40  
   vi.widgets.task, 42  
   vi.widgets.tooltip, 43  
   vi.widgets.topbar, 43  
   vi.widgets.tree, 44  
   vi.widgets.userlogoutmsg, 47  
 msgOverlay() (*vi.log.LogButton method*), 60  
 MultiUploader (*class in vi.widgets.file*), 33

## N

navigationAction() (*vi.widgets.appnavigation.NavigationElement method*), 28  
 Navigationblock (*class in vi.widgets.appnavigation*), 29  
 NavigationElement (*class in vi.widgets.appnavigation*), 28  
 NavigationSeperator (*class in vi.widgets.appnavigation*), 29  
 next() (*vi.priorityqueue.StartupQueue method*), 64  
 nextChunk() (*vi.widgets.csvexport.ExportCsv method*), 29  
 nextChunk() (*vi.widgets.ExportCsv method*), 54  
 nextChunkComplete() (*vi.widgets.csvexport.ExportCsv method*), 29  
 nextChunkComplete() (*vi.widgets.ExportCsv method*), 54  
 nextChunkFailure() (*vi.widgets.csvexport.ExportCsv method*), 30  
 nextChunkFailure() (*vi.widgets.ExportCsv method*), 54  
 nodeWidget (*vi.widgets.file.FileWidget attribute*), 33  
 nodeWidget (*vi.widgets.FileWidget attribute*), 56  
 nodeWidget (*vi.widgets.tree.TreeBrowserWidget attribute*), 46



- nodeWidget (*vi.widgets.tree.TreeWidget* attribute), 45
- nodeWidget (*vi.widgets.TreeBrowserWidget* attribute), 56
- nodeWidget (*vi.widgets.TreeWidget* attribute), 54
- NotFound (*class in vi.views.notfound*), 25
- NotFoundWidget (*class in vi.views.notfound*), 25
- ## O
- onActiveNavigationChanged() (*vi.widgets.appnavigation.NavigationElement* method), 29
- onActiveViewChanged() (*vi.widgets.appnavigation.NavigationElement* method), 28
- 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), 8
- onAttach() (*vi.actions.hierarchy.DeleteAction* method), 9
- onAttach() (*vi.actions.hierarchy.EditAction* method), 8
- onAttach() (*vi.actions.hierarchy.SelectRootNode* method), 9
- onAttach() (*vi.actions.list.CloneAction* method), 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), 14
- onAttach() (*vi.actions.list.SelectAllAction* method), 14
- onAttach() (*vi.actions.list.SelectFieldsAction* method), 12
- onAttach() (*vi.actions.list.SelectInvertAction* method), 15
- onAttach() (*vi.actions.list.UnSelectAllAction* method), 15
- onAttach() (*vi.actions.list\_order.ShopMarkAction* method), 15
- onAttach() (*vi.actions.tree.DeleteAction* method), 17
- onAttach() (*vi.actions.tree.EditAction* method), 16
- onAttach() (*vi.actions.tree.SelectRootNode* method), 17
- onAttach() (*vi.framework.components.datatable.SelectTable* method), 19
- onAttach() (*vi.serversideaction.ServerSideActionWdg* method), 65
- onAttach() (*vi.sidebarwidgets.filterselector.FilterSelector* method), 22
- onAttach() (*vi.widgets.edit.EditWidget* method), 31
- onAttach() (*vi.widgets.EditWidget* method), 50
- onAttach() (*vi.widgets.HtmlEditor* method), 54
- onAttach() (*vi.widgets.htmleditor.HtmlEditor* method), 35
- onAttach() (*vi.widgets.list.ListWidget* method), 37
- onAttach() (*vi.widgets.ListWidget* method), 49
- onAttach() (*vi.widgets.SideBar* method), 52
- onAttach() (*vi.widgets.sidebar.SideBar* method), 39
- onAttach() (*vi.widgets.table.SelectTable* method), 40
- onAttach() (*vi.widgets.tree.TreeWidget* method), 45
- onAttach() (*vi.widgets.TreeWidget* method), 55
- onBoneChange() (*vi.widgets.edit.EditWidget* method), 31
- onBoneChange() (*vi.widgets.EditWidget* method), 50
- onBtnCloseReleased() (*vi.pane.Pane* method), 62
- onChange() (*vi.actions.file.FileSelectUploader* method), 6
- onChange() (*vi.actions.hierarchy.SelectRootNode* method), 9
- onChange() (*vi.actions.list.ListPreviewAction* method), 12
- onChange() (*vi.actions.list.LoadNextBatchAction* method), 13
- onChange() (*vi.actions.list.SetPageRowAmountAction* method), 13
- onChange() (*vi.actions.tree.SelectRootNode* method), 17
- onChange() (*vi.framework.components.datatable.SelectTable* method), 19
- onChange() (*vi.widgets.edit.EditWidget* method), 31
- onChange() (*vi.widgets.EditWidget* method), 50
- onChange() (*vi.widgets.InternalEdit* method), 53
- onChange() (*vi.widgets.internaledit.InternalEdit* method), 36
- onChange() (*vi.widgets.preview.Preview* method), 38
- onChange() (*vi.widgets.table.SelectTable* method), 40
- onChange() (*vi.widgets.task.TaskSelectWidget* method), 43
- onChange() (*vi.widgets.TaskSelectWidget* method), 53
- 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), 6
- 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), 8
- 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*), 14
- onClick() (*vi.actions.list.DeleteAction method*), 11
- onClick() (*vi.actions.list.EditAction method*), 11
- onClick() (*vi.actions.list.ExportCsvAction method*), 14
- onClick() (*vi.actions.list.ListPreviewAction method*), 12
- onClick() (*vi.actions.list.ListPreviewInlineAction method*), 12
- onClick() (*vi.actions.list.ListSelectFilterAction method*), 14
- onClick() (*vi.actions.list.LoadAllAction method*), 14
- onClick() (*vi.actions.list.LoadNextBatchAction method*), 13
- 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*), 14
- onClick() (*vi.actions.list.SelectFieldsAction method*), 12
- onClick() (*vi.actions.list.SelectInvertAction method*), 15
- onClick() (*vi.actions.list.SetPageRowAmountAction method*), 13
- 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*), 16
- onClick() (*vi.actions.tree.AddNodeAction method*), 16
- onClick() (*vi.actions.tree.DeleteAction method*), 17
- onClick() (*vi.actions.tree.EditAction method*), 17
- onClick() (*vi.actions.tree.ReloadAction method*), 17
- onClick() (*vi.admin.AdminScreen method*), 57
- onClick() (*vi.AdminScreen method*), 68
- onClick() (*vi.log.logA method*), 59
- onClick() (*vi.log.LogButton method*), 60
- onClick() (*vi.login.BaseLoginHandler method*), 61
- onClick() (*vi.pane.GroupPane method*), 63
- onClick() (*vi.pane.Pane method*), 63
- onClick() (*vi.serversideaction.ServerSideActionWdg method*), 65
- onClick() (*vi.sidebarwidgets.filterselector.FilterSelector method*), 22
- onClick() (*vi.widgets.accordion.AccordionSegment method*), 27
- onClick() (*vi.widgets.file.FileImagePopup method*), 32
- onClick() (*vi.widgets.file.FilePreviewImage method*), 32
- onClick() (*vi.widgets.htmleditor.TextInsertImageAction method*), 34
- onClick() (*vi.widgets.list.ListWidget method*), 36
- onClick() (*vi.widgets.ListWidget method*), 49
- onClick() (*vi.widgets.ToolTip method*), 51
- onClick() (*vi.widgets.tooltip.ToolTip method*), 43
- onClick() (*vi.widgets.topbar.Logout method*), 44
- onClick() (*vi.widgets.topbar.Tasks method*), 44
- onClick() (*vi.widgets.topbar.TopBarWidget method*), 44
- onClick() (*vi.widgets.topbar.UserState method*), 44
- onClick() (*vi.widgets.TopBarWidget method*), 49
- onCompletion() (*vi.widgets.list.ListWidget method*), 37
- onCompletion() (*vi.widgets.ListWidget method*), 50
- onCopyKey() (*vi.sidebarwidgets.internalpreview.InternalPreview method*), 22
- onCurrentUserAvailable() (*vi.widgets.topbar.Tasks method*), 44
- onCurrentUserAvailable() (*vi.widgets.topbar.UserState method*), 44
- onCursorMoved() (*vi.framework.components.datatable.DataTable method*), 20
- onCursorMoved() (*vi.widgets.DataTable method*), 51
- onCursorMoved() (*vi.widgets.table.DataTable method*), 41
- onDataChanged() (*vi.widgets.list.ListWidget method*), 37
- onDataChanged() (*vi.widgets.ListWidget method*), 49
- onDataChanged() (*vi.widgets.tree.TreeWidget method*), 45
- onDataChanged() (*vi.widgets.TreeWidget method*), 55
- onDbClick() (*vi.framework.components.datatable.SelectTable method*), 19
- onDbClick() (*vi.widgets.table.SelectTable method*), 40
- 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*), 8
- onDetach() (*vi.actions.hierarchy.DeleteAction method*), 9
- onDetach() (*vi.actions.hierarchy.EditAction method*), 8
- onDetach() (*vi.actions.hierarchy.SelectRootNode method*), 9
- onDetach() (*vi.actions.list.CloneAction method*), 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*), 14

- onDetach() (*vi.actions.list.SelectFieldsAction method*), 12  
 onDetach() (*vi.actions.list.SelectInvertAction method*), 15  
 onDetach() (*vi.actions.list.UnSelectAllAction method*), 15  
 onDetach() (*vi.actions.list\_order.ShopMarkAction method*), 15  
 onDetach() (*vi.actions.tree.DeleteAction method*), 17  
 onDetach() (*vi.actions.tree.EditAction method*), 17  
 onDetach() (*vi.actions.tree.SelectRootNode method*), 17  
 onDetach() (*vi.pane.Pane method*), 63  
 onDetach() (*vi.serversideaction.ServerSideActionWdg method*), 65  
 onDetach() (*vi.sidebarwidgets.filterselector.FilterSelector method*), 22  
 onDetach() (*vi.widgets.edit.EditWidget method*), 31  
 onDetach() (*vi.widgets.EditWidget method*), 50  
 onDetach() (*vi.widgets.HtmlEditor method*), 54  
 onDetach() (*vi.widgets.htmleditor.HtmlEditor method*), 35  
 onDetach() (*vi.widgets.list.ListWidget method*), 37  
 onDetach() (*vi.widgets.ListWidget method*), 49  
 onDetach() (*vi.widgets.SideBar method*), 52  
 onDetach() (*vi.widgets.sidebar.SideBar method*), 39  
 onDetach() (*vi.widgets.tree.TreeWidget method*), 45  
 onDetach() (*vi.widgets.TreeWidget method*), 55  
 onDownloadBtnClick() (*vi.widgets.file.FileImagePopup method*), 32  
 onDragOver() (*vi.widgets.tree.TreeWidget method*), 46  
 onDragOver() (*vi.widgets.TreeWidget method*), 55  
 onDrop() (*vi.widgets.file.FileWidget method*), 33  
 onDrop() (*vi.widgets.FileWidget method*), 56  
 onDrop() (*vi.widgets.tree.TreeWidget method*), 46  
 onDrop() (*vi.widgets.TreeWidget method*), 55  
 onEditorChange() (*vi.widgets.HtmlEditor method*), 54  
 onEditorChange() (*vi.widgets.htmleditor.HtmlEditor method*), 35  
 onError() (*vi.admin.AdminScreen method*), 58  
 onError() (*vi.AdminScreen method*), 69  
 onExportBtnClick() (*vi.widgets.csvexport.ExportCsvStarter method*), 30  
 onExportBtnClick() (*vi.widgets.ExportCsvStarter method*), 54  
 onFailed() (*vi.widgets.file.MultiUploader method*), 33  
 onFailed() (*vi.widgets.file.Uploader method*), 32  
 onFilterChanged() (*vi.sidebarwidgets.filterselector.CompoundFilter method*), 22  
 onFocus() (*vi.pane.GroupPane method*), 63  
 onGetAuthMethodsFailure() (*vi.login.LoginScreen method*), 62  
 onGetAuthMethodsFailure() (*vi.LoginScreen method*), 68  
 onGetAuthMethodsSuccess() (*vi.login.LoginScreen method*), 62  
 onGetAuthMethodsSuccess() (*vi.LoginScreen method*), 68  
 onHasSubItemsChanged() (*vi.widgets.appnavigation.NavigationElement method*), 29  
 onKeyDown() (*vi.framework.components.datatable.SelectTable method*), 19  
 onKeyDown() (*vi.widgets.InternalEdit method*), 53  
 onKeyDown() (*vi.widgets.internaedit.InternalEdit method*), 36  
 onKeyDown() (*vi.widgets.Search method*), 52  
 onKeyDown() (*vi.widgets.search.Search method*), 39  
 onKeyDown() (*vi.widgets.table.SelectTable method*), 40  
 onKeyDown() (*vi.widgets.tree.TreeWidget method*), 45  
 onKeyDown() (*vi.widgets.TreeWidget method*), 54  
 onKeyPress() (*vi.actions.list.PageFindAction method*), 14  
 onKeyPress() (*vi.login.LoginInputField method*), 61  
 onKeyPress() (*vi.login.UserPasswordLoginHandler method*), 61  
 onKeyUp() (*vi.framework.components.datatable.SelectTable method*), 19  
 onKeyUp() (*vi.widgets.table.SelectTable method*), 40  
 onKeyUp() (*vi.widgets.tree.TreeWidget method*), 45  
 onKeyUp() (*vi.widgets.TreeWidget method*), 54  
 onLoad() (*vi.widgets.file.MultiUploader method*), 33  
 onLoad() (*vi.widgets.file.Uploader method*), 32  
 onLoginClick() (*vi.login.GoogleAccountLoginHandler method*), 61  
 onLoginClick() (*vi.login.UserPasswordLoginHandler method*), 61  
 onLogoutSuccess() (*vi.login.LoginScreen method*), 62  
 onLogoutSuccess() (*vi.LoginScreen method*), 68  
 onMkdir() (*vi.actions.file.AddNodeAction method*), 7  
 onMouseDown() (*vi.framework.components.datatable.SelectTable method*), 19  
 onMouseDown() (*vi.widgets.table.SelectTable method*), 40  
 onMouseOut() (*vi.framework.components.datatable.SelectTable method*), 19  
 onMouseOut() (*vi.widgets.table.SelectTable method*), 40  
 onMouseUp() (*vi.framework.components.datatable.SelectTable method*), 19  
 onMouseUp() (*vi.widgets.table.SelectTable method*), 40  
 onNextBatchNeeded() (*vi.widgets.list.ListWidget method*), 37  
 onNextBatchNeeded() (*vi.widgets.ListWidget method*), 49  
 onPathRequestSucceeded() (*vi.widgets.tree.TreeBrowserWidget method*), 46  
 onPathRequestSucceeded()

(*vi.widgets.TreeBrowserWidget* method), 56  
 onProgress() (*vi.widgets.file.Uploader* method), 32  
 onRequestingFinished() (*vi.widgets.list.ListWidget* method), 36  
 onRequestingFinished() (*vi.widgets.list.ViewportListWidget* method), 38  
 onRequestingFinished() (*vi.widgets.ListWidget* method), 49  
 onRequestSucceeded() (*vi.widgets.tree.TreeWidget* method), 46  
 onRequestSucceeded() (*vi.widgets.TreeWidget* method), 55  
 onRootNodeChanged() (*vi.actions.hierarchy.SelectRootNode* method), 9  
 onRootNodeChanged() (*vi.actions.tree.SelectRootNode* method), 17  
 onRootNodesAvailable() (*vi.actions.hierarchy.SelectRootNode* method), 9  
 onRootNodesAvailable() (*vi.actions.tree.SelectRootNode* method), 17  
 onScroll() (*vi.actions.list.LoadNextBatchAction* method), 13  
 onScroll() (*vi.widgets.DataTable* method), 52  
 onScroll() (*vi.widgets.table.DataTable* method), 42  
 onSelectionActivated() (*vi.actions.file.EditAction* method), 7  
 onSelectionActivated() (*vi.actions.hierarchy.EditAction* method), 8  
 onSelectionActivated() (*vi.actions.list.EditAction* method), 11  
 onSelectionActivated() (*vi.actions.tree.EditAction* method), 17  
 onSelectionActivated() (*vi.framework.components.datatable.DataTable* method), 21  
 onSelectionActivated() (*vi.widgets.DataTable* method), 52  
 onSelectionActivated() (*vi.widgets.htmleditor.TextInsertImageAction* method), 34  
 onSelectionActivated() (*vi.widgets.list.ListWidget* method), 37  
 onSelectionActivated() (*vi.widgets.ListWidget* method), 50  
 onSelectionActivated() (*vi.widgets.table.DataTable* method), 42  
 onSelectionChanged() (*vi.actions.context.ContextAction* method), 5  
 onSelectionChanged() (*vi.actions.file.DownloadAction* method), 7  
 onSelectionChanged() (*vi.actions.file.EditAction* method), 7  
 onSelectionChanged() (*vi.actions.hierarchy.CloneAction* method), 8  
 onSelectionChanged() (*vi.actions.hierarchy.DeleteAction* method), 9  
 onSelectionChanged() (*vi.actions.hierarchy.EditAction* method), 8  
 onSelectionChanged() (*vi.actions.list.CloneAction* method), 11  
 onSelectionChanged() (*vi.actions.list.DeleteAction* method), 11  
 onSelectionChanged() (*vi.actions.list.EditAction* method), 11  
 onSelectionChanged() (*vi.actions.list.ListPreviewAction* method), 12  
 onSelectionChanged() (*vi.actions.list.ListPreviewInlineAction* method), 12  
 onSelectionChanged() (*vi.actions.list\_order.ShopMarkAction* method), 15  
 onSelectionChanged() (*vi.actions.tree.DeleteAction* method), 17  
 onSelectionChanged() (*vi.actions.tree.EditAction* method), 17  
 onSelectionChanged() (*vi.framework.components.datatable.DataTable* method), 21  
 onSelectionChanged() (*vi.serversideaction.ServerSideActionWdg* method), 65  
 onSelectionChanged() (*vi.widgets.DataTable* method), 52  
 onSelectionChanged() (*vi.widgets.hierarchy.HierarchyWidget* method), 34  
 onSelectionChanged() (*vi.widgets.HierarchyWidget* method), 56  
 onSelectionChanged() (*vi.widgets.table.DataTable* method), 42  
 onSendClick() (*vi.login.UserPasswordLoginHandler* method), 61  
 onSetDefaultRootNode() (*vi.widgets.tree.TreeWidget* method), 45  
 onSetDefaultRootNode() (*vi.widgets.TreeWidget* method), 55  
 onSkeyAvailable() (*vi.widgets.file.Uploader* method),



- 32
- onStartSearch() (*vi.sidebarwidgets.filterselector.FilterSelector method*), 22
- onStartSearch() (*vi.widgets.file.FileWidget method*), 33
- onStartSearch() (*vi.widgets.FileWidget method*), 56
- onSuccess() (*vi.widgets.file.MultiUploader method*), 33
- onSuccess() (*vi.widgets.file.Uploader method*), 32
- onTableChanged() (*vi.actions.list.SelectAllAction method*), 15
- onTableChanged() (*vi.actions.list.SelectFieldsAction method*), 12
- 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*), 21
- onTableChanged() (*vi.widgets.DataTable method*), 52
- onTableChanged() (*vi.widgets.table.DataTable method*), 42
- onTaskListAvailable() (*vi.widgets.topbar.Tasks method*), 44
- onTaskListFailure() (*vi.widgets.topbar.Tasks method*), 44
- onUploadAdded() (*vi.widgets.file.MultiUploader method*), 33
- onUploadAdded() (*vi.widgets.file.Uploader method*), 32
- onUploadUrlAvailable() (*vi.widgets.file.MultiUploader method*), 33
- onUploadUrlAvailable() (*vi.widgets.file.Uploader method*), 32
- onUserTestFail() (*vi.widgets.UserLogoutMsg method*), 53
- onUserTestFail() (*vi.widgets.userlogoutmsg.UserLogoutMsg method*), 47
- onUserTestSuccess() (*vi.widgets.UserLogoutMsg method*), 53
- onUserTestSuccess() (*vi.widgets.userlogoutmsg.UserLogoutMsg method*), 47
- onVerifyClick() (*vi.login.UserPasswordLoginHandler method*), 61
- onViewfocusedChanged() (*vi.views.list.listHandlerWidget method*), 24
- onViewfocusedChanged() (*vi.views.singleton.singletonHandlerWidget method*), 26
- openEdit() (*vi.widgets.topbar.UserState method*), 44
- openEditor() (*vi.actions.hierarchy.CloneAction method*), 8
- 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*), 59
- openLog() (*vi.log.LogButton method*), 60
- openModule() (*vi.actions.context.ContextAction method*), 5
- openNewMainView() (*vi.admin.AdminScreen method*), 57
- openNewMainView() (*vi.AdminScreen method*), 68
- openNewPopup() (*vi.admin.AdminScreen method*), 57
- openNewPopup() (*vi.AdminScreen method*), 69
- openView() (*vi.admin.AdminScreen method*), 57
- openView() (*vi.AdminScreen method*), 68
- Overview (*class in vi.views.overview*), 26
- OverviewWidget (*class in vi.views.overview*), 26
- ## D
- DataTable
- PageFindAction (*class in vi.actions.list*), 14
- Pane (*class in vi.pane*), 62
- parseAnswer() (*vi.login.BaseLoginHandler method*), 61
- ParsedErrorItem (*class in vi.widgets.edit*), 30
- ParsedErrorItem (*class in vi.widgets.internaledit*), 35
- parseHashParameters() (*in module vi.widgets.edit*), 30
- PassiveErrorItem (*class in vi.widgets.edit*), 30
- PassiveErrorItem (*class in vi.widgets.internaledit*), 35
- performLogics() (*vi.widgets.edit.EditWidget method*), 31
- performLogics() (*vi.widgets.EditWidget method*), 50
- performLogics() (*vi.widgets.InternalEdit method*), 53
- performLogics() (*vi.widgets.internaledit.InternalEdit method*), 36
- performReload() (*vi.actions.edit.Refresh method*), 6
- pollInterval (*vi.widgets.UserLogoutMsg attribute*), 53
- pollInterval (*vi.widgets.userlogoutmsg.UserLogoutMsg attribute*), 47
- postInit() (*vi.actions.list.TableItems method*), 13
- postInit() (*vi.actions.list.TableNextPage method*), 13
- postInit() (*vi.actions.list.TablePrevPage method*), 13
- prepareCol() (*vi.framework.components.datatable.SelectTable method*), 20
- prepareCol() (*vi.widgets.table.SelectTable method*), 41
- Preview (*class in vi.widgets.preview*), 38
- protocolWrapperClassSelector (*in module vi.priorityqueue*), 64
- protocolWrapperInstanceSelector (*in module vi.priorityqueue*), 64
- ## Q
- queue (*vi.utils.indexeddb attribute*), 66

## R

- rebuildCB() (*vi.actions.list.ListPreviewAction* method), 12  
 rebuildChildrenClassInfo() (*vi.pane.Pane* method), 63  
 rebuildPath() (*vi.widgets.tree.TreeBrowserWidget* method), 46  
 rebuildPath() (*vi.widgets.TreeBrowserWidget* method), 56  
 rebuildTable() (*vi.framework.components.datatable.DataTable* method), 20  
 rebuildTable() (*vi.framework.components.datatable.ViewportDataTable* method), 21  
 rebuildTable() (*vi.widgets.DataTable* method), 52  
 rebuildTable() (*vi.widgets.table.DataTable* method), 42  
 recalHeight() (*vi.widgets.DataTable* method), 51  
 recalHeight() (*vi.widgets.table.DataTable* method), 41  
 receivedStructure() (*vi.widgets.list.ListWidget* method), 37  
 receivedStructure() (*vi.widgets.ListWidget* method), 49  
 receivedStructure() (*vi.widgets.tree.TreeWidget* method), 45  
 receivedStructure() (*vi.widgets.TreeWidget* method), 54  
 redirectNoAdmin() (*vi.login.LoginScreen* method), 62  
 redirectNoAdmin() (*vi.LoginScreen* method), 68  
 reevaluate() (*vi.sidebarwidgets.filtersselector.CompoundFilter* method), 22  
 reevaluate() (*vi.widgets.Search* method), 52  
 reevaluate() (*vi.widgets.search.Search* method), 39  
 Refresh (class in *vi.actions.edit*), 6  
 registerScroll() (*vi.actions.list.LoadNextBatchAction* method), 13  
 ReloadAction (class in *vi.actions.hierarchy*), 9  
 ReloadAction (class in *vi.actions.list*), 13  
 ReloadAction (class in *vi.actions.tree*), 17  
 reloadData() (*vi.widgets.edit.EditWidget* method), 31  
 reloadData() (*vi.widgets.EditWidget* method), 50  
 reloadData() (*vi.widgets.hierarchy.HierarchyWidget* method), 34  
 reloadData() (*vi.widgets.HierarchyWidget* method), 56  
 reloadData() (*vi.widgets.list.ListWidget* method), 37  
 reloadData() (*vi.widgets.ListWidget* method), 49  
 reloadData() (*vi.widgets.repeatdate.RepeatDatePopup* method), 38  
 reloadData() (*vi.widgets.tree.TreeBrowserWidget* method), 46  
 reloadData() (*vi.widgets.tree.TreeWidget* method), 46  
 reloadData() (*vi.widgets.TreeBrowserWidget* method), 56  
 reloadData() (*vi.widgets.TreeWidget* method), 55  
 reloadDataWidget() (*vi.widgets.hierarchy.HierarchyWidget* method), 34  
 reloadDataWidget() (*vi.widgets.HierarchyWidget* method), 56  
 remove() (*vi.framework.components.datatable.DataTable* method), 20  
 remove() (*vi.screen.Screen* method), 65  
 remove() (*vi.widgets.DataTable* method), 51  
 remove() (*vi.widgets.table.DataTable* method), 42  
 RemoveAction() (*vi.widgets.appnavigation.NavigationElement* method), 28  
 removeChildPane() (*vi.pane.Pane* method), 62  
 removeInfo() (*vi.log.LogButton* method), 60  
 removeNavigationPoint() (*vi.widgets.appnavigation.AppNavigation* method), 29  
 removeNewCls() (*vi.log.Log* method), 60  
 removeRow() (*vi.framework.components.datatable.SelectTable* method), 20  
 removeRow() (*vi.widgets.table.SelectTable* method), 41  
 removeSelectedRow() (*vi.framework.components.datatable.SelectTable* method), 19  
 removeSelectedRow() (*vi.widgets.table.SelectTable* method), 40  
 removeWidget() (*vi.admin.AdminScreen* method), 58  
 removeWidget() (*vi.AdminScreen* method), 69  
 removeWidget() (*vi.pane.Pane* method), 63  
 renderPopOut() (*vi.log.LogButton* method), 60  
 renderStructure() (*vi.widgets.InternalEdit* method), 53  
 renderStructure() (*vi.widgets.internaledit.InternalEdit* method), 35  
 RepeatDatePopup (class in *vi.widgets.repeatdate*), 38  
 replaceWithMessage() (*vi.widgets.csvexport.ExportCsv* method), 30  
 replaceWithMessage() (*vi.widgets.ExportCsv* method), 54  
 replaceWithMessage() (*vi.widgets.file.MultiUploader* method), 33  
 replaceWithMessage() (*vi.widgets.file.Uploader* method), 32  
 requestChildren() (*vi.widgets.tree.TreeWidget* method), 45  
 requestChildren() (*vi.widgets.TreeWidget* method), 55  
 requestStructure() (*vi.widgets.list.ListWidget* method), 37  
 requestStructure() (*vi.widgets.ListWidget* method), 49  
 requestStructure() (*vi.widgets.tree.TreeWidget* method), 45  
 requestStructure() (*vi.widgets.TreeWidget* method),

- 54
- reset() (*vi.admin.AdminScreen* method), 57
- reset() (*vi.AdminScreen* method), 68
- reset() (*vi.log.Log* method), 60
- reset() (*vi.log.LogButton* method), 60
- reset() (*vi.login.BaseLoginHandler* method), 61
- reset() (*vi.login.UserPasswordLoginHandler* method), 61
- reset() (*vi.priorityqueue.StartupQueue* method), 64
- 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), 7
- 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), 11
- 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), 13
- resetLoadingState() (*vi.actions.list.TableNextPage* method), 13
- resetLoadingState() (*vi.actions.tree.AddLeafAction* method), 16
- resetLoadingState() (*vi.actions.tree.AddNodeAction* method), 16
- resetLoadingState() (*vi.actions.tree.DeleteAction* method), 17
- resetLoadingState() (*vi.actions.tree.EditAction* method), 17
- resetLoadingState() (*vi.actions.tree.ReloadAction* method), 17
- resetLoadingState() (*vi.framework.components.actionbar.ActionBar* method), 18
- resetLoadingState() (*vi.serversideaction.ServerSideActionWdg* method), 65
- resetLoadingState() (*vi.widgets.htmleditor.TextInsertImageAction* method), 34
- resetLoadingState() (*vi.widgets.Search* method), 52
- resetLoadingState() (*vi.widgets.search.Search* method), 39
- resetSearch() (*vi.widgets.Search* method), 52
- resetSearch() (*vi.widgets.search.Search* method), 39
- run() (*vi.priorityqueue.StartupQueue* method), 64
- ## S
- s (*in module vi*), 69
- save() (*vi.widgets.edit.EditWidget* method), 31
- save() (*vi.widgets.EditWidget* method), 50
- save() (*vi.widgets.repeatdate.RepeatDatePopup* method), 38
- 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*), 69
- scinv (*in module vi*), 69
- Screen (*class in vi.screen*), 65
- Search (*class in vi.widgets*), 52
- Search (*class in vi.widgets.search*), 39
- searchWidget() (*vi.widgets.file.FileWidget* method), 33
- searchWidget() (*vi.widgets.FileWidget* method), 56

- SelectAction (class in *vi.actions.list*), 12  
 selectAll() (*vi.framework.components.datatable.SelectTable* method), 20  
 selectAll() (*vi.widgets.table.SelectTable* method), 41  
 SelectAllAction (class in *vi.actions.list*), 14  
 SelectFieldsAction (class in *vi.actions.list*), 12  
 SelectFieldsPopup (class in *vi.actions.list*), 12  
 selectHandler() (*vi.login.LoginScreen* method), 62  
 selectHandler() (*vi.LoginScreen* method), 68  
 SelectInvertAction (class in *vi.actions.list*), 15  
 selectorReturn() (*vi.widgets.list.ListWidget* method), 36  
 selectorReturn() (*vi.widgets.ListWidget* method), 49  
 selectorReturn() (*vi.widgets.tree.TreeWidget* method), 45  
 selectorReturn() (*vi.widgets.TreeWidget* method), 54  
 SelectRootNode (class in *vi.actions.hierarchy*), 9  
 SelectRootNode (class in *vi.actions.tree*), 17  
 selectRow() (*vi.framework.components.datatable.SelectTable* method), 19  
 selectRow() (*vi.widgets.table.SelectTable* method), 41  
 SelectTable (class in *vi.framework.components.datatable*), 18  
 SelectTable (class in *vi.widgets.table*), 40  
 seperatorAction() (*vi.widgets.appnavigation.Navigationblock* method), 29  
 serializeForDocument() (*vi.widgets.edit.EditWidget* method), 31  
 serializeForDocument() (*vi.widgets.EditWidget* method), 51  
 serializeForDocument() (*vi.widgets.InternalEdit* method), 53  
 serializeForDocument() (*vi.widgets.internaledit.InternalEdit* method), 36  
 serializeForPost() (*vi.widgets.edit.EditWidget* method), 31  
 serializeForPost() (*vi.widgets.EditWidget* method), 51  
 serializeForPost() (*vi.widgets.InternalEdit* method), 53  
 serializeForPost() (*vi.widgets.internaledit.InternalEdit* method), 35  
 ServerSideActionWdg (class in *vi.serversideaction*), 65  
 ServerTaskWidget (class in *vi.widgets.task*), 43  
 setActions() (*vi.framework.components.actionbar.Actionbar* method), 18  
 setActiveTask() (*vi.widgets.task.TaskSelectWidget* method), 43  
 setActiveTask() (*vi.widgets.TaskSelectWidget* method), 53  
 setAmount() (*vi.widgets.list.ListWidget* method), 36  
 setAmount() (*vi.widgets.list.ViewportListWidget* method), 37  
 setAmount() (*vi.widgets.ListWidget* method), 49  
 setCell() (*vi.framework.components.datatable.SelectTable* method), 20  
 setCell() (*vi.widgets.table.SelectTable* method), 41  
 setCellRender() (*vi.framework.components.datatable.DataTable* method), 21  
 setCellRender() (*vi.widgets.DataTable* method), 52  
 setCellRender() (*vi.widgets.table.DataTable* method), 42  
 setCellRenderers() (*vi.framework.components.datatable.DataTable* method), 21  
 setCellRenderers() (*vi.widgets.DataTable* method), 52  
 setCellRenderers() (*vi.widgets.table.DataTable* method), 42  
 setContext() (*vi.widgets.list.ListWidget* method), 37  
 setContext() (*vi.widgets.ListWidget* method), 49  
 setCurrentModulDescr() (*vi.widgets.topbar.TopBarWidget* method), 44  
 setCurrentModulDescr() (*vi.widgets.TopBarWidget* method), 49  
 setCursorRow() (*vi.framework.components.datatable.SelectTable* method), 19  
 setCursorRow() (*vi.widgets.table.SelectTable* method), 41  
 setData() (*vi.widgets.edit.EditWidget* method), 31  
 setData() (*vi.widgets.EditWidget* method), 50  
 setData() (*vi.widgets.repeatdate.RepeatDatePopup* method), 38  
 setDataProvider() (*vi.framework.components.datatable.DataTable* method), 20  
 setDataProvider() (*vi.widgets.DataTable* method), 51  
 setDataProvider() (*vi.widgets.table.DataTable* method), 41  
 setFields() (*vi.widgets.list.ListWidget* method), 37  
 setFields() (*vi.widgets.ListWidget* method), 50  
 setFile() (*vi.widgets.file.FilePreviewImage* method), 32  
 setFilter() (*vi.widgets.list.ListWidget* method), 37  
 setFilter() (*vi.widgets.ListWidget* method), 49  
 setFinalElem() (*vi.priorityqueue.StartupQueue* method), 64  
 setHeader() (*vi.framework.components.datatable.SelectTable* method), 19  
 setHeader() (*vi.widgets.table.SelectTable* method), 40  
 setItemImage() (*vi.pane.Pane* method), 62  
 setListView() (*vi.widgets.hierarchy.HierarchyWidget* method), 34  
 setListView() (*vi.widgets.HierarchyWidget* method), 56  
 setPage() (*vi.widgets.list.ListWidget* method), 36  
 setPage() (*vi.widgets.list.ViewportListWidget* method), 37  
 setPage() (*vi.widgets.ListWidget* method), 49



- setPageAmount() (*vi.actions.list.SetPageRowAmountAction* method), 13  
 setPageRowAmountAction (*class in vi.actions.list*), 13  
 setPath() (*vi.Application* method), 69  
 setPayed() (*vi.actions.list\_order.ShopMarkAction* method), 15  
 setPayedFailed() (*vi.actions.list\_order.ShopMarkAction* method), 15  
 setPayedSucceeded() (*vi.actions.list\_order.ShopMarkAction* method), 15  
 setPreventUnloading() (*in module vi.utils*), 66  
 setRootNode() (*vi.widgets.tree.TreeWidget* method), 46  
 setRootNode() (*vi.widgets.TreeWidget* method), 55  
 setSelector() (*vi.widgets.list.ListWidget* method), 36  
 setSelector() (*vi.widgets.ListWidget* method), 49  
 setSelector() (*vi.widgets.tree.TreeWidget* method), 45  
 setSelector() (*vi.widgets.TreeWidget* method), 54  
 setShownFields() (*vi.framework.components.datatable.DataTable* method), 21  
 setShownFields() (*vi.widgets.DataTable* method), 52  
 setShownFields() (*vi.widgets.table.DataTable* method), 42  
 setStyle() (*vi.widgets.file.FileLeafWidget* method), 33  
 setStyle() (*vi.widgets.file.FileNodeWidget* method), 33  
 setStyle() (*vi.widgets.tree.BreadcrumbNodeWidget* method), 46  
 setStyle() (*vi.widgets.tree.BrowserLeafWidget* method), 46  
 setStyle() (*vi.widgets.tree.BrowserNodeWidget* method), 46  
 setTableActionBar() (*vi.widgets.list.ListWidget* method), 36  
 setTableActionBar() (*vi.widgets.list.ViewportListWidget* method), 38  
 setTableActionBar() (*vi.widgets.ListWidget* method), 49  
 setText() (*vi.pane.Pane* method), 62  
 setTitle() (*vi.Application* method), 69  
 setTitle() (*vi.screen.Screen* method), 65  
 setTitle() (*vi.widgets.topbar.TopBarWidget* method), 44  
 setTitle() (*vi.widgets.TopBarWidget* method), 49  
 setUrl() (*vi.widgets.preview.Preview* method), 38  
 setView() (*vi.sidebarwidgets.filtersselector.FilterSelector* method), 22  
 setWidget() (*vi.widgets.SideBar* method), 52  
 setWidget() (*vi.widgets.sidebar.SideBar* method), 39  
 ShopMarkAction (*class in vi.actions.list\_order*), 15  
 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), 31  
 showErrorMsg() (*vi.widgets.EditWidget* method), 50  
 showErrorMsg() (*vi.widgets.list.ListWidget* method), 36  
 showErrorMsg() (*vi.widgets.ListWidget* method), 49  
 showErrorMsg() (*vi.widgets.repeatdate.RepeatDatePopup* method), 38  
 showErrorMsg() (*vi.widgets.tree.TreeWidget* method), 45  
 showErrorMsg() (*vi.widgets.TreeWidget* method), 55  
 showListView() (*vi.widgets.hierarchy.HierarchyWidget* method), 34  
 showListView() (*vi.widgets.HierarchyWidget* method), 56  
 showLoginWindow() (*vi.widgets.UserLogoutMsg* method), 53  
 showLoginWindow() (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), 47  
 showErrorMessage() (*vi.widgets.UserLogoutMsg* method), 53  
 showMessage() (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), 47  
 SideBar (*class in vi.widgets*), 52  
 SideBar (*class in vi.widgets.sidebar*), 39  
 singletonHandler (*class in vi.views.singleton*), 26  
 singletonHandlerWidget (*class in vi.views.singleton*), 26  
 stackWidget() (*vi.admin.AdminScreen* method), 58  
 stackWidget() (*vi.AdminScreen* method), 69  
 start() (*in module vi*), 69  
 startFind() (*vi.actions.list.PageFindAction* method), 14  
 startPolling() (*vi.widgets.UserLogoutMsg* method), 53  
 startPolling() (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), 47  
 startup() (*vi.admin.AdminScreen* method), 57  
 startup() (*vi.AdminScreen* method), 68  
 startup() (*vi.Application* method), 69  
 startupFailure() (*vi.Application* method), 69  
 StartupQueue (*class in vi.priorityqueue*), 64  
 startupQueue (*in module vi.priorityqueue*), 64  
 stopInterval() (*vi.widgets.UserLogoutMsg* method), 53  
 stopInterval() (*vi.widgets.userlogoutmsg.UserLogoutMsg* method), 47  
 style (*vi.widgets.edit.ParsedErrorItem* attribute), 30  
 style (*vi.widgets.edit.PassiveErrorItem* attribute), 30  
 style (*vi.widgets.internaledit.ParsedErrorItem* attribute), 35  
 style (*vi.widgets.internaledit.PassiveErrorItem* attribute), 35  
 switchDisabledState() (*vi.serversideaction.ServerSideActionWdg* method), 65

- switchFullscreen() (*vi.admin.AdminScreen* method), 58
- switchFullscreen() (*vi.AdminScreen* method), 69
- ## T
- tableInitialization() (*vi.widgets.list.ListWidget* method), 36
- tableInitialization() (*vi.widgets.list.ViewportListWidget* method), 37
- tableInitialization() (*vi.widgets.ListWidget* method), 49
- 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*), 44
- TaskSelectWidget (class in *vi.widgets*), 53
- TaskSelectWidget (class in *vi.widgets.task*), 43
- TaskWidget (class in *vi.widgets*), 53
- TaskWidget (class in *vi.widgets.task*), 43
- testIfNextBatchNeededImmediately() (*vi.widgets.DataTable* method), 51
- testIfNextBatchNeededImmediately() (*vi.widgets.table.DataTable* method), 41
- TextInsertImageAction (class in *vi.widgets.htmleditor*), 34
- toggle() (*vi.widgets.accordion.AccordionSegment* method), 27
- toggleIntPrev() (*vi.actions.list.ListPreviewInlineAction* method), 12
- toggleListView() (*vi.widgets.hierarchy.HierarchyWidget* method), 34
- toggleListView() (*vi.widgets.HierarchyWidget* method), 56
- toggleMsgCenter() (*vi.log.Log* method), 60
- ToolTip (class in *vi.widgets*), 51
- ToolTip (class in *vi.widgets.tooltip*), 43
- TopBarWidget (class in *vi.widgets*), 48
- TopBarWidget (class in *vi.widgets.topbar*), 44
- toplevelActionSelector (in *vi.priorityqueue* module), 64
- tpl (*vi.widgets.appnavigation.NavigationElement* attribute), 28
- TreeBrowserWidget (class in *vi.widgets*), 55
- TreeBrowserWidget (class in *vi.widgets.tree*), 46
- treeHandler (class in *vi.views.tree*), 26
- treeHandlerWidget (class in *vi.views.tree*), 27
- TreeWidget (class in *vi.widgets*), 54
- TreeWidget (class in *vi.widgets.tree*), 45
- ## U
- unlock() (*vi.login.BaseLoginHandler* method), 61
- unlock() (*vi.pane.Pane* method), 62
- unlock() (*vi.screen.Screen* method), 65
- unselectAll() (*vi.framework.components.datatable.SelectTable* method), 20
- unselectAll() (*vi.widgets.table.SelectTable* method), 41
- UnselectAllAction (class in *vi.actions.list*), 15
- unserialize() (*vi.widgets.edit.EditWidget* method), 31
- unserialize() (*vi.widgets.EditWidget* method), 51
- unserialize() (*vi.widgets.InternalEdit* method), 53
- unserialize() (*vi.widgets.internaledit.InternalEdit* method), 36
- update() (*vi.actions.hierarchy.SelectRootNode* method), 9
- update() (*vi.actions.tree.SelectRootNode* method), 17
- update() (*vi.framework.components.datatable.DataTable* method), 20
- update() (*vi.framework.components.datatable.ViewportDataTable* method), 21
- update() (*vi.widgets.topbar.Tasks* method), 44
- update() (*vi.widgets.topbar.UserState* method), 44
- updateConf() (in *vi.config* module), 58
- updateEmptyNotification() (*vi.widgets.list.ListWidget* method), 37
- updateEmptyNotification() (*vi.widgets.ListWidget* method), 50
- Uploader (class in *vi.widgets.file*), 32
- UserLogoutMsg (class in *vi.widgets*), 52
- UserLogoutMsg (class in *vi.widgets.userlogoutmsg*), 47
- UserPasswordLoginHandler (class in *vi.login*), 61
- UserState (class in *vi.widgets.topbar*), 44
- ## 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, 15
- vi.actions.tree module, 16
- vi.admin module, 57
- vi.config module, 58
- vi.exception

- module, 59
- vi.framework
  - module, 18
- vi.framework.components
  - module, 18
- vi.framework.components.actionbar
  - module, 18
- vi.framework.components.datatable
  - module, 18
- vi.log
  - module, 59
- vi.login
  - module, 60
- vi.pane
  - module, 62
- vi.priorityqueue
  - module, 63
- vi.screen
  - module, 65
- vi.serversideaction
  - module, 65
- vi.sidebarwidgets
  - module, 22
- vi.sidebarwidgets.filterselector
  - module, 22
- vi.sidebarwidgets.internalpreview
  - module, 22
- vi.translations
  - module, 23
- vi.translations.de
  - module, 23
- vi.translations.en
  - module, 23
- vi.utils
  - module, 66
- vi.views
  - module, 23
- vi.views.edit
  - module, 23
- vi.views.hierarchy
  - module, 24
- vi.views.list
  - module, 24
- vi.views.log
  - module, 25
- vi.views.notfound
  - module, 25
- vi.views.overview
  - module, 25
- vi.views.singleton
  - module, 26
- vi.views.tree
  - module, 26
- vi.widgets
  - module, 27
- vi.widgets.accordion
  - module, 27
- vi.widgets.appnavigation
  - module, 28
- vi.widgets.csvexport
  - module, 29
- vi.widgets.edit
  - module, 30
- vi.widgets.file
  - module, 32
- vi.widgets.hierarchy
  - module, 34
- vi.widgets.htmleditor
  - module, 34
- vi.widgets.internaledit
  - module, 35
- vi.widgets.list
  - module, 36
- vi.widgets.preview
  - module, 38
- vi.widgets.repeatdate
  - module, 38
- vi.widgets.search
  - module, 39
- vi.widgets.sidebar
  - module, 39
- vi.widgets.table
  - module, 40
- vi.widgets.task
  - module, 42
- vi.widgets.tooltip
  - module, 43
- vi.widgets.topbar
  - module, 43
- vi.widgets.tree
  - module, 44
- vi.widgets.userlogoutmsg
  - module, 47
- vi\_conf (in module vi), 68
- vi\_conf (in module vi.config), 58
- ViewportDataTable (class in vi.framework.components.datatable), 21
- ViewportListWidget (class in vi.widgets.list), 37
- viInitializedEvent (in module vi.admin), 58
- visibilityChanged() (vi.widgets.UserLogoutMsg method), 53
- visibilityChanged() (vi.widgets.userlogoutmsg.UserLogoutMsg method), 47