

What's New in OpenFaces 2.0

General Information

[OpenFaces](#) is an open-source library of AJAX-powered JSF components, an [Ajax framework](#) and a client-side [validation framework](#). OpenFaces is based on the set of JSF components formerly known as QuipuKit library. Here's a list of components inherited from the QuipuKit library: Calendar, Chart, Confirmation, DataTable, DateChooser, DynamicImage, FoldingPanel, GraphicText, HintLabel, PopupLayer, SuggestionField, TabSet, TabbedPane, TreeTable, TwoListSelection, Utility Components (Focus, ScrollPosition, LoadBundle). See the [OpenFaces Components](#) page for the full description of these components. The rest of this document describes the changes and new features added in OpenFaces version 2.0 since the last release of QuipuKit (version 1.6.2).

Ajax Framework

OpenFaces 2.0 introduces an [Ajax framework](#) which allows adding Ajax-based interactions in cases when components' built-in Ajax capabilities are not enough. OpenFaces Ajax framework addresses a common scenario where the whole page has to be submitted to perform some application-specific action and reload the page with the results. OpenFaces Ajax framework allows resolving such use cases without full page reload using a new non-visual component called Ajax, and an appropriate client-side API. The main purpose of this component is reloading one or more components with Ajax, plus it provides additional features such as choosing the components whose data should be submitted to the server, invoking a server action as part of the Ajax request, and handling frequent requests.

Here's a simple example which demonstrates the usage of the new component:

```
<h:inputText id="in" value="#{requestScope['ajaxText']}" />
<h:commandButton value="Update">
  <o:ajax execute="in" render="out" />
</h:commandButton>
<h:outputText id="out" value="#{requestScope['ajaxText']}" />
```

You can see Ajax Framework in action in [OpenFaces demo](#).

In addition to the declarative tag-based syntax, you can use a client-side API that provides access to the same functionality for a JavaScript code. OpenFaces also allows customizing various aspects of Ajax framework using the `AjaxSettings` component, which can be used to customize Ajax progress message and the way of handling session expiration during Ajax requests. See the [documentation](#) for more information.

New Components

- **CommandButton** - an extended version of the standard `<h:commandButton>` which includes the Ajax features and an ability to customize button's content with sub-components or HTML.
- **CommandLink** - an extended version of the standard `<h:commandLink>` which includes the Ajax features.
- **CompositeFilter** - allows the user to build complex filter criteria with multiple filter conditions. Works as a standalone component or can be bound to `DataTable/TreeTable` components to filter their data.
- **BorderLayoutPanel** - a container used to layout groups of components by container's edges. The separators between component groups can either be fixed to introduce a static layout, or can be made draggable and/or collapsible to make a dynamic layout.
- **DayTable** - a component that allows displaying and editing schedule of events for a day. Each event has a start time, end time, event name, description and other fields. There is also the possibility to show timetable for multiple resources, for example the schedule of tasks for several people.
- **ForEach** - an iterator component that renders the specified set of components multiple times based on its parameters. The `ForEach` component is similar to the JSTL `<c:forEach>` tag, though it can be used to overcome some of the shortcomings of `<c:forEach>` tag and to eliminate the need for an additional JSTL dependency.
- **InputText** - supports all of the features of the standard JSF `InputText` component and extends it with some additional features like rollover and focused styles and prompt text.
- **InputTextarea** - supports all of the features of the standard JSF `InputTextarea` component and extends it with some additional features like rollover and focused styles and prompt text.
- **LayeredPane** - a container that allows switching between different sets of displayed components.
- **PopupMenu** - a component that displays a set of actions in a pop-up box for a user to choose from, and is often used as a context menu. The `PopupMenu` component can be attached to any component to appear when any particular event occurs in that component. A `PopupMenu` can be configured to have sub-menus to represent complex hierarchies of actions.
- **SelectBooleanCheckbox** - an extended version of the standard `<h:selectBooleanCheckbox>` component that adds such possibilities as tri-state support, customizing checkbox images and state-dependent styles.
- **SelectOneRadio** - an extended version of the standard `<h:selectOneRadio>` component, which provides additional styling options and makes it possible to customize radio buttons with images.

- **SelectManyCheckbox** - an extended version of the standard `<h:selectManyCheckbox>` component, which provides additional styling options and makes it possible to customize radio buttons with images.
- **Spinner** - an input field for entering numbers, which consists of a text field and the attached increase/decrease buttons.
- **Window** - a window-styled container for other JSF components that is displayed over the page content. The Window component has a caption and a content area, it can be dragged around on the page, and resized. Besides, it has the usual maximize/restore, minimize and close buttons in the caption.

New Features

- **Ajax settings:**
 - It's now possible to customize Ajax progress message position with new `horizontalAlignment/verticalAlignment` attributes of `<o:defaultProgressMessage>` tag.
 - Ajax progress message can now be made semitransparent and shown with a smooth transparency transition (see `<o:defaultProgressMessage>` tag documentation).
 - Added an option to disable mouse interoperation by shading window contents while Ajax request is in progress (see `<o:defaultProgressMessage>` tag documentation).
 - It's now possible to customize the appearance of session expiration confirmation (see the `<o:defaultSessionExpiration>` tag documentation).
- **DataTable/TreeTable:**
 - Added support for content scrolling with header/footer row(s) freezing for DataTable and TreeTable components. Both vertical and horizontal scrolling with fixed columns is supported. See the [Content Scrolling](#) section in DataTable and TreeTable documentation for details.
 - Added support for interactive drag & drop column reordering. See the [Drag & Drop Column Reordering](#) section in DataTable and TreeTable documentation for details.
 - Added support for interactive column visibility customization. See the [Displaying Column Menus](#) section in DataTable and TreeTable documentation for details.
 - Numerous filtering enhancements, see the "DataTable/TreeTable Filtering Enhancements" section below.
 - Added support for specifying a popup menu for table columns, containing either the standard or customized set of actions.
 - Added header and footer attributes to all column tags to serve as a convenient replacement for specifying the text in the "header" and "footer" facets.
 - Significantly simplified handling large data sets using Hibernate library. See the [Using DataTable with the Hibernate Library](#) section in DataTable documentation for details.
 - There's a new "subHeader" facet in the column tags for DataTable and TreeTable components that allows customizing the contents that appear in the table's header section under the column headers.
 - Added a new `DataTable.getPageIndexForRowKey` method for scenarios like detecting the page(s) for selected item(s).
 - Improved TreeTable configuration checking and error reporting.
 - DataTable is now automatically focused when the pagination buttons are used.
 - Resolved the known issue where editable and command components couldn't be used in the Ajax-enabled TreeTable.
- **TabbedPane:**
 - Added a new loading mode "ajaxAlways", which allows loading a fresh tab content on each tab switch. Also note the API change: the "ajax" mode has been renamed to "ajaxLazy".
- **FoldingPanel:**

- Added a new loading mode "ajaxAlways", which allows loading a fresh content on each expansion of FoldingPanel. Also note the API change: the "ajax" mode has been renamed to "ajaxLazy".
- FoldingPanel's expansion state is now properly saved when placed inside of iteration components such as `<o:dataTable>` and `<o:forEach>`.
- **Other:**
- Extended validation framework: added the possibility for annotation-based validation with `<o:validateAll>` tag.
- Added the caption attribute for `<o:foldingPanel>` and `<o:subPanel>` (former `<o:tabbedPanelItem>`) tags, which is a short replacement for the "caption" facet if you need to specify the caption as text.
- Added oncontextmenu event for all components.
- Added built-in Ajax support to `<o:singleRowSelection>`, `<o:multipleRowSelection>`, `<o:singleNodeSelection>`, and `<o:multipleNodeSelection>` tags with render and execute attributes.
- Improved JavaScript performance for the DataTable/TreeTable components.
- All popups in components like DropDownField, SuggestionField, Calendar, DateChooser, PopupMenu can now be closed by pressing Esc key.
- Added support for handling mouse events in `<o:floatingIconMessage>` tag.
- Improved error checking when processing default validation message presentation parameters in web.xml.
- A new simpler way of attaching Confirmation to components: now you can simply add a Confirmation component inside of other component (like it can be done with `<o:ajax>` tag) and it will automatically be attached to the "onclick" event of the parent component. The event to which confirmation is attached can be customized with the "event" attribute.
- New "draggableByContent" attribute for `<o:confirmation>` tag (false by default). If confirmation is configured to be draggable, chooses whether the confirmation window can be dragged by window caption (the default behavior), or by any part of the confirmation window.
- Added an ability to place custom caption buttons into Confirmation and FoldingPanel components.
- Added an ability to place custom caption buttons into FoldingPanel's caption
- Added pressedButtonStyle/pressedButtonClass attributes for DropDownButton and DateChooser.
- Added focusedStyle/focusedClass for all input components.
- Added tabIndex attribute for TwoListSelection component.
- Added standard client events to DynamicImage component.
- Added mouse events to GraphicText component.
- Extended client-side API for TabSet and TabbedPane components: added getTabCount and getPageCount functions respectively.
- Extended client-side API for TwoListSelection component: added selectAll and unselectAll functions.
- Added typingAllowed attribute for DateChooser component to enable scenario where a date should only be selectable from drop-down calendar.
- FoldingPanel's expansion state is now correctly saved when placed inside of iteration components such as `<o:dataTable>` and `<o:forEach>`.
- DataTable/TreeTable filter fields' text is now not marked with selection for convenient typing after filtering.
- Rollover button style for DateChooser and DropDownField is now applied when mouse is over the button itself rather than over the field.

DataTable/TreeTable Filtering Enhancements

The filtering API for DataTable and TreeTable components has been significantly reviewed and enhanced. The previous filtering API consisted of filtering-related attributes in `<o:column>` tag. Although this was enough for a simple API, different kinds of filters require different sets of attributes and additional features require

much more attributes, so filters are now organized as separate tags – one tag per one type of filter: `<o:inputTextFilter>`, `<o:dropDownFieldFilter>`, and `<o:comboBoxFilter>`. Each tag has a set of attributes applicable for that particular type of filter. It's quite easy to migrate to the new API though, see the "Migrating to the New Filtering API" sub-section in the [Migrating from QuipuKit 1.6.2](#) document.

Here are the new features that were implemented as part of the new API:

- It's possible to customize whether search should be case sensitive or not.
- It's now possible to customize the condition by which search has to be performed (contains, equals, less, greater, etc.).
- It's now possible to specify style and all other component-specific attributes of filtering components.
- There's now an option (turned on by default) that makes filtering to be performed on the fly as the user types in the filter field.
- DropDownFieldFilter now suggests filter text as the user types it.
- Filters can now be placed outside of their DataTable/TreeTable components.
- The filtering expression can now be detected automatically when a filter resides in a table column.

Please see the [Filtering](#) section in the DataTable and TreeTable documentation for the comprehensive usage details.

Extended Keyboard Support

OpenFaces 2.0 has extended keyboard support for its components. The following components now include keyboard support:

- Calendar
- CommandButton
- CommandLink
- Confirmation
- CompositeFilter
- DataTable
- DateChooser
- DropDownField
- InputText (native keyboard support)
- InputTextarea (native keyboard support)
- FoldingPanel
- PopupMenu
- SelectBooleanCheckbox
- SelectOneRadio
- SelectManyCheckbox
- Spinner
- SuggestionField
- TabSet
- TabbedPane
- TreeTable
- TwoListSelection (native keyboard support)

Migrating from QuipuKit 1.6.2

Please see the [Migrating from QuipuKit 1.6.2](#) document for the migration instructions.

-