Eclipse Scout

Release Notes

Judith Gull, Matthias Villiger

Version 6.0.0-SNAPSHOT

Table of Contents

!!! WORK IN PROGRESS !!!

Release Notes for Scout 6.0

Here are the release notes for the Scout Release 6.0

TODO: a lot is still missing!

What's New

HTML

Use the class `org.eclipse.scout.rt.platform.html.HTML` to build HTML content instead of concatenate the strings manually.

Security: MalwareScanner

Facility used to scan files and resources for malware. The new @Bean MalwareScanner assumes that an appropriate malware scanner is in place on the webapp deployment machine and is configured to scan the TEMP folder (as used by File#createTempFile) using a realtime filesystem scan strategy. Malware should therefore immediately be removed or blocked by the malware implementation when placed in that folder. The new MalwareScanner is used in the ui.html file upload handler and thus checks every uploaded file.

IMimeTypeDetector [New]

The new interface IMimeTypeDetector provides multiple ordered implementations that can detect mime types PrimaryMimeTypeDetector with order 0 defines important webapp mime types ServletContextMimeTypeDetector with order 10 uses ServletContext.getMimeType JavaNioMimeTypeDetector uses java.nio Files.probeContentType

UploadRequestHandler [Added features]

The UploadRequestHandler checks for malware and limits the file types that can be uploaded.

API Changes

Table

- Behavior change of `Table` in `AbstractTableField`: Do not execute `AbstractTable.execContentChanged()` when `valueChangeTriggers` flag on `IFormField` is `false`.
- Returned collection of `ITableColumnFilterManager.getFilters()` is now unmodifiable.
- New method `ITableColumnFilterManager.removeFilter(IColumn col)`.

Tree

- `ITree.getConfiguredDefaultIconId()` in addition to `ITree.getConfiguredIconId`. The difference is as follows: *DefaultIconId* is used as default for all tree nodes that don't have an icon on their own. *IconId* may be used in the same way as the title, e.g as outline icon.
- Nodes may now be expanded in a lazy way. This means only those child nodes are visible which are expanded as well and the parent gets a '+' symbol. If the user clicks on this symbol all child nodes gets visible. The model can define whether child pages of a page should be added immediately to the outline tree or lazily. If nodes are added lazily, a dummy "show all" node is shown instead.

  Node pages never add child pages lazily. Table pages add child nodes lazily when they have more than a specific number of child pages (default 1).

  The behavior may be controlled using:

  - `boolean getConfiguredLazyAddChildPagesToOutline()` -→ default `false`, for `AbstractPageWithTable` the default is `true`.
  - `int getConfiguredLazyAddChildPagesToOutlineThreshold()` -→ setting for `AbstractPageWithTable`, after how many child pages the lazy setting should be active (default 1)

Chart Box

Removed chart box because it has not been used and there is no UI implementation.

File Chooser (File Upload)

File upload size is always limited now (otherwise server might run out of memory if too large files are sent). Default size is 50 MB, but every field might specify lower/higher sizes:

- `org.eclipse.scout.rt.client.ui.IDNDSupport.setDropMaximumSize(long)`
- `org.eclipse.scout.rt.client.ui.IDNDSupport.getDropMaximumSize()`
- `org.eclipse.scout.rt.client.ui.basic.filechooser.IFileChooser.setMaximumUploadSize(long)`
- `org.eclipse.scout.rt.client.ui.basic.filechooser.IFileChooser.getMaximumUploadSize()`
- `org.eclipse.scout.rt.client.ui.form.fields.filechooserfield.IFileChooserField.setMaximumUploadSize(long)`
- `org.eclipse.scout.rt.client.ui.form.fields.filechooserfield.IFileChooserField.getMaximumUploadSize()`
- Also added `getConfigured…()` methods were applicable for properties above.

Group Box

`AbstractGroupBox.setBorderVisible(false)` does not change visibility of label anymore. Label of group-box must be made invisible by calling `setLabelVisible(false)`.

Smart Field

- `AbstractContentAssistField` and `IProposalChooser` implementations now have the ability to provide an inner class which extend `AbstractTree` or `AbstractTable` to provide a custom implementation used in the proposal chooser.
- Added `cssClass` property to formField, column and cell for custom css styling. See also interface `IStyleable` and class `CssClasses`.

Date Field

The date field no longer inherits from *BasicField*. Instead it inherits directly from *ValueField*. This means that the `PROP_UPDATE_DISPLAY_TEXT_ON_MODIFY` is no longer supported on date fields. The reason for this change was the separation of UI (Browser) and UI-Server (Java). To get a good performance, fast date predictions and offline capability, the parsing must be done in the UI and not on the server. Because the "update on modify" flag had no effect anyway, it was completely removed from the UI. (More details in the Migration Guide).

Split Box

- `AbstractSplitBox` now returns `IFormField.FULL_SIZE` in `getConfiguredGridW()` by default. Reason: The split box widget does not really have a representation of its own, but is more like a container for other fields. It can never have label, mandatory indicator etc. Its layout should behave like a group box or a tab box, therefore the default gridW value was adjusted accordingly.
- Split boxes now support absolute splitter positions. The old relative position is the default, which uses a value between 0 and 1 for the `splitterPosition`. By changing the property `splitterPositionType`, the interpretation of the `splitterPosition` value can be changed to pixels (either fixed for the first or the second inner box).
- `AbstractSplitBox` now provides a *collapse* state for one of the two fields. When a field is marked as collapsible, the UI shows a toggle-button which allows to collapse and expand that field. The following methods and configurations are new:
  - `Class<? extends IFormField> getConfiguredCollapsibleField()` returns the class of the field which should be collapsible. Default is null.
  - `boolean getConfiguredFieldCollapsed()` returns whether or not the field is initially collapsed. Default is false.
  - `String getConfiguredCollapseKeyStroke()` returns the key-stroke used to trigger the collapse button. Default is null.
  - At runtime use the methods `setCollapsedField(IFormField)`, `setFieldCollapsed(boolean)` and `setCollapseKeyStroke(String)` to change the properties described above.

Desktop / Outline

- Menus of a page are now added to the detail form. This was necessary because the outline tree does not show any menus anymore. See also method `AbstractPageWithNodes.enhanceDetailFormWithPageMenus`.

- Added `AbstractDesktop.getConfiguredAutoTabKeyStrokesEnabled`: It should be possible to change view Tabs with modifier+number. The number should be generated by the ui. 9 is reserved to jump to the last tab, 0 to jump to the first tab. If this property is set to false there is no Keystroke for tab change on the ui.
  - `getConfiguredAutoTabKeyStrokeModifier`: if the property is set to `true` the modifier specified by this property is used in combination with a number to change to the specific tab.
- Added `ISearchOutline`: The intention of the search outline is to provide a search over several table pages. The `AbstractSearchOutline` provides a frame, the search itself has to be implemented by the project. In order to use it add the `SearchOutline` to the desktop using `getConfiguredOutlines` (don't create an outlineViewButton).
- Added default detail form on outline: It is now possible to configure a default detail form for outlines. The default detail form gets shown when no page is selected. API added `getConfiguredDefaultDetailForm`, `execInitDefaultDetailForm`, `createDefaultDetailForm`, `startDefaultDetailForm`.
- Added `getConfiguredTableStatusVisible` on `IPageWithTable`: It is now possible to configure whether the table status should be visible for a table page. Until now table status was set visible by the `OutlineTableForm`.
- API added to `AbstractTable` and `ITable`:
  - `List<ITableControl> getTableControls()`
  - `<T extends ITableControl> T getTableControl(Class<T> controlClass)`
  - `boolean isTableStatusVisible()`
  - `void setTableStatusVisible(boolean visible)`
  - `String getMenuBarPosition()`
  - `void setMenuBarPosition(String position)`
- API added to ITableUIFacade
  - `void fireTableReloadFromUI()`
  - `void fireSortColumnRemovedFromUI(IColumn<?> column)
- Improved page search form disposal: Search form is now closed when the page gets disposed.
- `AbstractDesktop.isOutlineChanging` added.
- The methods `traverseFocusNext()` and `traverseFocusPrevious` were removed from *IDesktop* because traversing is not supported by the HTML UI. (Neither was it supported by the former RAP UI). The corresponding *DesktopEvent* types (`TYPE_TRAVERSE_FOCUS_NEXT`, `TYPE_TRAVERSE_FOCUS_PREVIOUS`) were removed as well.

Form

Added `IForm.start()`: Mainly useful for forms with just one handler. (detail forms, tool forms etc.). May be implemented by the concrete form. The default implementation at AbstractForm uses getHandler() to start the form.

Bean Field / Bean Column [New]

Added bean field and bean column. See `AbstractBeanField`, `AbstractBeanColumn` for details.

Wizard Progress Field [New]

A new widget has been added: `WizardProgressField`. It is normally visualized as a list of steps with some indication which step is the current step etc. It replaces the old "HTML status" field on the default wizard container form.

HTMLUtility

`org.eclipse.scout.rt.platform.html.HTMLUtility` and `org.eclipse.scout.rt.platform.html.CSSPatch` were deprecated because they contained only legacy code of doubtful quality. Do not use them anymore — they will be removed in the next Scout release. The support for most of the contained methods was dropped, because they are not required anymore with the new UI. A slightly improved version of the `getPlainText()` method is available on the new bean `org.eclipse.scout.rt.platform.html.HtmlHelper`.

Various Changes

- `ILabelField`, `IHtmlField` and `IStringField` are now `IHtmlCapable`.
- `UiLayer`: Removed values `JSP`, `JSF`, `RAP`, `SWING` and added value `HTML`.
- `UserAgentUtility`: API removed `isRapUi()`, `isSwingUi()`
- `AbstractFormField.set/isStatusVisible` added.
- `org.eclipse.scout.commons` reorganized, classes moved to `org.eclipse.scout.rt.platform` and other projects
- The unused, obsolete classes `org.eclipse.scout.rt.client.ui.form.fields.ValueFieldEvent` and `org.eclipse.scout.rt.client.ui.form.fields.ValueFieldListener` were removed.

Moved to correct module

move org.eclipse.scout.rt.server.commons.authentication.ConfigFileCredentialVerifier to org.eclipse.scout.rt.platform.security.ConfigFileCredentialVerifier
move org.eclipse.scout.rt.server.commons.authentication.ICredentialVerifier to org.eclipse.scout.rt.platform.security.ICredentialVerifier
move org.eclipse.scout.rt.server.commons.authentication.IPrincipalProducer to org.eclipse.scout.rt.platform.security.IPrincipalProducer
move org.eclipse.scout.rt.server.commons.authentication.SimplePrincipalProducer to org.eclipse.scout.rt.platform.security.SimplePrincipalProducer

URL Hints (security)

The new config property `scout.urlHints.enabled` controls if the URL parameters ?cache, ?compress, ?minify, ?debug, ?inspector are active. By default these are not enabled. This means, that by default it is not possible (for an attacker) to disable minify and get the original javascript sources.

Known Bugs and Limitations

Selection in disabled text fields

Firefox (tested with v 35.0.1): it's not possible to select text in a disabled text field. There is no workaround. An often suggested workaround is to use the readonly attribute instead of disabled, but this leads to a completely different behavior when it comes to event handling. When a field is disabled it doesn't fire any events (for instance: click), but when it's set to readonly event handling is still enabled. In other words: we'd have to implement browser/widget default behavior to work around this problem — additionally the workaround should only affect Firefox but none of the other browsers (where this is not an issue at all).

Drag and drop support

IE9 does not support dropping files. Therefore, drag and drop in file chooser is not supported for that browser.

Input placeholders

On-field labels are not visible in IE9 because it lacks support for the "placeholder" attribute.

HttpServiceTunnel

It is now possible again to call services on a different scout server by specifying server org.eclipse.scout.rt.servicetunnel.targetUrl and scout.auth.privatekey. For all interfaces annotated with @TunnelToServer without implementation, a proxy is created.

Session Cookie (JSESSIONID Cookie) configuration validation

The HTML UI checks if the application is configured safe by validating some flags set on the session cookie. For more details on how to configure your session cookie please refer to the Scout Documentation chapter "Session Cookie (JSESSIONID Cookie) Configuration".

UserIdAccessControlService [Removed]

`org.eclipse.scout.rt.shared.services.common.security.UserIdAccessControlService` was removed in M7. When creating a new Scout project via SDK, an own implementation of AbstractAccessControlService is now explicitly created in shared, named `AccessControlService` (as replacement for the removed `UserIdAccessControlService`). The class generated at the server side now replaces (@Replace) the new one generated in shared (class name of server side changed from `AccessControlService` to `ServerAccessControlService`).

Do you want to improve this document? Please edit this page on GitHub.