Eclipse Scout: Release Notes

Table of Contents

⚠️ This document is referring to a past Scout release. Please click here for the recent version.
Looking for something else? Visit https://eclipsescout.github.io for all Scout related documentation.

About This Release

Eclipse Scout 6.1 is a preview of the Eclipse *Oxygen* release (release schedule). The latest public

build on this branch is *6.1.0.B009*. The official release for Eclipse Oxygen is Eclipse Scout 7.0 and will be released in June 2017.

This document shows some of the new features delivered with the release 6.1. The release contains a lot of bugfixes and even some features not mentioned here.

You can see the [detailed change log](#) on GitHub.

Obtaining the Latest Version

Runtime (Scout RT)

Scout RT artifacts are distributed via Maven:

- [6.1.0.B009](#) on *Maven Central*

Usage example in the parent POM of your Scout application:

```xml
<dependency>
    <groupId>org.eclipse.scout.rt</groupId>
    <artifactId>org.eclipse.scout.rt</artifactId>
    <version>6.1.0.B009</version>
    <type>pom</type>
    <scope>import</scope>
</dependency>
```

Eclipse IDE Tooling (Scout SDK)

You can download the complete Eclipse IDE with Scout SDK included (EPP) here: [Eclipse for Scout Developers](#)

Strong Content Security Policy (CSP)

The stronger CSP disables inline javscript in html. Therefore the 'New Scout Project' wizard now creates a js file per html file and includes it using the script element. To migrate existing projects, see the [Scout Migration Guide](#).

Enforcement of Model Thread

Every operation which results in a modification of the model and eventually of the ui has to be performed by the model thread. This has been true for a long time and still is. If the wrong thread was used, unexpected behavior was the result, like a delayed update of the ui or concurrency exceptions. To prevent such behavior in the future, an exception will be thrown if an operation is executed in the wrong thread.

If you get such an exception, you'll need to wrap your operation in a model job and schedule it using `ModelJobs.schedule()`, see the chapter ModelJobs in the [Scout Technical Documentation](#) for

details.

Properties Support for Lists, Maps and Imports

Config properties support list- and map-data-structures. Furthermore other properties files can be imported. See the tech documentation section "Configuration Management" for more details.

BinaryResource Support for HtmlField and BeanField

Binary resources such as images or videos can now be used in the following widgets:

- HtmlField
- BeanField

Figure 1. Binary resource on a model field.

- Html enabled StringColumn
- BeanColumn

Figure 2. Binary resource on a column

TriState Capability for Check Boxes

Added support for tri-state value (`true`, `false` and `null` instead of just `true` and `false`) to boolean field and boolean column.

The new property `triStateEnabled` controls whether the boolean field/column behaves as a normal checkbox (false) or a tri-state checkbox (true).

A normal checkbox has values true/false. A tri-state checkbox has values true/false/null. The null value is interpreted as "undefined" and rendered as a filled square.

New UriOpenAction: SAME_WINDOW

The enum `UriOpenAction` provides a new value to open a URI in the current window: `SAME_WINDOW`

New Methods in StringUtility

`StringUtility` provides the following new methods:

- `containsString()`
- `containsStringIgnoreCase()`
- `containsRegEx()`
- `matches()`
- `endsWidth()`
- `startsWith()`
- `length()`
- `indexOf()`
- `lastIndexOf()`
- `split()` (with *limit* argument)

All methods are null-safe, unit tested and documented with JavaDoc.

New ObjectUtility

ObjectUtility was added as new utility for generic object methods and provides null-safe implementations of various Object methods. Various methods from former `CompareUtility`:

- `equals()`
- `notEquals()`
- `nvl()`
- `isOneOf()`
- `compareTo()`

And a new method `ObjectUtility.toString(Object)` providing a null-safe implementation of `Object.toString()` returning `null` if specified object is `null`.

Multiple Dimensions Support

Some components now support more dimensions for various attributes. E.g. until now there have been two dimensions for Form Field visibility: visible and visible-granted. Now there are also custom dimensions available. See the chapter 'Multiple Dimensions Support' in the Scout Technical Documentation for details and examples.

Currently the following attributes support multiple dimensions:

- Actions: visible, enabled
- Columns: visible
- Tree Nodes: visible, enabled
- Outlines: visible
- Form Fields: visible, enabled, label-visible
- Data Model Attributes: visible
- Data Model Entity: visible
- Wizard Steps: visible, enabled
- Trees: enabled
- Tables: enabled

Form Field

Enabled Inheritance

The inheritance of the enabled property for Form Fields has been changed. Now the enabled properties are no longer propagated to children if it is changed on a composite field. Instead a field is only considered to be enabled if itself and all of its parents are enabled. This allows to toggle an entire box to disabled and back to enabled without touching the child fields. This has the advantage that the original state is restored when the box is set back to enabled.

With this change the `getConfiguredEnabled` on composite fields now also automatically affects children. There is no need to overwrite `execInit()` and call `setEnabled(false)` anymore.

New Property "preventInitialFocus"

By default, the first enabled field on a form gets the focus when the form is opened. This may not be desired in some cases (e.g. if the first field is a HTML field that contains app links). The new property `PROP_PREVENT_INITIAL_FOCUS` can be used to prevent the initial focus to be set to this field. The default value is `false`. For `AbstractHtmlField` and `AbstractBeanField`, the default is set to `true`.

Menus in Tooltip

Form fields may have a tooltip and a context menu. Until now, it was not possible to display both simultaneously. This has been changed so that the menu items are now included in the tooltip if a tooltip text and menus are configured.
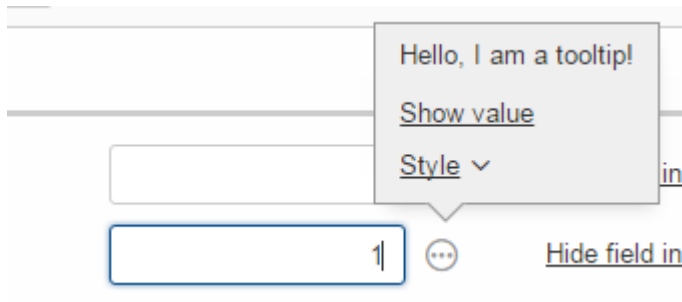

Figure 3. Tooltip with menus

Table

New Property "groupingStyle"

The new property `groupingStyle` can be set to `bottom` (default) or `top`. Depending on the value aggregate rows are rendered on the bottom of grouped rows or on the top of grouped rows. The new top style can be set to have an aggregate row as a title for a group of table rows, this is useful for separating a table into multiple categories.

NumberColumn: New Property "allowedAggregationFunctions", New Aggregation Type for no Aggregation

The new property `allowedAggregationFunctions` can be set to any array of the aggregation functions `sum`, `avg`, `min`, `max` and `none` (default: all aggregation functions are allowed). It defines the allowed aggregation functions for this number column (e.g. a sum aggregation is not always useful for all number columns). Also a new aggregation type `none` was introduced, with the new type it is possible to remove an aggregation from a column which has previously been aggregated. For the new aggregation type no additional button has been introduced, if the new aggregation type `none` is enabled an aggregation which previously has been used can be removed by using the same aggregation button again.

UnloadRequestHandler for `navigator.sendBeacon()`

When a client leaves the application (e.g. puts `about:blank` in the address bar) one last "unload" request to the UI server is sent in order to property clean up the session on the server.

If the browser supports the Beacon API `navigator.sendBeacon()` is used for this request.

Unfortunately `application/json` is not a CORS-safelisted request-header which implies that we can't use the `JsonMessageRequestHandler` for the unload handling. Therefore a separate `UnloadRequestHandler` was introduced which handles all requests to `/unload/[UiSessionId]`. (For more Information, see https://git.eclipse.org/r/#/c/89422/)

To cut a long story short, new traffic to /unload will be sent by the clients. Please check your container and firewall configuration.

New Annotation RemoteServiceWithoutAuthorization

Remote services called through IServiceTunnel may whitelist authorization exclusions using this new annotation.

Preparations for Scout JS

A classic Scout application has a client model written in Java, and a UI which is rendered using JavaScript. With this approach you can write your client code using a mature and type safe language. Unfortunately you cannot develop applications which have to run offline because the UI and the Java model need to be synchronized.

With Scout JS this will change. You will be able to create applications running without a UI server because there won't be a Java model anymore. The client code will be written using JavaScript (or TypeScript) and executed directly in the browser.

This release (6.1) is the first step in this direction. Several actions have been performed:

1. Created scout.App
   The new App object represents the *Single Page Application*. It will be initialized when the page loads and prepares all the necessary things the application needs to run, like texts, codes, fonts, logger and the session. These things may be different in case of a classic remote application and a Scout JS application. That is why there is another app called `scout.RemoteApp` which extends the `scout.App`. For you it basically means: if you create a Scout Classic App, use `scout.RemoteApp`, otherwise use `scout.App`.

2. Separated Widget and Model Adapter
   A `ModelAdapter` is the connector with the server, it takes the events sent from the server and calls the corresponding methods on the widget. It also sends events to the server whenever an action happens on the widget. So if there is no server, there is no need for such adapters. This means in a Scout JS app you will only work with widgets, adapters are only required for remote apps.

3. Enhanced Widgets
   With a Scout Classic app a lot happens on the UI server, like validating a form when the ok button is pressed. We started to enhance the JavaScript widgets with similar functionality and added API to use them. One example: The `ValueField` on Java side has a value and a display text. If a text is entered it will be parsed to get the value, or if a value is set the format function is called to get the text. This has not existed on JavaScript side, because the server only sent the text. This has been changed, parse and format functions now exist on the JS `ValueField` as well.

The preparations done in this release are just the first step. You could create a Scout JS app with this release, but a lot of the widgets are not ready to use yet. See also the Scout Migration Guide to migrate your existing JavaScript code.

Button: New Property "defaultButton"

A button may now be marked as default button which gives him a dedicated look to attract users attention. It will just change the look, the behavior stays the same.

Note: The first button or menu which has an `Enter` keystroke will automatically get that look too. This is existing behavior and hasn't changed. The new property has been added to give you more control, but actually you should always prefer the *enter keystroke approach* to provide a consistent behavior.

Icons for Tree Nodes

As in earlier Scout releases with Swing, SWT and RAP UI, the Outline and all Trees in Scout now support an icon per tree node. Simply set the `iconId` property on a TreeNode and reference either a character from an icon-font in your Scout project or a bitmap icon which is defined in your Scout project. See the migration guide for more details and the global property `showTreeIcons` which can turn on/off icons for all Tree instances. You should take care that all icons you use in a single tree have the same size. Here's an example for an outline with icons:
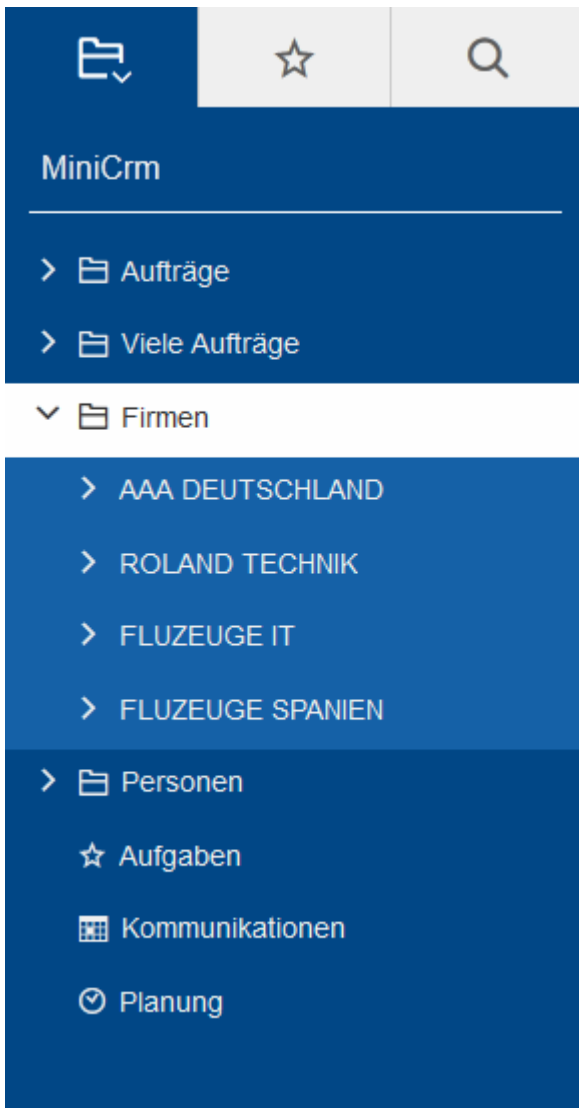
Figure 4. Outline with icons

File Chooser Field: Improved Usability

The file chooser field does not open an intermediate window anymore when clicked. Instead, the native file chooser is opened directly. This saves several clicks when a file needs to be uploaded. Furthermore, the whole field is now clickable. Until now the icon had to be clicked to choose a file which did not work well on touch devices.

Configurable Desktop Bench Layout

The Desktop Layout can be configured using the `IDesktop.setBenchLayoutData` method. This property is observed and might be changed during the applications lifecycle. For more information take a look at teh Technical Guide.

JMS connection failover

The scout MOM / JMS managed code supports for connection failover.

Connection failover is achieved using a connection wrapper and a session wrapper around the real

jms connection and session. Connection loss is discovered with (jms)connection.setExceptionListener. J2EE jms providers are excluded since those do failover themselves.

The main goal of connection failover is to maintain subscription listeners. All session wrapper methods do a one-time retry in case of failure. Default connection failover tries to reconnect 15 times every 2 seconds.

Subscription listeners are not stopped when the connection is dropped. However they try to receive messages again and again until the connection is restored or the session is closed by custom code.

Another improvement with similar scope is the subscription process itself. When calling MOM.subscribe the call is blocked at maximum for 30 seconds in order to wait for the subscription event loop to effectively start waiting and receiving messages. That way it cannot happen anymore that in the snippet MOM.subscribe(…) → schedules a event loop job MOM.publish(message) the message is published BEFORE being received from the subscriber due to latency in starting the background job.

Configuration: There are 3 new config.properties defined in IMom and IMomImplementor - scout.mom.failover.connectionRetryCount default 15 - scout.mom.failover.connectionRetryIntervalMillis default 2000 - scout.mom.failover.sessionRetryIntervalMillis default 5000

Migration: The interfaces and api are still stable, however customized jms code must check if some of the following old types/methods are being used or accessed: - method with IJmsSessionProvider.getSession() now throws JMSException - check override of JmsMomImplementor.createConnection() and postCreateConnection() so they do not call connection.setExceptionListener(…) - check override of IJmsSessionProvider, JmsSessionProvider since these are wrapped in the new JmsSessionProviderWrapper - do not use JmsMomImplementor.m_connection directly since during reconnect the member gets null and changes

Test: JmsMomImplementorTest.testSubscribeFailover

Do you want to improve this document? Have a look at the sources on GitHub.