Eclipse Scout

Release Notes

Scout Team

Version 7.0

Table of Contents

About This Release . . . 1
- Service Releases . . . 1
- Obtaining the Latest Version . . . 1

Upgrade to jQuery 3 . . . 3

Scout JS . . . 4
- Validators, Parser and Formatter (JS) . . . 4
- Simplified API (JS) . . . 4
- Logical Grid Validation (JS) . . . 4

New HTTP Abstraction Layer: Google HTTP Client for Java . . . 5

Support for REST Services . . . 7

Prevent Double Clicks on Buttons and Menus . . . 8

SplitBox . . . 9

New Smart Field: ISmartField2 (since 7.0.100) . . . 10

About This Release

Eclipse Scout 7.0 is part of the Eclipse *Oxygen* release. It was released in June 2017 (release schedule). The latest version of this release is *7.0.0.008_Oxygen*.

If you are upgrading from version 6.0, please also read the release notes for the 6.1 (*Oxygen preview*) release:
https://eclipsescout.github.io/6.1/release-notes.html

You can see the detailed change log on GitHub.

Service Releases

The following changes were made after the initial 7.0 release (Eclipse Oxygen release). The following notes relate to a *service release*.

Oxygen.1 (7.0.100) Release expected on September 27, 2017

Attention: The here described functionality has not yet been released and is part of an upcoming release.

- New Smart Field: ISmartField2 (since 7.0.100)

Obtaining the Latest Version

Runtime (Scout RT)

Scout RT artifacts are distributed via Maven:

- 7.0.0.008_Oxygen on *Maven Central*
- 7.0.0.008_Oxygen on *mvnrepository.com*

Usage example in the parent POM of your Scout application:

```
<dependency>
    <groupId>org.eclipse.scout.rt</groupId>
    <artifactId>org.eclipse.scout.rt</artifactId>
    <version>7.0.0.008_Oxygen</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

To install the Scout SDK into your existing Eclipse IDE, use this update site:
http://download.eclipse.org/scout/releases/7.0/7.0.0/008_Oxygen/

Demo Applications

The demo applications for this version can be found on the features/version/7.0.0.008_Oxygen branch of our docs repository on GitHub.

If you just want to play around with them without looking at the source code, you can always use the deployed versions:

- https://scout.bsi-software.com/contacts/
- https://scout.bsi-software.com/widgets/
- https://scout.bsi-software.com/jswidgets/

Upgrade to jQuery 3

We upgraded from jQuery 2.1.4 to jQuery 3.2.1. The new version contains a lot of improvements, including performance optimizations for animations and event handling. It also allowed us to remove some code which is now included in jQuery directly (e.g. addClassSVG, removeClassSVG, hasClassSVG).

Beside upgrading jQuery, we upgraded the Jasmine Maven Plugin to version 2.2 and the PhantomJS Maven Plugin to version 2.1.1, too. The YUI compressor has also been upgraded from 2.4.8 to version 2.4.9, but unfortunately it is not compatible with jQuery 3. To make it work again we had to fork it and now use the version 2.4.9-BSI-1.

Along with the jQuery upgrade we decided to remove jQuery UI and jQuery Mobile completely. Scout only included parts of these libraries and the usage was limited to 2 or 3 functions. Without these dependencies it will be easier in the future to upgrade jQuery itself.

Scout JS

With release 6.1 first steps had been taken to allow creating Scout applications with JavaScript only (see 6.1 Release Notes). The development has been continued so that the current release comes with a cleaner API, more working widgets and even some documentation (Tech Doc) and sample code (https://scout.bsi-software.com/jswidgets/). Still, it has not been finished yet and will be improved with the upcoming service releases. Let us know what you think!

It is important to note that *Scout JS* is not a separate framework. It actually just names the JavaScript part of the Scout framework. Classic Scout applications use it as well. This means effort made in this area also improves the stability of the Scout framework itself and even makes the development of custom widgets easier.

Validators, Parser and Formatter (JS)

The `ValueField.js` now provides an API to add custom parse, validate and format logic. This is especially useful for the `StringField`, but you may also use it for the `NumberField` and the `DateField`. See the technical guide for details.

Simplified API (JS)

The parameter `$parent` is now optional when calling widget.render(). The `$parent` may be resolved using `this.parent`. No need to always write `widget.render(this.$container)` anymore, instead just write `widget.render()` if the `$container` of the `parent` should be used as `$parent`.

The property change event has been simplified. Instead of `newProperties`, `oldProperties` and `changedProperties`, the event now contains `propertyName`, `oldValue` and `newValue`. This makes handling the event easier.

`DateField`, `StringField` and `NumberField` now use the value based API provided by `ValueField`. This means you can write the value using `field.setValue(value)`, and read it using `field.value`. The validators and formatter will be called accordingly.

Logical Grid Validation (JS)

When writing a Scout Form with Java, you don't have to care about the logical grid. You only have to specify some grid hints like width and height of a cell. The positioning of the cell is calculated automatically by the logical grid.

This is now also possible with JS based Scout applications. There is no need to manually create a Logical Grid (e.g. `VerticalSmartGroupBoxBodyGrid` or `HorizontalGroupBoxBodyGrid` and validate it anymore, this will be done automatically by the `LogicalGridLayout` itself.

New HTTP Abstraction Layer: Google HTTP Client for Java

The `org.eclipse.scout.rt.shared.servicetunnel.http.HttpServiceTunnel` class and other HTTP usages were changed to use the Google HTTP Client Library for Java 1.22. This library adds a HTTP abstraction layer and allows to use different low-level libraries like `java.net.HttpURLConnection` (one and only layer used in previous versions) or Apache HTTP Client 4.5.3 (new default).

Different HTTP clients with different parameters (even with different low-level libraries) may be used and kept using (custom) implementations of `org.eclipse.scout.rt.shared.http.IHttpTransportManager`. Currently there are two internal implementations of this interface: `HttpServiceTunnelTransportManager` (only used by the service tunnel) and `DefaultHttpTransportManager` (used for all other HTTP connections).

The following new configuration properties (none of them is required to be set, defaults are provided for all of them) were added:

- `scout.http.transport_factory`, possible values are `org.eclipse.scout.rt.shared.http.ApacheHttpTransportFactory` (default, see above), `org.eclipse.scout.rt.shared.http.NetHttpTransportFactory` (to use previous HttpURLConnection layer) or any custom implementation of an `org.eclipse.scout.rt.shared.http.IHttpTransportFactory`.

For the `HttpServiceTunnelTransportManager`:

- `org.eclipse.scout.rt.servicetunnel.apache_connection_time_to_live`, time to live (milliseconds) for kept alive connections (default: 1 hour, only applicable for Apache HTTP Client).
- `org.eclipse.scout.rt.servicetunnel.apache_max_connections_per_route`, maximum number of connections per route (default: 2048, only applicable for for Apache HTTP Client).
- `org.eclipse.scout.rt.servicetunnel.apache_max_connections_total`, maximum number of connections in total (default: 2048, only applicable for for Apache HTTP Client).

For the `DefaultHttpTransportManager`:

- `scout.http.apache_connection_time_to_live`, time to live (milliseconds) for kept alive connections (default: 1 hour, only applicable for Apache HTTP Client).
- `scout.http.apache_max_connections_per_route`, maximum number of connections per route (default: 32, only applicable for Apache HTTP Client).
- `scout.http.apache_max_connections_total`, maximum number of connections in total (default: 128, only applicable for Apache HTTP Client).

For each Apache HTTP Client created using the `org.eclipse.scout.rt.shared.http.ApacheHttpTransportFactory` (by default each `org.eclipse.scout.rt.shared.http.IHttpTransportManager` using the Apache HTTP Client) their own `org.eclipse.scout.rt.shared.http.ApacheMultiSessionCookieStore` and `org.eclipse.scout.rt.shared.http.proxy.ConfigurableProxySelector` (see javadoc for detailed

description and configurability) are created. These instances are therefore not registered globally for the java virtual machine anymore.

Support for REST Services

The following new Scout modules have been added to support REST services with Jackson as marshaller:

- `org.eclipse.scout.rt.rest`
- `org.eclipse.scout.rt.rest.test`
- `org.eclipse.scout.rt.jackson`
- `org.eclipse.scout.rt.jackson.test`

The most important class is the `org.eclipse.scout.rt.rest.RestApplication` which searches for all implementations of `IRestResource` and exposes them as REST services. It also registers `ExceptionMappers` and setups Jackson to work with Jandex.

So if you want to use REST services, you could use the Jersey REST servlet (`org.glassfish.jersey.servlet.ServletContainer`), pass the `RestApplication` as parameter and install the `org.eclipse.scout.rt.server.context.ServerRunContextFilter` to have the proper run context for every REST call. Creating the REST resource is straight forward using the annotations from `javax.ws.rs`. Just make sure the resource implements the interface `IRestResource` so that it will be registered by the `RestApplication` on startup.

Prevent Double Clicks on Buttons and Menus

If a button or a menu is clicked twice within a short period of time, the corresponding action is executed twice. This can be convenient (e.g. when inserting new rows in a table) or unproblematic (e.g. when closing a form - the second click will just be ignored). However, there are cases where executing an action twice would break things. To instruct the UI to block double clicks, a new property "preventDoubleClick" is provided on buttons and menus:

- `AbstractButton.getConfiguredPreventDoubleClick()`
- `AbstractMenu.getConfiguredPreventDoubleClick()`

The default value is `false`.

SplitBox

The SplitBox widget now supports a minimum splitter position according to the collapsible field. The collapsible field size is limited between minimum splitter position and maximum available size. The collapse buttons now toggles between three modes of the collapsible field: `default`, `minimized` and `collapsed`. The default value for minimal splitter size is `null`, which means, no minimal splitter size is set and no change in existing behavior.

New API methods on `AbstractSplitBox`:

- `Double getMinSplitterPosition()`
- `void setMinSplitterPosition(Double minPosition)`
- `boolean isFieldMinimized()`
- `void setFieldMinimized(boolean minimized)`

Additional to the existing three splitbox position types a new `SPLITTER_POSITION_TYPE_RELATIVE_SECOND` type was added. This new splitter position type allows to specify the size of the second field relative to the full size of the splitbox.

New Smart Field: ISmartField2 (since 7.0.100)

This release introduces a new smart field: `ISmartField2`. It has almost the same interface as the old smart field `ISmartField`, which still exists in this Scout release, but will be removed with 7.1. The main differences to the old smart field:

- In "Scout classic" (with a Java UI server) there is no longer a model representation of the proposal chooser. In the new smart field the whole state of the proposal chooser is kept on client side in the browser. The Java UI server only sends lookup rows to the client. Depending on the smart field configuration `SmartField2.js` will render either a proposal chooser with a table or a tree (hierarchical). It's still possible to replace the default proposal chooser, but now you have to write a bit of JavaScript code to do that.
- The smart field can now be used with Scout JS. This means you're no longer restricted to "Scout classic" when you want to use a smart field and you can use the smart field with any static or dynamic data source, for instance a REST service. Take a look at the jswidgets demo app to see examples how to use the smart field with JavaScript.

Migrating from `ISmartField` to `ISmartField2` should be simple in most cases, since the interfaces of the old and the new smart field are almost identical. Differences are:

- There is no longer a `IMixedSmartField` with two generic types for VALUE and LOOKUP_TYPE, since these two types are identical in 99.9% of all cases. When you migrate an old Scout application that uses different types you could either provide a new LookupCall that has the same lookup type as the smart field value, or you could simply cast the value of the smart field where needed.
- The value of the proposal field is now always a String. The generic type you pass to the proposal field is the lookup type. Use the methods `setValueAsString` and `getValueAsString` to read and write the value of the proposal field. Additionally you can still access the selected lookup row of the proposal field and get the key of the lookup row.
- When you migrate an old Scout application that has a custom proposal chooser, you should probably create a custom JavaScript implementation for your smart field. There you can override the behavior of the default implementation.

Note: With 7.1 the old smart field will be deleted and replaced by the new smart field `ISmartField2`, additionally in 7.1 `ISmartField2` will be renamed to `ISmartField` again. When you start a new Scout project with this release you should use `ISmartField2`.

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