Eclipse Scout

Release Notes

Scout Team

Version 7.0

Table of Contents

Overview

Eclipse Scout 7.0 will be the Eclipse Oxygen release.

The official release for Eclipse Oxygen will be released in June 2017. See also https://wiki.eclipse.org/Oxygen/Simultaneous_Release_Plan for the release schedule.

Lastest stable version

Runtime (Scout RT)

Eclipse Runtime on maven central: Version `7.0.0.007_RC3`

Usage example in the parent POM of your scout application:

```
<dependency>
    <groupId>org.eclipse.scout.rt</groupId>
    <artifactId>org.eclipse.scout.rt</artifactId>
    <version>7.0.0.007_RC3</version>
    <type>pom</type>
    <scope>import</scope>
</dependency>
```

See this artifact on mvnrepository.com: org.eclipse.scout.rt

Eclipse Tooling (Scout SDK)

Eclipse IDE distribution (also called EPP): Eclipse for Scout Developers (RC3)

Update site to install the Scout SDK in your Eclipse IDE (already installed in the Scout EPP):

```
http://download.eclipse.org/scout/releases/7.0/7.0.0/007_RC3/
```

Demo Applications

The demo applications using this version can be found on the features/version/7.0.0.007_RC3 branch of our docs repository on GitHub.

What's New

This document shows some of the new features delivered with the release 7.0. The release contains a lot of bugfixes and even some features not mentioned here. If you are interested in the detailed change log see https://github.com/eclipse/scout.rt/compare/releases/6.1.x...releases/7.0.x.

If you upgrade from version 6.0, also see the release notes for the 6.1 release.
https://eclipsescout.github.io/6.1/release-notes.html

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.

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.

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`.

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.

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