Eclipse Scout: Migration Guide

Table of Contents

About This Document

This document describes all relevant changes **from Eclipse Scout 10.0 to Eclipse Scout 11.0**. If existing code has to be migrated, instructions are provided here.

Service Releases

Scout 11.0 will continue to be maintained for a while and new builds may be released from time to time. Beside bugfixes, these service releases may even contain some minor features.

Updated 3rd party dependencies

All 3rd party build- and runtime-dependencies have been updated to the latest versions. This requires the following modifications:

If you are using the `jaxws-maven-plugin`, please change the groupId in your pom.xml files from `com.helger.maven` to `com.sun.xml.ws`.

In all your `web.xml` files update the `web-app` xml document element to:

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                      http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0"
  metadata-complete="true">
...
</web-app>
```

If you have Jax-Ws web service providers (this is the case if you use the `org.eclipse.scout.rt.server.jaxws.provider.annotation.WebServiceEntryPoint` annotation), the classpath of the annotation processor must be updated as described for each IDE:

For Eclipse Developers: Update the content of all `.factorypath` files as follows:

```
<factorypath>
<factorypathentry kind="VARJAR"
id="M2_REPO/org/eclipse/scout/rt/org.eclipse.scout.jaxws.apt/11.0-
SNAPSHOT/org.eclipse.scout.jaxws.apt-11.0-SNAPSHOT.jar" enabled="true"
runInBatchMode="false"/>
<factorypathentry kind="VARJAR"
id="M2_REPO/org/glassfish/jaxb/codemodel/2.3.3/codemodel-2.3.3.jar" enabled="true"
runInBatchMode="false"/>
<factorypathentry kind="VARJAR"
id="M2_REPO/org/eclipse/scout/rt/org.eclipse.scout.rt.platform/11.0-
SNAPSHOT/org.eclipse.scout.rt.platform-11.0-SNAPSHOT.jar" enabled="true"
runInBatchMode="false"/>
<factorypathentry kind="VARJAR"
id="M2_REPO/org/eclipse/scout/rt/org.eclipse.scout.rt.server.jaxws/11.0-
SNAPSHOT/org.eclipse.scout.rt.server.jaxws-11.0-SNAPSHOT.jar" enabled="true"
runInBatchMode="false"/>
<factorypathentry kind="VARJAR" id="M2_REPO/jakarta/servlet/jakarta.servlet-
api/4.0.4/jakarta.servlet-api-4.0.4.jar" enabled="true" runInBatchMode="false"/>
<factorypathentry kind="VARJAR" id="M2_REPO/org/slf4j/slf4j-api/1.7.30/slf4j-api-
1.7.30.jar" enabled="true" runInBatchMode="false"/>
<factorypathentry kind="VARJAR" id="M2_REPO/jakarta/jws/jakarta.jws-
api/2.1.0/jakarta.jws-api-2.1.0.jar" enabled="true" runInBatchMode="false"/>
<factorypathentry kind="VARJAR" id="M2_REPO/jakarta/annotation/jakarta.annotation-
api/1.3.5/jakarta.annotation-api-1.3.5.jar" enabled="true" runInBatchMode="false"/>
<factorypathentry kind="VARJAR" id="M2_REPO/jakarta/xml/ws/jakarta.xml.ws-
api/2.3.3/jakarta.xml.ws-api-2.3.3.jar" enabled="true" runInBatchMode="false"/>
</factorypath>
```

For IntelliJ IDEA Developers: The annotationProcessing path in the file `.idea/compiler.xml` is automatically updated when performing a maven reload.

New ESLint Settings

The eslint rules used by Scout have been adjusted. So if you have a dependency to `@eclipse-scout/eslint-config` and you update to the latest version, your JavaScript code may report some new warnings.

The following rules have been turned on: `no-var`, `prefer-arrow-callback`,`prefer-rest-params`, `prefer-spread`.

If you don't like the warnings, you can adjust your `eslintrc.js` and disable the rules. But we suggest migrating your code because it really benefits from using these new ES6 features.

Luckily, eslint provides an automatic fix for the first two. Just run the following command in the root of your project:

```
node_modules/.bin/eslint . --fix
```

You may also need to ignore some files or folders first, just add them to the `.eslintignore` file.

`prefer-rest-params` and `prefer-spread` need a manual migration, unless you choose to disable them.

Removal of module `@eclipse-scout/testing`

The code of Node module `@eclipse-scout/testing` has been moved into `@eclipse-scout/core/src/testing` because in the past these two modules had cyclic dependencies. Now the testing helper code can be found in the core module (`@eclipse-scout/core`) within the `src/testing` folder. The `@eclipse-scout/testing` module has been deleted.

Migration:

- Remove `@eclipse-scout/testing` from your package.json files
- Change the JavaScript imports from `… from '@eclipse-scout/testing';` to `… from '@eclipse-scout/core/src/testing/index';`.

Automatic JavaScript Chunks

Before Scout 11 JavaScript chunks have been declared explicitly. While you have full control on the output using this explicit definition, it is often not optimal concerning performance and might require modifications when a new module or library is added to the project. Therefore starting with version 11 Scout additionally supports automatic chunk creation for your JavaScript assets.

When using this new feature (which is the default configuration now), chunks are automatically created based on size and code usage (JavaScript imports). Therefore, it is no longer necessary to list every chunk manually in the html file. Instead, Scout automatically inserts the necessary script tags according to the chunks created.

Migration if you want to use automatic chunk creation (recommended)

Replace all `scout:script` tags in your html files with a single `scout:scripts` tag (please note the extra "s" at the end). So instead of:

```
<scout:script src="vendors.js"/>
<scout:script src="jquery.js"/>
<scout:script src="eclipse-scout.js"/>
<scout:script src="yourApp.js"/>
```

replace it with:

```
<scout:scripts entryPoint="yourApp"/>
```

The name to write in the `entryPoint` attribute is the same as used in the `entry` section of your `webpack.config.js` file. Typically you also have entries named `login` and `logout`. These are the names to put in the `entryPoint` attribute within the corresponding html file that should launch this entry point.

Accordingly, the example html file above will include all chunks required by the webpack `entry` named `yourApp`.

If you specified custom chunk cache groups in your webpack.config.js (e.g. by defining `config.optimization.splitChunks.cacheGroups.yourChunkName = {…}`), remove these chunk definitions.

In your `web.xml` files remove the old chunk names (`/jquery*.js`, `/login*.js`, `/logout*.js`, `/eclipse-scout*.js`) from the `filter-exclude` values of the servlet filter. Instead, add new exclusions of the form `/*entryPointName*.js` for each entry point that may be accessed without authentication. Typically, these are `/*login*.js` and `/*logout*.js`.

Migration if you don’t want to use automatic chunks (not recommended)

As Scout no longer includes default chunks, you have to define all chunks in your project yourselves.

- To include the old default Scout chunks copy the chunk definitions from the old Scout defaults.
- If you already have custom chunk definitions, you can keep them as they are.

Web Resource Resolver

The `IWebResourceResolver` interface has been extended to return multiple resources instead of one.

For this the methods now return `List<WebResourceDescriptor>` instead of `Optional<WebResourceDescriptor>`.

If the `FilesystemWebResourceResolver` or the `ClasspathWebResourceResolver` has been replaced in your project, perform the following migration:

- Adjust the return type of `getResourceImpl` to `Stream<URL>`.
- Adapt the returned value depending on your implementation:
  - If the resolver observes different locations: return all resources found from all the locations.
  - If the resolver only uses one location: return a stream containing one element or an empty stream.

You may use the inherited methods `resolveUrls` and `toUrl` if your resolver replaces the `FilesystemWebResourceResolver`. These methods may be handy to search for a relative path in one or multiple root directories.

TabBox: New Behavior of LabelVisible

As already mentioned in the release notes, it is now possible to hide the tab box header with the property labelVisible. Since this had no effect in previous Scout versions you should check whether you accidentally set the property to false. The default of the property is true, so if the property was not set at all or set to true it will be fine.

SmartColumns: New Behavior of PrepareLookupCall Events

Upto version 10.0 no prepareLookupCall event has been triggered when doing key lookup calls for cells inside `SmartColumns`. These key lookup calls are for example relevant for initally loading the values of the cells inside the column. Starting from version 11.0 a key lookup call always triggers a prepareLookupCall events.

New property event.row

The property row was given to the prepareLookupCall event in `SmartColumns`. There are a few things to consider:

- The property `selectedRow` from `Table` must not be used anymore when a reference to the selected row is needed. Instead, `event.row` should be used to find the selected row.
- `event.row` can be `null` or `undefined`. The latter is for example the case when initially loading the values from the cells inside the column.
- Batch lookup calls have no property `event.row` when triggering the prepareLookupCall event. Hence, if your prepareLookupCall event depends on the selected row during key lookups, you should set the property `batch: false` inside the corresponding lookupCall of your `ConfigFormModel.js` file.

MessageBox: New Behavior of doClose()

The priorities of the answer of `MessageBox.doClose()` will now be `CANCEL_OPTION`, `NO_OPTION`, `YES_OPTION`. The old behavior is still available in the form of the method `MessageBox.doOk()`.

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