Eclipse Scout Migration Guide

Version 22.0

Table of Contents

Looking for something else? Visit https://eclipsescout.github.io for all Scout related documentation.

About This Document

This document describes all relevant changes **from Eclipse Scout 11.0 to Eclipse Scout 22.0**. If existing code has to be migrated, instructions are provided here.

Obtaining the Latest Version

Scout Runtime for Java

Scout Runtime artifacts for Java are distributed using Maven Central:

- 22.0-SNAPSHOT on *Maven Central*
- 22.0-SNAPSHOT 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>22.0-SNAPSHOT</version>
    <type>pom</type>
    <scope>import</scope>
</dependency>
```

Scout Runtime for JavaScript

Scout Runtime artifacts for JavaScript are distributed using npm:

- Scout Core Runtime
- All official Scout JavaScript packages

Usage example in your package.json:

```
{
  "name": "my-module",
  "version": "1.0.0",
  "dependencies": {
    "@eclipse-scout/core": "22.0.0-snapshot",
    "jquery": "3.5.1"
  }
}
```

The pre-built Scout JavaScript assets are also available using a CDN (e.g. to be directly included in a html document): https://www.jsdelivr.com/package/npm/@eclipse-scout/core?path=dist

IDE Tooling (Scout SDK)

Starting with Scout 11 the IDE Tooling requires at least Java 11 to run. With the help of the SDK plugins you can still develop applications running with Java 8, but the IDE itself requires Java 11 or newer.

Scout officially supports IntelliJ IDEA and Eclipse for Scout Developers.

IntelliJ IDEA

You can download the Scout plugin for IntelliJ IDEA from the JetBrains Plugin Repository or you can use the plugins client built into IntelliJ IDEA. Please refer to the IntelliJ Help on how to install and manage plugins.

Eclipse

You can download the complete Eclipse IDE with Scout SDK included here:
Eclipse for Scout Developers

To install the Scout SDK into your existing Eclipse IDE, use this P2 update site:
http://download.eclipse.org/scout/releases/11.0/

Update the Scout version of an existing application

1. Scout uses Node 12.18.4 and pnpm 5.7.0. Update your development tools accordingly.
2. Update the version of the `maven_rt_plugin_config-master` in your `pom.xml` files to `3.14.1`.
3. Update the Scout version (usually in the `org.eclipse.scout.rt_version` property of the parent pom) to `22.0-SNAPSHOT`.
4. Update all Node modules having the `@eclipse-scout` namespace in your package.json files to `22.0.0-snapshot`. There is one exception: The module `@eclipse-scout/releng` stays at the version `^10.0.0`.
5. Optional switch from npm to pnpm (recommended)
   a. Overwrite your `js build.sh` (or `js build.cmd` on Windows) with the following content and replace the `${rootArtifactId}` with your module name:
      i. js build.sh: sample content
      ii. js build.cmd: sample content
   b. Remove all file dependencies in your `package.json` files. So dependency versions like `file:../helloworld.ui/` are replaced with the version of the module previously referenced (e.g. `1.0.0-SNAPSHOT`).
   c. In the root directory of your project (next to the root `pom.xml`) create a new file `pnpm-workspace.yaml` with the following example content:

      ```
      packages:
      - 'helloworld.ui'
      - 'helloworld.app'
      ```

      Instead of the sample modules above: list all your modules having a `package.json` file.
6. Update 3rd party JavaScript dependencies in all your `package.json` files:
   a. `eslint` to `7.9.0`
   b. `babel-eslint` to `10.1.0`
   c. `eslint-plugin-babel` to `5.3.1`
   d. `jquery` to `3.5.1`

3rd party dependency upgrade migration

All 3rd party build- and runtime-dependencies have been updated to newer 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 root 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/22.0-
SNAPSHOT/org.eclipse.scout.jaxws.apt-22.0-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/22.0-
SNAPSHOT/org.eclipse.scout.rt.platform-22.0-SNAPSHOT.jar" enabled="true"
runInBatchMode="false"/>
<factorypathentry kind="VARJAR" id=
"M2_REPO/org/eclipse/scout/rt/org.eclipse.scout.rt.server.jaxws/22.0-
SNAPSHOT/org.eclipse.scout.rt.server.jaxws-22.0-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 Script Chunks

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

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

Example for scripts:

```
<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"/>
```

Example for stylesheet:

```
<scout:stylesheet src="yourApp-theme.css"/>
```

replace it with:

```
<scout:stylesheets entrypoint="yourApp-theme"/>
```

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.

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`, `/yourApp-theme*.css`) from the `filter-exclude` values of the servlet filter. Instead, add new exclusions of the form `/*entryPointName*.js` and `/*entryPointName*.css` for each entry point that may be accessed without authentication. Typically, these are `/*login*.js`, `/*logout*.js` and the themes css files.

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()`.

Maven master release profiles

Some maven profiles have been removed from the maven master (e.g. `org.eclipse.scout:maven_rt_plugin_config-master`) (since 3.14.1). These profiles have been used for release management. No Scout build jobs do still use git commands through maven. Instead, any git command is issued directly (usually from a jenkins pipeline). Therefore, we cannot support these profiles anymore. If any of these profiles or configurations are still in use in your project, you may copy them from an older maven master version into your own parent maven pom.

Removed profiles

- `release.setversion`
- `release.checkin`
- `release.tag`

Removed related properties

- `master_release_pushChanges`
- `master_release_tagName`
- `master_release_checkinMessage`

Also, the default plugin configuration of `org.apache.maven.plugins:maven-release-plugin` was removed.

The following script is a template which can be used to change the project versions for a new release build without the removed profiles and direct git commands.

Listing 1. Shell script to set project version to a new version for release builds

```
# set version of maven project modules
mvn -f ${mavenParentPom} org.codehaus.mojo:versions-maven-plugin:set
-DnewVersion=${newVersion} -DgenerateBackupPoms=false
# set version of npm project packages
mvn -f ${mavenParentPom} process-sources -N -P npm-install-node,npm-install-
workspace,npm-deploy -Dmaster_node_dir=\${NODE_BIN} -Dmaster_npm_release_build=true
-Dmaster_release_newVersion=${newVersion}
# add changed files to git changes
git add "**/pom.xml" "**/package.json"
```

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