Eclipse

IDE

Configuration quickstart for developing CzechIdM in Eclipse.

Install Eclipse

Download the last stable version of Eclipse from Eclipse download page. When running the installer, choose "IDE for Java EE Developers". It is advised to download Eclipse directly from download page, since when installing from distribution repositories via dnf or yum, you are not able to choose from distinct Eclipse editions.

If you installed Java separately from your system libraries, you should run Eclipse with the specified JDK:

```
cd /path/to/eclipse
export JAVA_HOME=/path/to/installed/jdk/
./eclipse
```

Next set the correct Java Runtime in Eclipse: Window → Preferences → Java → Installed JREs → button Add… → Standard VM - Next. In the following window, fill the "JRE home" with the path to JDK, then click Finish

Remove Eclipse pom.xml error - `Plugin execution not covered by lifecycle configuration: org.bsc.maven:maven-processor-plugin:3.3.1:process (execution: process, phase: generate-sources)` - go to `Window` → `Preferences` → `Maven` → `Errors/Warnings` → set `Plugins execution not covered by lifecycle configuration` to `warning`.

Import project

Open Eclipse and choose to import project from existing Maven projects: File → Import → Maven – Existing Maven Projects → Next → Select root directory

Choose the location of your repository from file explorer, i.e. <path>/CzechIdM/Realization/backend.

In the "Projects" window, tick checkboxes for projects under core module and other modules. Do not import the "gui" module. Thus you should have following modules selected:

- ic
- acc
- app
- core-api
- core-test-api
- core-impl

- aggregator
- example
- parent
- vs
- rpt
- rpt-api
- rpt-impl

Finish. Project should be imported.

Set the "dev" profile for developing (because it uses postgresql DB and module acc): Right click "idm-app" → Maven → Select Maven Profiles → check "dev"

Metamodel generation

After creating the project, make sure following modules are available in explorer (with prefix "idm-"):

- acc
- app
- core-api
- core-impl
- core-test-api
- example
- ic
- parent
- rpt
- rpt-api
- rpt-impl

This setup has to be done for modules **core-api**, **core-impl**, and other optional modules, which uses criteria api (i. e., **acc**, **example**, **rpt-impl**).

Note: If you don't set metamodel generation, you will see Java problems like `ExampleProduct_ cannot be resolved to a variable`.

- Go to Project → Properties → Java Compiler → Annotation Processing → check "Enable project specific settings" and fill "Generated source directory" = "target/metamodel".
- Go to Project → Properties → Java Compiler → Annotation Processing → Factory path → check "Enable project specific settings" and add external jar **hibernate-jpamodelgen.jar** (version 5.x.x). Artefact could be found in local maven repository or downloaded from any public maven repository.

If this solution didn't help and you still see the `ExampleProduct_ cannot be resolved to a variable` error, you have to build the project in command line. Go to the relevant paths (e. g., /git/CzechIdMng/Realization/backend/rpt) and run `mvn clean install`. Do this for every directory that shows this error.

Build the Project

After that you can build the projects from Eclipse by Project → Build Project or you can check "Build Automatically". If you see some dependency errors try this approach: Right-click the project "idm-aggregator" and choose "Run as" → "Maven install". The first build of the project will download necessary libraries to your local Maven repository (e.g. "forest" from Nexus repository). If dependency errors still remain, for all projects: Right click on the project → Maven → Update Project... → click Select All → Click OK

Install Tomcat

Download Tomcat 9.0.* (tested version 9.0.80) from Apache website. Unzip it somewhere in your home directory.

Next we will install Tomcat as a Server to the Eclipse: Show the tab "Servers" if it isn't visible yet: Window → Show View → Servers

Create new server: In the tab "Servers", right-click and choose New → Server → Apache → Tomcat v8.0 Server - Next → For „Tomcat installation directory“ select the path to the installed Tomcat directory. As "JRE" choose installed JDK. Then Next → Next → Finish.

Add idm-app as an application to Tomcat: Right-click on the Tomcat server, choose "Add and Remove" and select "idm-app"

Optionally set some other options for the server: Double-click on the Tomcat server, then

- in the section "Timeouts" increase the timeout for start to 240 s.
- in the section "Server Options", uncheck "Modules auto reload by default".

Deploy changes without reloading

Small changes in code may be applied to the server without restarting the server or reloading modules (hot code replacement). This can be configured as follows:

1. Double-click on the Tomcat server, in the section "Publishing" choose the option "Automatically publish when resources change"
2. Switch to the "Modules" tab of the Tomcat server and choose the module "idm-app". Click the button "Edit..." and uncheck "Auto reloading enabled". After submitting you should see the value "Disabled" in the column "Auto Reload".
3. Save changes in the Tomcat server configuration
4. Start the Tomcat server in "DEBUG" mode.

Run unit & integration tests

When you want to run unit tests, you must switch to Maven profile **test**. Otherwise some tests (especially those which use database connection) may fail. Set the "test" profile: Right click "idm-app" (or the module which you want to test) → Maven → Select Maven Profiles → check "test"

If you want to run only tests in some class:

```
1) open this class,
2) right click on the class name,
3) choose Run As → JUnit Test.
```

Common development & tips

Tomcat server fails to start

If Tomcat server fails to start, try following:

- Check all common development tips
- Check build errors - check that the tab Markers or Problems doesn't show any Java or Maven build error. (You can ignore XML validation problems.) If it does, try updating Maven project.
- Try Clean/Publish actions on the Tomcat server.
- Try restarting Eclipse.

The exception: `org.springframework.context.ApplicationContextException: Unable to start embedded container; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'authenticationFilter': Injection of autowired dependencies failed …… very long stacktrace ……. Error creating bean with name 'flywayCore' defined in class path resource [eu/bcvsolutions/idm/core/config/flyway/CoreFlywayConfig.class]: Initialization of bean failed; nested exception is java.lang.IllegalArgumentException: Comparison method violates its general contract!` is probably caused by a bug in Flyway.

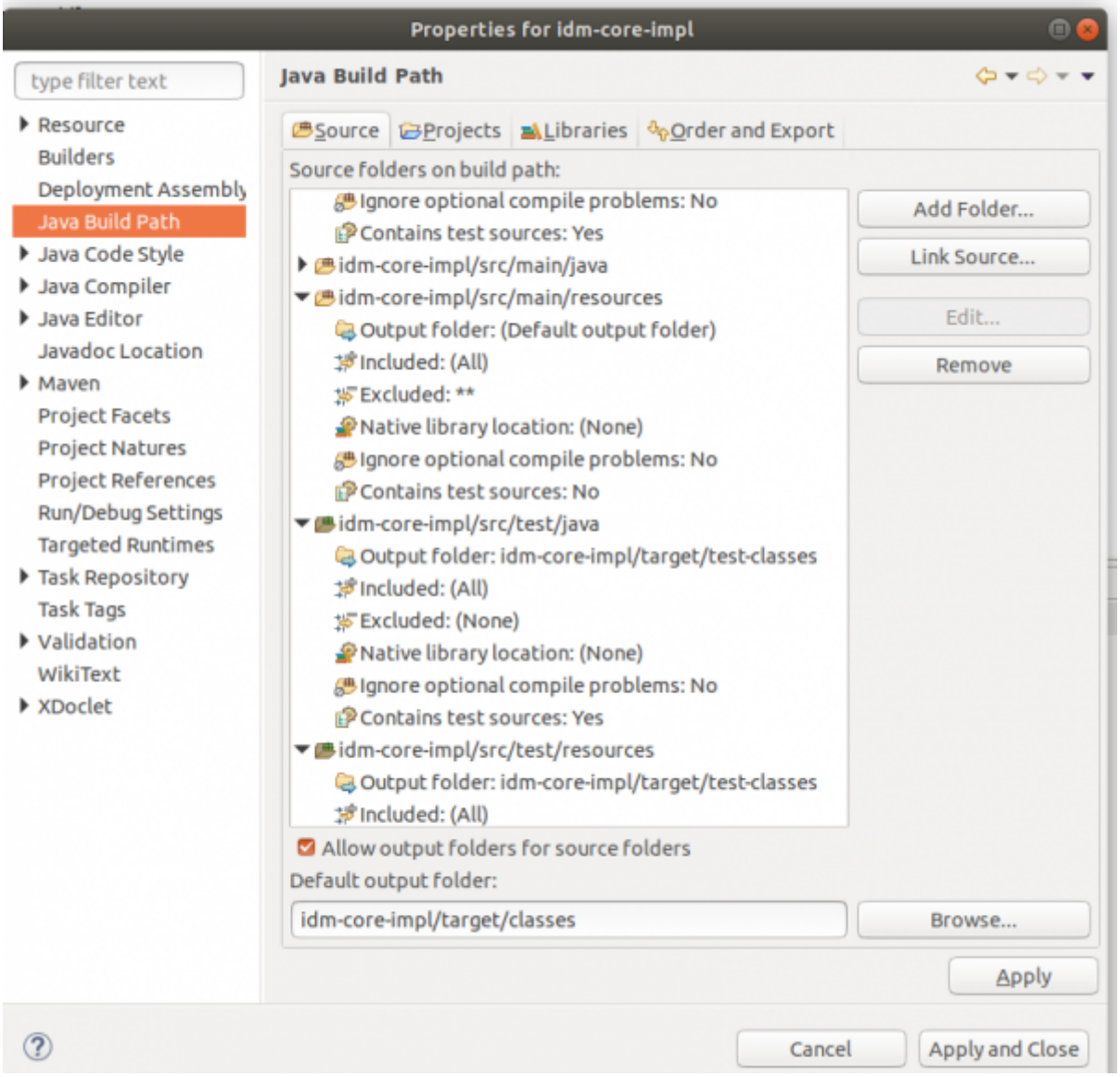
In such case, go to Windows → Preferences → Installed JREs→ edit JRE and add to default VM arguments `-Djava.util.Arrays.useLegacyMergeSort=true`

Another possible issue may be an exception like this one:
`java.util.concurrent.ExecutionException: org.apache.catalina.LifecycleException: Failed to start component [StandardEngine[Catalina].StandardHost[localhost].StandardContext[/idm-app]] Caused by: org.apache.catalina.LifecycleException: Failed to start component [StandardEngine[Catalina].StandardHost[localhost].StandardContext[/idm-app]] at org.apache.catalina.util.L Caused by:`

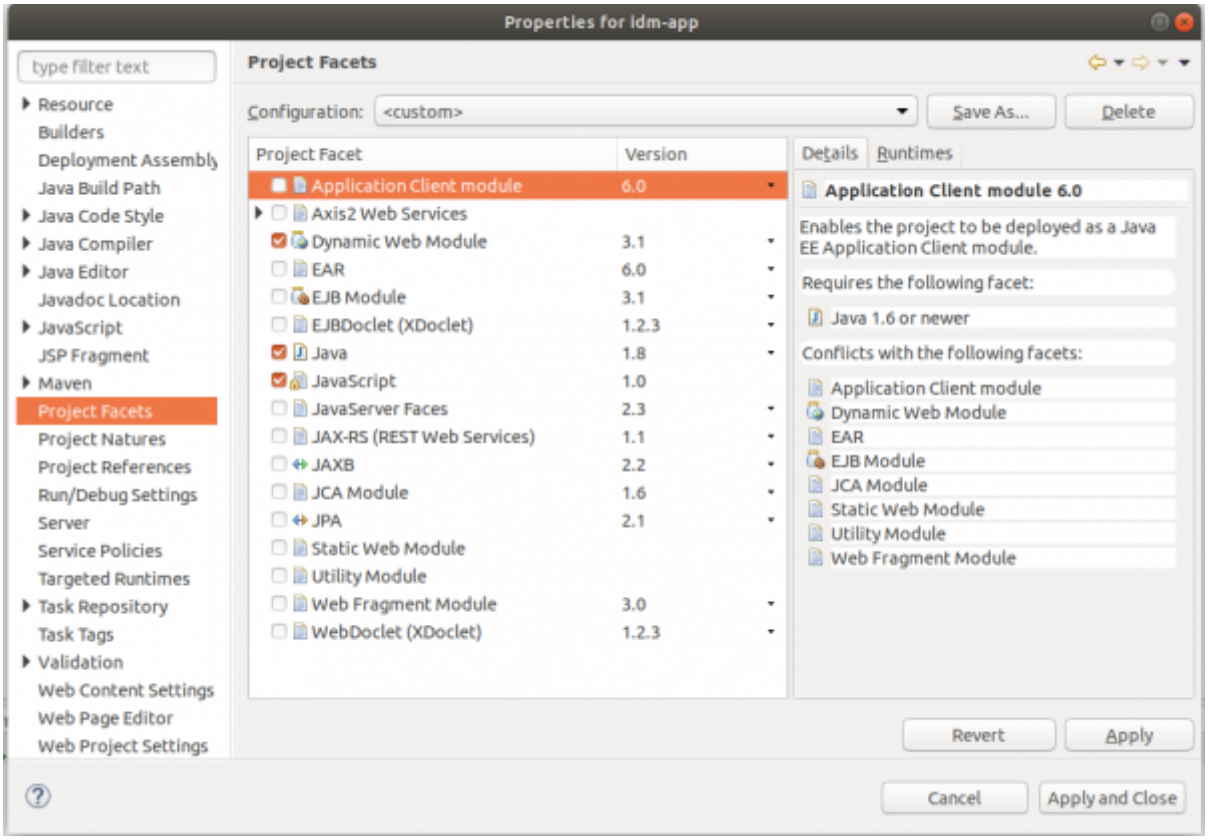
```
org.springframework.beans.factory.BeanDefinitionStoreException: Failed to
parse configuration class [eu.bcvsolutions.idm.IdmApplication]; nested
exception is java.io.FileNotFoundException: class path resource
[eu/bcvsolutions/idm/test/api/AbstractIntegrationTest.class] cannot be opened
because it does not exist at
org.springframework.context.annotation.ConfigurationClassParser.parse(Configu
rationClassParser.java:182)
```

If you see this, make sure that

- you have the project idm-core-test-api open
- in Properties of the idm-core-impl (or any problematic project), check Java Build Path and make sure that each test folder has the Output folder set to test-classes.



See below: If you have a problem in not being able to Add and Remove, try to open idm-app, look at Properties, Project Facets and check you have the correct version of **Java** (should be 11) abd **Dynamic Web Module** (3.1):



Maven build passes, but Eclipse shows errors

It can happen that Maven builds the project fine, but eclipse Project → Clean will throw an error, such as:

```
Description    Resource    Path    Location    Type
Failed to execute mojo org.apache.maven.plugins:maven-dependency-
plugin:3.1.1:copy {execution: copy} (org.apache.maven.plugins:maven-
dependency-plugin:3.1.1:copy:copy:compile)

org.eclipse.core.runtime.CoreException: Failed to execute mojo
org.apache.maven.plugins:maven-dependency-plugin:3.1.1:copy {execution:
copy}
    at
org.eclipse.m2e.core.internal.embedder.MavenExecutionContext.executeMojo(Mav
enExecutionContext.java:340)
    at
org.eclipse.m2e.core.internal.embedder.MavenExecutionContext.lambda$0(MavenE
xecutionContext.java:291)
    at
org.eclipse.m2e.core.internal.embedder.MavenExecutionContext.executeBare(Mav
enExecutionContext.java:394)
    at
org.eclipse.m2e.core.internal.embedder.MavenExecutionContext.execute(MavenEx
ecutionContext.java:275)
    at
org.eclipse.m2e.core.internal.embedder.MavenExecutionContext.execute(MavenEx
```

```
ecutionContext.java:290)
    at
org.eclipse.m2e.core.project.configurator.MojoExecutionBuildParticipant.buil
d(MojoExecutionBuildParticipant.java:57)
    at
org.eclipse.m2e.core.internal.builder.MavenBuilderImpl.lambda$2(MavenBuilder
Impl.java:153)
```

In that case, update the Maven project (right click project → Maven → Update Project.

Update project after pulling new version

When you pull new version of the project from Git and there were some changes including Maven dependencies (e.g. newer version of some artifact), you should update your project:

- select all projects
- right-click and choose Maven → Update Project The IDE will update dependencies and rebuild the projects if necessary.

Implementing a ConnId connector

We came across problematic behavior of Eclipse when implementing a ConnId connector. If you develop the connector that is a dependency of CzechIdM and the same connector (in the same version) is opened in Eclipse, then Tomcat with CzechIdM may fail to start with the following exception: `Exception encountered during context initialization - cancelling refresh attempt: org.springframework.context.ApplicationContextException: Unable to start embedded container; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'authenticationFilter': Injection of autowired dependencies failed …. nested exception is org.springframework.beans.BeanInstantiationException: Failed to instantiate [eu.bcvsolutions.idm.ic.service.impl.DefaultIcConfigurationFacade]: Constructor threw exception; nested exception is java.lang.NullPointerException`

If you encounter this exception and you want to use Debug server mode for implementing the connector (deploying without restarts), try the following steps:

- Open **pom.xml** of the connector and add inside the tags `<build><plugins>`:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-jar-plugin</artifactId>
  <configuration>
    <archive>
      <addMavenDescriptor>false</addMavenDescriptor>
      <manifestEntries>
        <ConnectorBundle-
```

```
FrameworkVersion>${connid.version}</ConnectorBundle-FrameworkVersion>
        <ConnectorBundle-Name>${project.artifactId}</ConnectorBundle-Name>
        <ConnectorBundle-Version>${project.version}</ConnectorBundle-
Version>
      </manifestEntries>
    </archive>
  </configuration>
</plugin>
```

- Right-click on the project with the connector and run Maven → Update Project
- Right-click on the Tomcat server and run Clean or Publish
- Go to the deployments folder of the Tomcat server (this depends on your installation, e.g.: `/home/user/bin/apache-tomcat-8.0.36/wtpwebapps/idm-app/WEB-INF/lib/`) and remove the folder `META-INF/maven/` from the JAR containing your connector:

```
cd /home/user/bin/apache-tomcat-8.0.36/wtpwebapps/idm-app/WEB-INF/lib/
zip -d csv-connector-1.0.0.jar "META-INF/maven/*"

deleting: META-INF/maven/
deleting: META-INF/maven/eu.bcvsolutions.idm.connectors.csv/
deleting: META-INF/maven/eu.bcvsolutions.idm.connectors.csv/csv-connector/
deleting: META-INF/maven/eu.bcvsolutions.idm.connectors.csv/csv-
connector/pom.properties
deleting: META-INF/maven/eu.bcvsolutions.idm.connectors.csv/csv-
connector/pom.xml
```

- Check that the file `META-INF/MANIFEST.MF` is first or second in the JAR file:

```
jar -tf csv-connector-1.0.0.jar

META-INF/
META-INF/MANIFEST.MF
...
```

- Now you can start Tomcat in the Debug mode. The server starts and your changes in the connector can be deployed without restarts (if you don't change methods etc.)

Explanation & further improvements

The problem is in fact caused by Eclipse Maven plugin, which skips the standard `assembly` Maven goal and doesn't put the `META-INF/MANIFEST.MF` at the beginning of the generated JAR package. CzechIdM loads all available ConnId connectors during initialization. ConnId tries to read the manifest of all connectors, because there is information about the supported version of ConnId framework. However, reading of manifests assumes that the manifest is first or second file of the JAR archive (this is assumed in Oracle JDK). If the manifest is not at the beginning, ConnId fails to read it with NullPointerException, so it doesn't load the connector and so CzechIdM fails to start.

The steps above are a workaround for these problems. The configuration of `maven-jar-plugin` ensures that the MANIFEST.MF generated during the `jar` goal contains additional properties for the ConnId framework (connector name, version, framework version), which is in standard Maven build added during the `assembly` goal. But I found no way to enforce the correct order of the files in JAR. Using existing MANIFEST.MF (POM property `manifestFile`) didn't help. Removing the folder `META-`

`INF/maven/` from the JAR couldn't be done in POM (`addMavenDescriptor=false` seems to have no effect in Eclipse, `maven-shade-plugin` is not started in Eclipse build). So in the end, I remove it from the deployed JAR. Note that the folder `META-INF/maven/` will be added again during hot deploy, but since the connectors are already initialized, it's not a problem.

Of course, any improvements are welcome.

By standard Maven build I mean running `mvn clean install -Dlicense.skip=true` from the command line.

Stucked Eclipse settings

If the error you see during the build is not one of frequently known, you can try deleting ".settings" folder for concrete module. Best way to restart module settings is to delete problematic module, then do Project → clean and then delete the ".settings" folder and then try to import module into Eclipse again. You can try deleting other Eclipse folders starting with dot to be sure that build for this module will be clear.

Troubleshooting - stuck Eclipse module build

Sometimes, Eclipse stucks after pulling new version of an external (non-core) module. The module's automatic build stucks on around 77% and the application cannot be closed.

The following guide describes recovery from such situation for development environment on Windows.

Suppose that an external module [idm-extmod] is stuck.

1. Eclipse menu: switch off Project → Build automatically
2. End task Eclipse in Windows Task manager
3. Start Eclipse
4. Right-click on idm-app module → Maven → Update project → click "Select all" → OK (this removes "czechidm-extmod" from idm-app's Deployment assembly)
5. Right-click on [idm-extmod] → Delete → leave unchecked the "Delete project content on disk" checkbox
6. Go to `[ModuleDir]/Realization/backend/[idm-extmod]` and remove the `.settings` and `target` dirs and `.classpath` and `.project` files
7. Go to `[ModuleDir]/Realization/frontend/[idm-extmod]` and remove the `node_modules` shortcut
8. In Eclipse: File → Import → Existing Maven Project, select the [ModuleDir]/Realization/backend/[idm-extmod] directory
9. Right-click on idm-app module → Properties → Deployment Assembly → Add, select the [idm-extmod] module
10. Eclipse menu: switch on Project → Build automatically
11. Right-click on idm-app module → Maven → Update project → click "Select all" → OK (this removes "czechidm-extmod" from idm-app's Deployment assembly again)
12. Repeat step 8: Right-click on idm-app module → Properties → Deployment Assembly → Add,

select the [idm-extmod] module

13. Run the server
14. Reconfigure the test/debug environment: Right-click on [idm-extmod] → Properties → Java Build Path → Libraries → click on Classpath → Add Library → JUnit 5 → Apply and Close