Maven plugins

celerio, springfuse, metadata, packer

Jaxio

Maven plugins: celerio, springfuse, metadata, packer

Jaxio

v3.0.49-SNAPSHOT

Abstract

Celerio plugins

Legal notice

Table of Contents

List of Tables

Chapter 1. maven-celerio-plugin

celerio:generate

Full name

`com.jaxio.celerio:maven-celerio-plugin:3.0.44-SNAPSHOT:generate`

Description

The core Celerio Engine is invoked by this plugin. This plugin can either connect directly to a database and extract the metadata information or use the metadata.xml file produced by the maven-db-metadata-plugin:extract-metadata goal. Please refer to the section called “dbmetadata:extract-metadata”.

Dependency

- Since version: `1.0.0`.
- Binds by default to the lifecycle phase: `generate-sources`.

Attributes

- `baseDir` : The current folder
- `jdbcCatalog` : Specify the JDBC catalog.
- `jdbcDriver` : Specify the JDBC driver. Example: `org.postgresql.Driver`
- `jdbcOracleRetrieveRemarks` : Should the Oracle remarks be retrieved ? Please note that this will impact the speed of the reverse engineering of your database.
- `jdbcOracleRetrieveSynonyms` : Should the synonyms be retrieved ?
- `jdbcPassword` : Specify the JDBC password.
- `jdbcSchema` : Specify the JDBC schema.
- `jdbcUrl` : Specify the JDBC url. Example: `jdbc:h2:~/.h2/sampledatabase`
- `jdbcUser` : Specify the JDBC user, this user needs to have the privilege to access the database metadata.
- `outputDirectory` : The output folder.
- `project` : Maven project, this is by default the current maven project.
- `skip` : Should the source code generation be skipped ? This is a common pattern in Maven, where you can skip plugins using profiles to fully adapt your build.
- `xmlConfiguration` : The relative path to the Maven Celerio configuration file. The default value is `src/main/config/maven-celerio-plugin/maven-celerio-plugin.xml`

- `xmlMetadata` : The relative path to the metadata.xml file produced by the maven-db-metadata-plugin:extract-metadata goal. If this file exists it will be used, otherwise Celerio will access the database directly. The main purpose of this file is to speed-up the generation process, as for large database schema the reverse engineering takes time. An other very important benefit of this feature is to store the file in your source control, thus having a reproducible build.
- `xmlTemplatePacksOverride` : The relative path to a Maven Celerio configuration file dedicated to override the template packs definition present in the main xml configuration file. This configuration file is useful when working on multi-modules project. Indeed you can set for each module exactly the template packs that should be used. Keep in mind that only the template packs definition will be extracted from this file. The default value is `src/main/config/maven-celerio-plugin/celerio-template-packs.xml`

Table 1.1. `celerio:generate` plugin attributes list

Name	Type	Expression	Default
baseDir	String	${basedir}	
jdbcCatalog	String	${jdbc.catalog}	
jdbcDriver	String	${jdbc.driver}	
jdbcOracleRetrieveRemarks	boolean	${jdbc.oracleRetrieveRemarks}	false
jdbcOracleRetrieveSynonyms	boolean	${jdbc.oracleRetrieveSynonyms}	true
jdbcPassword	String	${jdbc.password}	
jdbcSchema	String	${jdbc.schema}	
jdbcUrl	String	${jdbc.url}	
jdbcUser	String	${jdbc.user}	
outputDirectory	String	${maven-celerio-plugin.directory}	${basedir}
project	MavenProject	${project}	
skip	boolean	${maven-celerio-plugin.skip}	false
xmlConfiguration	String	${maven-celerio-plugin.configuration}	${basedir}/src/main/config/maven-celerio-plugin/maven-celerio-plugin.xml
xmlMetadata	String	${maven-celerio-plugin.xml.metadata}	${basedir}/src/main/config/maven-celerio-plugin/metadata.xml
xmlTemplatePacksOverride	String	${maven-celerio-plu-	${basedir}/src/main/con-

Name	Type	Expression	Default
		gin.packs.config uration}	fig/ maven-cel- erio-plu- gin/cel- erio-tem- plate-packs.xml

generate goal examples

With explicit configuration

To use metadata plugin in your maven build, add the plugin definition with the jdbc information to access the database from which you want to extract the model in xml.

```
<plugin>
  <groupId>com.jaxio.celerio</groupId>
  <artifactId>maven-celerio-plugin</artifactId>
  <version>3.0.0</version>
  <executions>
    <execution>
      <id>Generates files using the extracted database schema.</id> ❶
      <goals>
        <goal>generate</goal>
      </goals>
    </execution>
    <configuration>                                                ❷
      <jdbcUrl>jdbc:h2:~/.h2/sampledatabase</jdbcUrl>
      <jdbcUser>myuser</jdbcUser>
      <jdbcPassword>myuser</jdbcPassword>
    </configuration>
  </executions>
  <dependencies>
    <dependency>                                                   ❸
      <groupId>com.jaxio.celerio.packs</groupId>
      <artifactId>backend</artifactId>
      <version>3.0.0</version>
    </dependency>
    <dependency>                                                   ❹
      <groupId>com.h2database</groupId>
      <artifactId>h2</artifactId>
      <version>1.2.125</version>
    </dependency>
  </dependencies>
</plugin>
```

❶ The `id` element is mostly used for documentation as it is displayed during the build process.
❷ Specification of the configuration by stating explicitly all the parameters
❸ Specify the dependency on the packs to be used when producing the files.
❹ Please note that you need to specify the dependency for your jdbc driver to the plugin.

With maven properties

```
<properties>                                                       ❶
  <jdbc.driver>org.h2.Driver</jdbc.driver>
  <jdbc.url>jdbc:h2:~/.h2/sampledatabase</jdbc.url>
  <jdbc.user>myuser</jdbc.user>
  <jdbc.password>myuser</jdbc.password>
</properties>
<build>
  <plugins>
    <plugin>
      <groupId>com.jaxio.celerio</groupId>
      <artifactId>maven-celerio-plugin</artifactId>
      <version>3.0.0</version>
      <executions>
        <execution>
          <id>Generates files using the extracted database schema.</id>
          <goals>
            <goal>generate</goal>
          </goals>
        </execution>
      </executions>                                                     ❷
      <dependencies>
        <dependency>
          <groupId>com.jaxio.celerio.packs</groupId>
          <artifactId>backend</artifactId>
          <version>3.0.0</version>
        </dependency>
        <dependency>
          <groupId>com.h2database</groupId>
          <artifactId>h2</artifactId>
          <version>1.2.125</version>
        </dependency>
      </dependencies>
    </plugin>
  </plugins>
</build>
```

❶ Specify maven properties, they will be picked-up by the metadata plugin.
❷ No need of configuration, as all the required properties are available as maven properties.

Generation logs

For each file generated, the Celerio `generate` goal writes a one-line log message giving status information. The status line has the following format: [<template pack>][<message>:<generated file path>]

The template pack is simply the template pack name.

The message indicates how the generation went; it could be of the form

- `GENERATE` : The target file does not exists yet, Celerio generates it normaly.
- `GENERATE AND REPLACE` : The target file is already present, it is not under user's control, and it is different from the one that Celerio wants to generate. Celerio replaces the file with its new version.
- `GENERATE IN COLLISIONS FODLER` : The target file is already present, it is under user's control, and it is different from the one that Celerio wants to generate. Celerio leaves the user's file where it is and generate its file's version in the collisions folder.
- `IDENTICAL EXISTS` : The file already exists and is indentical to the file that should have been generated. Celerio does not replace it to preserve the timestamp of the original file and for obvious performance reasons.
- `COPY` : The target file, a static resource such an image, does not exist yet. Celerio copy it normally.

- `COPY IN COLLISION FOLDER` : The target file is already present, it is under user control and it is different from the one that Celerio wants to copy. Celerio leaves the user's file where it is and copy its file's version in the collisions folder.

celerio:cleanGenerated

Full name

`com.jaxio.celerio:maven-celerio-plugin:3.0.44-SNAPSHOT:cleanGenerated`

Description

This goal deletes the files that were previously generated by Celerio. Files that have been modified (file size has changed or date of last modification has changed) or file that have been added to SVN since their generation are not deleted. The list of candidate files for deletion is stored under the folder ./celerio

Dependency

- Since version: `1.0.0`.
- Binds by default to the lifecycle phase: `clean`.

Attributes

- `baseDir` : The current folder
- `project` : Maven project, this is by default the current maven project.
- `skip` : Should the clean goal be skipped ? This is a common pattern in Maven, where you can skip plugins using profiles to fully adapt your build.

Table 1.2. `celerio:cleanGenerated` plugin attributes list

Name	**Type**	**Expression**	**Default**
`baseDir`	`String`	`${basedir}`	
`project`	`MavenProject`	`${project}`	
`skip`	`boolean`	`${maven-celerio-plugin.cleanGenerated.skip}`	`false`

Chapter 2. maven-dbmetadata-plugin

dbmetadata:extract-metadata

Full name

```
com.jaxio.celerio:maven-dbmetadata-plugin:3.0.44-SNAPSHOT:extract-meta
data
```

Description

This plugin connects to a relational database using JDBC and reverses the database schema meta data. The reverse engineering consists in serializing the information returned by the JDBC driver into an XML file (see the documentation of the java.sql.DatabaseMetaData for more information). The `metadata.xml` file produced by this plugin is used by Celerio's generate goal. Please refer to the the section called "celerio:generate" .

Dependency

- Since version: `3.0.0`.
- Binds by default to the lifecycle phase: `generate-sources`.

Attributes

- `jdbcCatalog` : Specify the JDBC catalog.
- `jdbcDriver` : Specify the JDBC driver class. Example: `org.postgresql.Driver`
- `jdbcOracleRetrieveRemarks` : Should the Oracle remarks be retrieved ? Please note that this will impact the speed of the reverse engineering of your database.
- `jdbcOracleRetrieveSynonyms` : Should the synonyms be retrieved ?
- `jdbcPassword` : Specify the JDBC password.
- `jdbcSchema` : Specify the JDBC schema.
- `jdbcUrl` : Specify the JDBC url to connect to your database. Make sure that you connect with enough privileges to access the meta data information. Example: `jdbc:h2:~/.h2/sampledatabase`
- `jdbcUser` : Specify the JDBC user, this user needs to have the privilege to access the database metadata.
- `skip` : Should the database meta data extraction be skipped ? This is a common pattern in Maven, where you can skip plugins using profiles to fully adapt your build.
- `targetFilename` : The fully qualified name of the XML file created by this plugin.

Table 2.1. `dbmetadata:extract-metadata` plugin attributes list

Name	Type	Expression	Default
`jdbcCatalog`	`String`	`${jdbc.catalog}`	
`jdbcDriver`	`String`	`${jdbc.driver}`	
`jdbcOracleRetrieveRemarks`	`boolean`	`${jdbc.oracleRetrieveRemarks}`	`false`
`jdbcOracleRetrieveSynonyms`	`boolean`	`${jdbc.oracleRetrieveSynonyms}`	`true`
`jdbcPassword`	`String`	`${jdbc.password}`	
`jdbcSchema`	`String`	`${jdbc.schema}`	
`jdbcUrl`	`String`	`${jdbc.url}`	
`jdbcUser`	`String`	`${jdbc.user}`	
`skip`	`boolean`	`${maven-celerio-plugin.skip}`	`false`
`targetFilename`	`String`	`${maven-metadata-plugin.targetFilename}`	`${basedir}/src/main/config/maven-celerio-plugin/metadata.xml`

Examples

With explicit configuration

To use `dbmetadata` plugin in your Maven build, add the plugin definition with the JDBC information to access the database from which you want to extract the meta data.

```
<plugin>
  <groupId>com.jaxio.celerio</groupId>
  <artifactId>maven-dbmetadata-plugin</artifactId>
  <version>3.0.11</version>
  <executions>
    <execution>
      <id>Extract the database schema.</id>                    ❶
      <goals>
        <goal>extract-metadata</goal>
      </goals>
    </execution>
    <configuration>                                             ❷
      <jdbcUrl>jdbc:h2:~/.h2/sampledatabase</jdbcUrl>
      <jdbcUser>myuser</jdbcUser>
      <jdbcPassword>myuser</jdbcPassword>
    </configuration>
  </executions>
  <dependencies>
    <dependency>                                                ❸
      <groupId>com.h2database</groupId>
      <artifactId>h2</artifactId>
      <version>1.2.125</version>
    </dependency>
  </dependencies>
</plugin>
```

❶ The `id` element is mostly used for documentation as it is displayed during the build process.
❷ Specification of the configuration by stating explicitly all the parameters
❸ Please note that you need to specify the dependency for your JDBC driver to the plugin.

Important

The `jdbcUser` must have enough privileges to access the database meta-information. Otherwise Celerio cannot reverse the database and generate the code.

With maven properties

The metadata plugin can get its data from the maven properties.

This allows you to keep the plugin configuration short and reuse the `jdbc.user` property elsewhere, for example when working with the `sql-maven-plugin` to recreate your database schema.

Important

Using properties is really convenient so you can switch configuration using maven profiles

You can refer to the list of configuration elements. .

Celerio can get its jdbc information from your current properties

```
<properties>                                                          ❶
  <jdbc.driver>org.h2.Driver</jdbc.driver>
  <jdbc.url>jdbc:h2:~/.h2/sampledatabase</jdbc.url>
  <jdbc.user>myuser</jdbc.user>
  <jdbc.password>myuser</jdbc.password>
</properties>
<build>
  <plugins>
    <plugin>
      <groupId>com.jaxio.celerio</groupId>
      <artifactId>maven-dbmetadata-plugin</artifactId>
      <version>3.0.0</version>
      <executions>
        <execution>
          <id>Extract the database schema.</id>
          <goals>
            <goal>extract-metadata</goal>
          </goals>
        </execution>
      </executions>                                                   ❷
      <dependencies>
        <dependency>
          <groupId>com.h2database</groupId>
          <artifactId>h2</artifactId>
          <version>1.2.125</version>
        </dependency>
      </dependencies>
    </plugin>
  </plugins>
</build>
```

❶ Specify maven properties, they will be picked-up by the metadata plugin.
❷ No need of configuration, as all the required properties are available as maven properties.

This configuration has the same effect as the previous example.

Chapter 3. maven-packer-plugin

packer:pack

Full name

`com.jaxio.celerio:maven-packer-plugin:3.0.44-SNAPSHOT:pack`

Description

(no description)

Dependency

- Binds by default to the lifecycle phase: `generate-resources`.

Attributes

- `folderToPack` : (no description)
- `skip` : skip process ?
- `targetPackFile` : (no description)

Table 3.1. `packer:pack` plugin attributes list

Name	Type	Expression	Default
`folderToPack`	`String`	`${basedir}/src/main/templates`	
`skip`	`boolean`	`false`	
`targetPackFile`	`String`		

Chapter 4. maven-springfuse-plugin

springfuse:extract-metadata

Full name

```
com.springfuse:maven-springfuse-plugin:3.0.44-SNAPSHOT:extract-metadat
a
```

Description

This plugin connects to a relational database using JDBC and reverses the database schema metadata. The reverse engineering consists in serializing the information returned by the JDBC driver into an XML file (see the documentation of the java.sql.DatabaseMetaData for more information). The metadata.xml file produced by this plugin is used by Celerio's generate goal. Please refer to the the section called “celerio:generate” .

Dependency

- Since version: `3.0.0`.
- Binds by default to the lifecycle phase: `generate-sources`.

Attributes

- `jdbcCatalog` : Specify the JDBC catalog.
- `jdbcDriver` : Specify the JDBC driver class. Example: `org.postgresql.Driver`
- `jdbcOracleRetrieveRemarks` : Should the Oracle remarks be retrieved ? Please note that this will impact the speed of the reverse engineering of your database.
- `jdbcOracleRetrieveSynonyms` : Should the synonyms be retrieved ?
- `jdbcPassword` : Specify the JDBC password.
- `jdbcSchema` : Specify the JDBC schema.
- `jdbcUrl` : Specify the JDBC url to connect to your database. Make sure that you connect with enough privileges to access the meta data information. Example: `jdbc:h2:~/.h2/sampledatabase`
- `jdbcUser` : Specify the JDBC user, this user needs to have the privilege to access the database metadata.
- `skip` : Should the database meta data extraction be skipped ? This is a common pattern in Maven, where you can skip plugins using profiles to fully adapt your build.
- `targetFilename` : The fully qualified name of the XML file created by this plugin.

Table 4.1. `springfuse:extract-metadata` plugin attributes list

Name	Type	Expression	Default
`jdbcCatalog`	`String`	`${jdbc.catalog}`	
`jdbcDriver`	`String`	`${jdbc.driver}`	
`jdbcOracleRetrieveRemarks`	`boolean`	`${jdbc.oracleRetrieveRemarks}`	`false`
`jdbcOracleRetrieveSynonyms`	`boolean`	`${jdbc.oracleRetrieveSynonyms}`	`true`
`jdbcPassword`	`String`	`${jdbc.password}`	
`jdbcSchema`	`String`	`${jdbc.schema}`	
`jdbcUrl`	`String`	`${jdbc.url}`	
`jdbcUser`	`String`	`${jdbc.user}`	
`skip`	`boolean`	`${maven-celerio-plugin.skip}`	`false`
`targetFilename`	`String`	`${maven-metadata-plugin.targetFilename}`	`${basedir}/src/main/config/maven-celerio-plugin/metadata.xml`

springfuse:remote-generation

Full name

`com.springfuse:maven-springfuse-plugin:3.0.44-SNAPSHOT:remote-generation`

Description

Generate a metadata from the database, and send it to the remote service for generation.

Dependency

- Since version: `3.0.0`.
- Binds by default to the lifecycle phase: `generate-sources`.
- Requires that Maven runs in online mode.

Attributes

- `appName` : (no description)

- `celerioVersion` : Specify the celerio version, if not set the latest version will be used
- `extractionLocation` : springfuse extraction of the zip
- `frontEnd` : Front end selection. Can be either none, springMvc, primeFacesSpringWebFlow, tbd...
- `generationServiceLocation` : springfuse generation service location
- `interactive` : Will the bootstrap ask interactively for appName and rootPackage?
- `jdbcCatalog` : Specify the JDBC catalog.
- `jdbcDriver` : Specify the JDBC driver class. Example: `org.postgresql.Driver`
- `jdbcOracleRetrieveRemarks` : Should the Oracle remarks be retrieved ? Please note that this will impact the speed of the reverse engineering of your database.
- `jdbcOracleRetrieveSynonyms` : Should the synonyms be retrieved ?
- `jdbcPassword` : Specify the JDBC password.
- `jdbcSchema` : Specify the JDBC schema.
- `jdbcUrl` : Specify the JDBC url to connect to your database. Make sure that you connect with enough privileges to access the meta data information. Example: `jdbc:h2:~/.h2/sampledatabase`
- `jdbcUser` : Specify the JDBC user, this user needs to have the privilege to access the database metadata.
- `login` : springfuse login
- `password` : springfuse password
- `proxyEnable` : Http proxy
- `proxyHost` : Http proxy host
- `proxyNtlmDomain` : (no description)
- `proxyNtlmEnable` : (no description)
- `proxyNtlmWorkstation` : (no description)
- `proxyPassword` : Http proxy password
- `proxyPort` : Http proxy port
- `proxyUsername` : Http proxy username
- `rootPackage` : (no description)
- `skip` : Should the database meta data extraction be skipped ? This is a common pattern in Maven, where you can skip plugins using profiles to fully adapt your build.
- `targetFilename` : The fully qualified name of the XML file created by this plugin.
- `xmlConfiguration` : The fully qualified name of the celerio configuration file
- `xmlMetadata` : The fully qualified name of the celerio configuration

- `zipResultLocation` : springfuse generation zip result location

Table 4.2. `springfuse:remote-generation` plugin attributes list

Name	Type	Expression	Default
appName	String	${maven-remote-generation-plugin.appName}	${project.artifactId}
celerioVersion	String	${maven-remote-generation-plugin.celerioVersion}	${celerioVersion}
extractionLocation	String	${maven-remote-generation-plugin.extractionLocation}	${basedir}/..
frontEnd	String	${maven-remote-generation-plugin.frontEnd}	springMvc
generationServiceLocation	String	${maven-remote-generation-plugin.generationServiceLocation}	http://v3.springfuse.com/remote/generate
interactive	String	${maven-remote-generation-plugin.interactive}	false
jdbcCatalog	String	${jdbc.catalog}	
jdbcDriver	String	${jdbc.driver}	
jdbcOracleRetrieveRemarks	boolean	${jdbc.oracleRetrieveRemarks}	false
jdbcOracleRetrieveSynonyms	boolean	${jdbc.oracleRetrieveSynonyms}	true
jdbcPassword	String	${jdbc.password}	
jdbcSchema	String	${jdbc.schema}	
jdbcUrl	String	${jdbc.url}	
jdbcUser	String	${jdbc.user}	
login	String	${maven-remote-generation-plugin.login}	${springfuse.login}
password	String	${maven-remote-generation-plugin.password}	${springfuse.password}
proxyEnable	boolean	${maven-remote-generation-plugin.proxy.enable}	
proxyHost	String	${maven-remote-generation-plu-	

Name	Type	Expression	Default
		gin.proxy.host}	
proxyNtlmDomain	String	${maven-remote-generation-plugin.proxy.ntlm.domain}	
proxyNtlmEnable	boolean	${maven-remote-generation-plugin.proxy.ntlm.enable}	
proxyNtlmWorkstation	String	${maven-remote-generation-plugin.proxy.ntlm.workstation}	
proxyPassword	String	${maven-remote-generation-plugin.proxy.password}	
proxyPort	int	${maven-remote-generation-plugin.proxy.port}	8080
proxyUsername	String	${maven-remote-generation-plugin.proxy.username}	
rootPackage	String	${maven-remote-generation-plugin.rootPackage}	${project.groupId}
skip	boolean	${maven-celerio-plugin.skip}	false
targetFilename	String	${maven-metadata-plugin.targetFilename}	${basedir}/src/main/config/maven-celerio-plugin/metadata.xml
xmlConfiguration	String	${maven-remote-generation-plugin.configuration}	${basedir}/src/main/config/maven-celerio-plugin/maven-celerio-plugin.xml
xmlMetadata	String	${maven-remote-generation-plugin.xml.metadata}	${basedir}/src/main/config/maven-celerio-plugin/metadata.xml
zipResultLocation	String	${maven-remote-generation-plugin.zipResultLoc	${basedir}/target/

Name	**Type**	**Expression**	**Default**
		`ation}`	