

Celerio

maven plugins

Jaxio

Celerio: maven plugins

Jaxio

v3.0.49-SNAPSHOT

Copyright © 2005-2011 Jaxio

Abstract

Celerio plugins

Legal notice

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from Jaxio.

All copyright, confidential information, patents, design rights and all other intellectual property rights of whatsoever nature contained herein are and shall remain the sole and exclusive property of Jaxio SARL. The information furnished herein is believed to be accurate and reliable.

However, no responsibility is assumed by Jaxio for its use, or for any infringements of patents or other rights of third parties resulting from its use.

All other trademarks are the property of their respective owners.

Table of Contents

1. maven-celerio-plugin	1
celerio:generate	1
Full name	1
Description	1
Dependency	1
Attributes	1
generate goal examples	3
With explicit configuration	3
With maven properties	3
Generation logs	4
celerio:cleanGenerated	5
Full name	5
Description	5
Dependency	5
Attributes	5

List of Tables

1.1. celerio:generate plugin attributes list	2
1.2. celerio:cleanGenerated plugin attributes list	5

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 ???.

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

Name	Type	Expression	Default
		gin.packs.configuration}	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>
</properties>

```



```
<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>
      <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 indential 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
<code>baseDir</code>	<code>String</code>	<code>\${basedir}</code>	
<code>project</code>	<code>MavenProject</code>	<code>\${project}</code>	
<code>skip</code>	<code>boolean</code>	<code>\${maven-celerio-plugin.cleanGenerated.skip}</code>	<code>false</code>