MyBatis Maven Migration Plugin 1.0.0 - Reference Documentation

Marco Speranza (MyBatis.org)

Chapter 1. Introduction

1.1. Why a Maven Plugin - Motivation

Being *Apache Maven* users to manage projects lyfecycle, and being at the same time *MyBatis* framework and *MyBatis Migration tool* to maintain the database structure, in order to have a fully integrated tool we started implementing a solution that mixes the *MyBatis Migration tool* power into a standard *Apache Maven* build life cycle.

The *MyBatis maven-migration-plugin* aims to perform a complete porting of all *MyBatis Migration tool* commands to a standard Maven plugin.

1.2. A little bit of history

The MyBatis maven-migration-plugin was born in April 2010 after a Skype conference between Marco Speranza and Simone Tripodi where both were looking for a solution to integrate MyBatis Migration tool into the softrware build life cycle. The project was created and maintained by 99soft.org and posted on Google Code with the name maven-migrate-plugin.

In September 2010 Simone proposed Marco as new MyBatis community member, Clinton Begin and the team accepted letting him bringing the *maven-migration-plugin* code base to become an official MyBatis subproject.

By that day, the MyBatis Migration Maven Plugin project is maintained by the MyBatis.org team.

1.3. Requirements

Before starting reading the manual, it is very important you're familiar with both MyBatis Migration Tool and Apache Maven and therminology, otherwise it would be very difficult to understand the described context.

1.4. Maven 3 and site generation

A significant difference between M2 and M3 is site generation. If you are used to configure the `pom` reporting section to generate code reports, you should migrate this feature to Maven 3. Indeed, the `reporting` and `reportSets` sections have been deprecated (it won't cause an error with Maven 3, they will just be ignored), and have been replaced by the `reportPlugins` section in the configuration block of the `maven-site-plugin` itself. A typical Maven 2 `pom.xml` file looks like this:

```
...
<reporting>
     <plugins>
         <plugin>
             <groupId>org.mybatis.maven</groupId>
             <artifactId>maven-migration-plugin</artifactId>
             <version>1.0.0</version>
             <configuration>
                 <repository> [migration repository path] </repository>
             </configuration>
         </plugin>
     </plugins>
 </reporting>
...
```

If you are using M3 for your project, users have to add the following snippet in order to generate MyBatis Migration Plugin report automatically.

```
...
<build>
     <plugins>
       <plugin>
           <groupId>org.apache.maven.plugins</groupId>
           <artifactId>maven-site-plugin</artifactId>
           <version>3.0-beta-3</version>
           <configuration>
               <reportPlugins>
                   <plugin>
                       <groupId>org.mybatis.maven</groupId>
                       <artifactId>maven-migration-plugin</artifactId>
                       <version>1.0.0</version>
                       <configuration>
                           <repository> [migration repository path] </repository>
                       </configuration>
                   </plugin>
               </reportPlugins>
           </configuration>
       </plugin>
   </plugins>
 </build>
...
```

1.5. Acknowledgements

A special thanks goes to all the special people who made the Maven Migration Plugin a reality, above all Clinton Begin, who's strongly supporting the subprojects proliferation.

Chapter 2. Usage

2.1. Introduction

In this chapter some examples will show how to use the Migration Plugin goals:

2.2. Generic Plugin configuration information

Before starting, please take a look at the following links to know more how to include and configure Maven plugins in the POM project:

- Configuring Plugins
- Plugin Development
- Plugin Prefix

2.3. `pom.xml` Plugin configuration

To use Migration Maven plugin in your project you have to configure your `pom.xml` file like this:

```
...
<plugins>
    <plugin>
        <groupId>org.mybatis.maven</groupId>
        <artifactId>maven-migration-plugin</artifactId>
        <version>1.0.0</version>
        <configuration>
            <repository> [migration repository path] </repository>
        </configuration>
        <dependencies>
            [ add your jdbc driver depencency ]
        </dependencies>
    </plugin>
  ...
<plugins>
```

2.4. The `migration:status` goal

This goal prints the current migration status of database. A tipical output could be:

```
mvn migrate:status -Dmigration.path=/path/to/migration/repository

  ...

[INFO] Executing  Apache Migration StatusCommand
[INFO] ID             Applied At          Description
[INFO] ================================================================================
[INFO] 20100400000001    ...pending...    create changelog
[INFO] 20100400000002    ...pending...    first migration
[INFO] 20100400000003    ...pending...    second migration
[INFO]
...
```

2.5. The `migration:check` goal

checks the current status of your database migration and fails if one or more script are pending. A typical use of this goal is check the migration status into your maven build life cycle:

```
<plugin>
    <groupId>org.mybatis.maven</groupId>
    <artifactId>maven-migration-plugin</artifactId>
    <version>1.0.0</version>
    <configuration>
        <repository> [migration repository path] </repository>
    </configuration>
    <executions>
    <execution>
        <id>migration-chack</id>
        <phase>test</phase>
        <goals>
          <goal>check</goal>
        </goals>
      </execution>
    </executions>
    <dependencies>
        <dependency> [your jdbc dependency] </dependency>
    </dependencies>
</plugin>
```

and then

```
mvn clean test
```

this goal fails if migration plugin founds one or more pending script. To *skip* the migration check set the properties `migration.skip` like this:

```
mvn -Dmigration.skip=true clean test
```

2.6. Site report for Maven 2 projects

You can configure your `pom.xml` to create a simple report of your database status:

```
...
<build>
    ...
    <plugins>
        <plugin>
            <groupId>org.mybatis.maven</groupId>
            <artifactId>maven-migration-plugin</artifactId>
            <version>1.0.0</version>
            <configuration>
                <repository> [migration repository path] </repository>
            </configuration>
            <executions>
                <execution>
                    <id>migration-chack</id>
                    <phase>test</phase>
                    <goals>
                        <goal>check</goal>
                    </goals>
                </execution>
            </executions>
            <dependencies>
                <dependency> [your jdbc dependency] </dependency>
            </dependencies>
        </plugin>
        ...
</build>

<reporting>
```

```
    <plugins>
        <plugin>
            <groupId>org.mybatis.maven</groupId>
            <artifactId>maven-migration-plugin</artifactId>
            <version>1.0.0</version>
            <configuration>
                <repository> [migration repository path] </repository>
            </configuration>
        </plugin>
    </plugins>
</reporting>
```

once the plugin is configured, users can invoke:

```
mvn site
```

2.7. Site report for Maven 3 projects

You can configure your `pom.xml` to create a simple report of your database status:

```
...
<build>
    ...
    <plugins>
        <plugin>
            <groupId>org.mybatis.maven</groupId>
            <artifactId>maven-migration-plugin</artifactId>
            <version>1.0.0</version>
            <configuration>
                <repository> [migration repository path] </repository>
            </configuration>
            <executions>
                <execution>
                    <id>migration-chack</id>
                    <phase>test</phase>
                    <goals>
                        <goal>check</goal>
                    </goals>
                </execution>
            </executions>
            <dependencies>
                <dependency> [your jdbc dependency] </dependency>
            </dependencies>
        </plugin>
        <plugin>
             <groupId>org.apache.maven.plugins</groupId>
             <artifactId>maven-site-plugin</artifactId>
             <version>3.0-beta-3</version>
             <configuration>
                   <reportPlugins>
                       <plugin>
                           <groupId>org.mybatis.maven</groupId>
                           <artifactId>maven-migration-plugin</artifactId>
                           <version>1.0.0</version>
                           <configuration>
                               <repository> [migration repository path] </repository>
                           </configuration>
                       </plugin>
                   </reportPlugins>
               </configuration>
           </plugin>
    </plugins>
    ...
```

once the plugin is configured, users can invoke:

```
mvn site
```

Chapter 3. Examples

3.1. Intregrate migrate in Maven life cycle

It can be useful integrate Migration steps into a build life cycle, i.e. integrating the database schema creation into a CI build lifecycle.

```
...
<plugins>
    <plugin>
        <groupId>org.mybatis.maven</groupId>
        <artifactId>maven-migration-plugin</artifactId>
        <version>1.0.0</version>
        <dependencies>
            [ add your jdbc driver depencency ]
        </dependencies>
        <executions>
          <execution>
            <id>apply-all-pending-migration</id>
            <phase>process-test-resources</phase>
            <goals>
                <goal>up</goal>
            </goals>
            <configuration>
                 <repository> [migration repository path] </repository>
            </configuration>
          </execution>
       </executions>
    </plugin>
  ...
<plugins>
```

If your project uses the sub-modules you can set your parent pom in this way:

```
...
<build>
    <pluginManagement>
        <plugins>
            <plugin>
                <groupId>org.mybatis.maven</groupId>
                <artifactId>maven-migration-plugin</artifactId>
                <version>1.0.0</version>
                <dependencies>
                    [ add your jdbc driver depencency ]
                </dependencies>
                <configuration>
                    <skip>true</skip>
                </configuration>
            </plugin>
        </plugins>
    </pluginManagement>

    <plugins>
        <plugin>
            <groupId>org.mybatis.maven</groupId>
            <artifactId>maven-migration-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Then, in submodules:

```
...
<plugins>
    <plugin>
        <groupId>org.mybatis.maven</groupId>
        <artifactId>maven-migration-plugin</artifactId>
        <configuration>
            <repository> [migration repository path] </repository>
            <skip>false</skip>
        </configuration>
        <executions>
            <execution>
                <id>migration-chack</id>
                <phase>test</phase>
                <goals>
                    <goal>check</goal>
                </goals>
                <inherited>false</inherited>
            </execution>
        </executions>
    </plugin>
  ...
<plugins>
```

3.2. Use Migration commands

Migration plugin aims to help you to administer your database via MyBatis Schema Migration, so, i.e, you can use this plugin to create new migration scripts and to apply the pending scripts to your database. The following example will show the common commands to create and administer your database.

First of all you have to initialize your migration repository:

```
mvn migration:init -Dmigration.path=/path/to/repository
```

This command inizializes the standard migration repository into `/path/to/repository` folder. After you have to modify and customize your enviroment: edit the file `/path/to/repository/enviroments/development.properties` and set the database coordinte. So you can apply the first migration to the database:

```
mvn migration:up -Dmigration.path=/path/to/repository
```

Now it is possible to create the new script:

```
mvn migration:new -Dmigration.path=/path/to/repository -Dmigration.description=my_second_schema_migration
```

the command creates a new empty sql file into the folder `/path/to/repository/scripts` like this:

```
--// First migration.
-- Migration SQL that makes the change goes here.

--//@UNDO
-- SQL to undo the change goes here.
```

now you can check the current status of your database by executing this the status goal:

```
mvn migration:status -Dmigration.path=/path/to/repository

[INFO] Executing  Apache Migration StatusCommand
[INFO] ID             Applied At            Description
[INFO] ================================================================================
[INFO] 20100400000001    2010-04-24 22:51:16    create changelog
[INFO] 20100400000002    2010-04-24 22:51:17    first migration
[INFO] 20100400000003    ...pending...          my second schema migration
[INFO]
```

finally you can apply the last migration pending script:

```
mvn migration:up -Dmigration.path=/path/to/repository

mvn migration:status -Dmigration.path=/path/to/repository

[INFO] Executing  Apache Migration StatusCommand
[INFO] ID             Applied At            Description
[INFO] ================================================================================
[INFO] 20100400000001    2010-04-24 22:51:16    create changelog
[INFO] 20100400000002    2010-04-24 22:51:17    first migration
[INFO] 20100400000003    2010-04-24 23:14:07    my second schema migration
[INFO]
```

3.3. Migration plugin report

3.4. Migration plugin report with warnings

Appendix A. Plugin Goals

A.1. list

Table A.1. Plugin goals list

Goal	Report?	Description
migration:bootstrap	No	Goal which execute the MyBatis migration bootstrap command.
migration:check	No	Goal which check the presence of pending migration.
migration:down	No	Goal which execute the MyBatis migration status command.
migration:init	No	Goal which executes the MyBatis migration init command. Init command creates a new migrate repository into 'repository' location.
migration:new	No	Goal which executes the MyBatis migration new command.
migration:pending	No	Goal which execute the MyBatis migration pending command.
migration:script	No	Goal which executes the MyBatis migration script command.
migration:status	No	Goal which execute the MyBatis migration status command.
migration:status-report	Yes	Extends AbstractMavenReport. Class to generate a maven report.
migration:up	No	Goal which execute the MyBatis migration status command.
migration:version	No	Goal which execute the MyBatis migration version command.

A.2. Commons parameters

All goal have the follow parameter:

Table A.2. List of common parameter

Name	Default value	Expression	Description
environment	development	migration.env	Environment to configure.
force	false	migration.force	Forces script to continue even if SQL errors are encountered.
repository	.	migration.path	Location of migrate repository.
skip	false	migration.skip	Skip migration actions.

A.3. Goal with extra parameter

A.3.1. migration:check

Table A.3. migration:check

Name	Default value	Expression	Description
downSteps	1	migration.down.steps	Steps to do. (type ALL to apply all down steps).

A.3.2. migration:new

Table A.4. migration:new

Name	Default value	Expression	Description
description	No default value!	migration.description	New file description.
template	No default value!	migration.template	Alternative template file description.

A.3.3. migration:script

Table A.5. migration:script

Name	Default value	Expression	Description
v1	No default value!	migration.v1	Initial version
v2	No default value!	migration.v2	Final version
output	No default value!	migration.output	The output file to be

Name	Default value	Expression	Description
			create.

A.3.4. migration:status-report

Table A.6. migration:status-report

Name	Default value	Expression	Description
aggregate	false	migration.aggregate	Aggregate report results.

A.3.5. migration:up

Table A.7. migration:up

Name	Default value	Expression	Description
upSteps	all steps	migration.up.steps	Steps to do

A.3.6. migration:version

Table A.8. migration:version

Name	Default value	Expression	Description
version	No default value!	migration.version	Version string.

Appendix B. Frequenly Asked Questions

B.1. What is MyBatis Migration Schema?

MyBatis Migrations Schema is a tool that will change the way you manage changes to your database. You can watch an amazing introductive video!!!

B.2. How can I configure my JDBC connection?

Users can configure their preferred JDBC connection in this way:

```
...
<plugins>
    <plugin>
        <groupId>org.mybatis.maven</groupId>
        <artifactId>maven-migration-plugin</artifactId>
        <version>1.0.0</version>
        <configuration>
            <repository> [your migration repository] </repository>
        </configuration>
        <dependencies>
            <dependency>
                <groupId>org.apache.derby</groupId>
                <artifactId>derby</artifactId>
                <version>10.5.3.0_1</version>
            </dependency>
        </dependencies>
    </plugin>
  ...
<plugins>
```

B.3. How can I skip the migration goals?

Simply setting the property `migration.skip` to `true`, i.e.

```
mvn -Dmigration.skip=true clean test
```