

CODE BREEZE !

liquibase - helloworld example

Sheng W

10:39 PM

7 Comments

Liquibase is a database change management tool. Rather than writing SQL directly against the database to create, update or drop database objects, developers define their desired database changes in XML files.

Any change to database are grouped into "**ChangeSet**", the best practice is one changeset per modification to make roll back easily. Changes to database can be tagged. e.g, you can tag you database structure to 1.0 after first release. Later, when some patches are made and ver 1.1 is release, you can tag all changes up to now to 1.1. (If it's not very clear now, it's ok, see the examples below will make it more obvious). With the help of those tags, you can easy rollback you database structure back to a certain version. (Also, liquibase can roll back without tags).

One notion need to be clarified first, liquibase **only manage schema changes of your database**, e.g. add extra index or rename a column, **the data in the tables are not managed!**

1. Basic concepts

ChangeSet is a logic group in which you can put any real operation. For example, a change set can has operations to create a table, rename a column, add foreign key or any other database operations.

How does liquibase identify a change set? changeset is identified by 3 elements, id + author + change log filename(with path). When run liquibase first time, it will create 2 extra tables in your database, **databasechangelog** and **databasechangeloglock**.

ID	AUTHOR	FILENAME	DATEEXECUTED	ORDEREXECUTED	TAG	MD5SUM
create_department	sheng.w	liquibase/db-changelog-1.0.xml	2016-04-05 17:03:44	1 EXECUTED	(Null)	7:eb367949
create_employee	sheng.w	liquibase/db-changelog-1.0.xml	2016-04-05 17:03:44	2 EXECUTED	(Null)	7:2ecf9f5bf
tag-1.0	sheng.w	liquibase/db-changelog-1.0.xml	2016-04-05 17:03:44	3 EXECUTED	1.0	7:55606556
rename_dept_column	sheng.w	liquibase/db-changelog-1.1.xml	2016-04-05 17:03:44	4 EXECUTED	(Null)	7:13015e85
tag-1.1	sheng.w	liquibase/db-changelog-1.1.xml	2016-04-05 17:03:44	5 EXECUTED	1.1	7:44457298
add-fk-between-emp-and-dept	sheng.w	liquibase/db-changelog-1.2.xml	2016-04-05 17:03:44	6 EXECUTED	(Null)	7:988c8cba
tag-1.2	sheng.w	liquibase/db-changelog-1.2.xml	2016-04-05 17:03:44	7 EXECUTED	1.2	7:e6e2007d

Liquibase will go through changelog xml file, see if there are some change sets not in this table. If found, execute them and put a record in this table. By using this table, liquibase can trace which changeset has already executed, which changeset is new. Tags can be used to specify a version you want to go, see below example for more. To

Search this Site...

Categories

Popular

Labels

- ActiveMQ
- Avro
- byte
- byte array
- Camel
- Concurrency
- Database
- debug
- Eclipse
- fiddler
- Flume
- Generic Type
- H2
- Hibernate
- High Level Concurrency
- HTTPS
- Integration test
- IO/NIO
- Java 8
- Java SE
- JEE
- Jersey
- JUnit
- JMS

use liquibase, you don't need to touch this *databasechangelog* table, but it can help you understand how liquibase works.

To use liquibase, you also need a **change log** file, in which all database operations are defined. In this tutorial, liquibase 3.4 and xml based change log is used.

2. How to run liquibase

Before the demo starts, let's first see how to run liquibase. In this tutorial, 2 ways are introduced, by command line or by maven plugin.

To run liquibase in command line, you need

- download liquibase, unpack it executable file *liquibase* or *liquibase.bat* in the package.
- download your database jdbc driver to you local disk.

To run liquibase by using maven, you need:

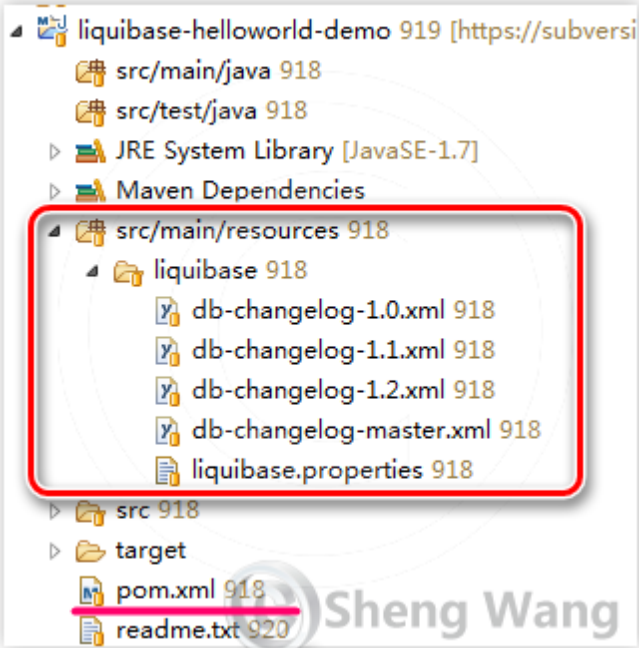
- change pom file, add ***liquibase-maven-plugin***

Since in pom.xml the jdbc driver dependency has already be added, you don't need the external jdbc jar file.

You can choose either command line or maven plugin to run liquibase. I personally perfer by maven plugin, cause the command can be much shorter.

3. Hello world demo for liquibase usage

First let's create a maven project in eclipse. In this demo there's no java class. We emphase on how to use liquibase. The hierarchy of the demo project looks like below.



Let's go through these files one by one.

3.1 pom.xml

First the pom.xml, to add liquibase plugin. If you decide no to use maven to run liquibase, this step can be omitted.

- [JPA](#)
- [JSF](#)
- [JUnit](#)
- [Kafka](#)
- [Lambda](#)
- [log](#)
- [logback](#)
- [Low Level Concurrency](#)
- [maven](#)
- [Netty](#)
- [Oracle](#)
- [Performance Test](#)
- [PowerMock](#)
- [Regular Expression](#)
- [REST](#)
- [Selenium](#)
- [Spark](#)
- [Spring](#)
- [Spring boot](#)
- [Spring JMS](#)
- [Spring MVC](#)
- [Spring Schedule](#)
- [Spring Security](#)
- [Spring Test](#)
- [SSL](#)
- [Stream](#)
- [Test](#)
- [Thrift](#)
- [Unit Test](#)

Recent Posts

Blog Archive

- ▶ [2017 \(4\)](#)
- ▼ [2016 \(26\)](#)
 - ▶ [August \(1\)](#)

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://maven.apache.org/xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
2   <modelVersion>4.0.0</modelVersion>
3
4   <groupId>com.sanss.demo</groupId>
5   <artifactId>liquibase-helloworld-demo</artifactId>
6   <version>1.0</version>
7   <packaging>jar</packaging>
8
9   <name>liquibase-helloworld-demo</name>
10  <url>http://maven.apache.org</url>
11
12  <properties>
13    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
14  </properties>
15
16  <dependencies>
17    <!-- MySQL -->
18    <dependency>
19      <groupId>mysql</groupId>
20      <artifactId>mysql-connector-java</artifactId>
21      <version>5.1.6</version>
22    </dependency>
23  </dependencies>
24
25  <build>
26    <finalName>liquibase-helloworld-demo</finalName>
27    <plugins>
28      <!-- Use Java 1.7 -->
29      <plugin>
30        <groupId>org.apache.maven.plugins</groupId>
31        <artifactId>maven-compiler-plugin</artifactId>
32        <version>2.5.1</version>
33        <configuration>
34          <source>1.7</source>
35          <target>1.7</target>
36        </configuration>
37      </plugin>
38
39      <!-- User liquibase plugin -->
40      <plugin>
41        <groupId>org.liquibase</groupId>
42        <artifactId>liquibase-maven-plugin</artifactId>
43        <version>3.4.2</version>
44        <configuration>
45          <propertyFile>liquibase/liquibase.properties</propertyFile>
46          <changeLogFile>liquibase/db-changelog-master.xml</changeLogFile>
47        </configuration>
48        <!-- I personally prefer run it manually -->
49        <executions>
50          <execution>
51            <phase>process-resources</phase>
52            <goals>
53              <goal>update</goal>
54            </goals>
55          </execution>
56        </executions>
57      </plugin>
58    </plugins>
59  </build>
60 </project>
```

- ▼ April (3)
- How to dump all dependencies of a maven project
- Understand <optional>true</optional> in maven depe...
- liquibase - helloworld example
- March (10)
- February (8)
- January (4)
- 2015 (41)
- 2014 (13)

I personally prefer not to bind it to any maven build lifecycle, but invoke it manually. There are 2 files configured in this plugin, "properties file" defines all parameters to connect a database. "changeLogFile" is the file from which it reads the change sets.

3.1 liquibase.properties

This file has all connection parameters. Here is the liquibase.properties file in this demo.

```
1 # MySQL
2 driver=com.mysql.jdbc.Driver
3 url=jdbc:mysql://localhost:3306/spring
4 username=root
5 password=yourPwdToDatabase
```

Nothing fancy here, just common database connection parameters.

3.2 ChangeLog files

In this demo, change log files are in xml format. Other avaiable formats are json and yaml.

The official recommand best practice is always using a xxxx-master.xml file as an entry file. This is also the file set in the maven plugin. In this db-changelog-master.xml file, there's no real logic defined, only a bunch of includes.

```
1 <databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbchan
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangel
4                       http://www.liquibase.org/xml/ns/dbchange
5
6   <include file="liquibase/db-changelog-1.0.xml"/>
7   <include file="liquibase/db-changelog-1.1.xml"/>
8   <include file="liquibase/db-changelog-1.2.xml"/>
9 </databaseChangeLog>
```

The included files have all change sets. Suppose the file *db-changelog-1.0.xml* is the database structure for release version 1.0.

```
1 <databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbcha
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangel
4                       http://www.liquibase.org/xml/ns/dbchang
5
6   <changeSet id="create_department" author="sheng.w">
7     <createTable tableName="department">
8       <column name="id" type="int">
9         <constraints primaryKey="true" nullable="false" />
10      </column>
11      <column name="name" type="varchar(50)">
12        <constraints nullable="false" />
13      </column>
14    </createTable>
15  </changeSet>
16
17  <changeSet id="create_employee" author="sheng.w">
18    <createTable tableName="employee">
19      <column name="id" type="int">
20        <constraints primaryKey="true" nullable="false" />
21      </column>
22      <column name="emp_name" type="varchar(50)">
23        <constraints nullable="false" />
24      </column>
25      <column name="dept" type="int"/>
26    </createTable>
27  </changeSet>
28
29  <changeSet id="tag-1.0" author="sheng.w">
30    <tagDatabase tag="1.0" />
31  </changeSet>
32
33 </databaseChangeLog>
```

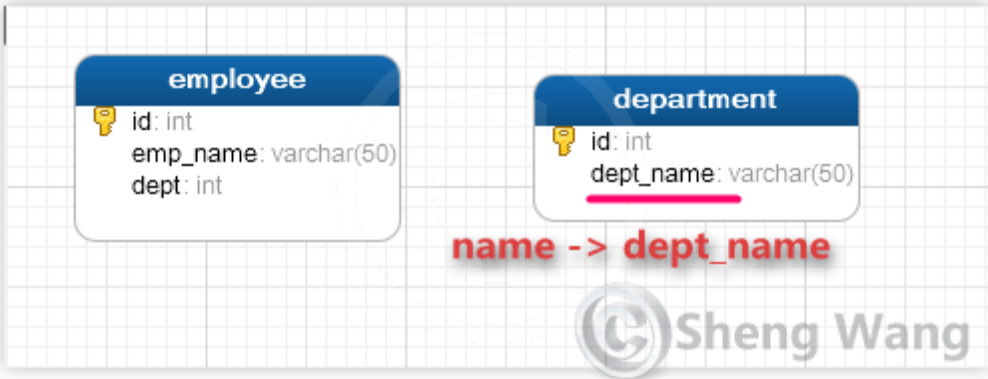
There are 3 change sets in our 1.0 database schema. Two tables are created and a tag for version 1.0 is added in the end. Every change set has an id and an author. This xml file demonstrate how to create table and primary key for it. The result up to version 1.0 is 2 tables in the database.



Let suppose later on, 2 new versions are released with a little change to the database, 1.1 and 1.2. Every version has a xml file, defining what has changed since last time. The db-changelog-1.1.xml change column 'name' of table 'department' to 'dept_name'

```
1 <databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbcha
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangel
4                       http://www.liquibase.org/xml/ns/dbchang
5
6   <changeSet id="rename_dept_column" author="sheng.w">
7     <renameColumn tableName="department" oldColumnName="name" n
8   </changeSet>
9
10  <changeSet id="tag-1.1" author="sheng.w">
11    <tagDatabase tag="1.1" />
12  </changeSet>
13
14 </databaseChangeLog>
```

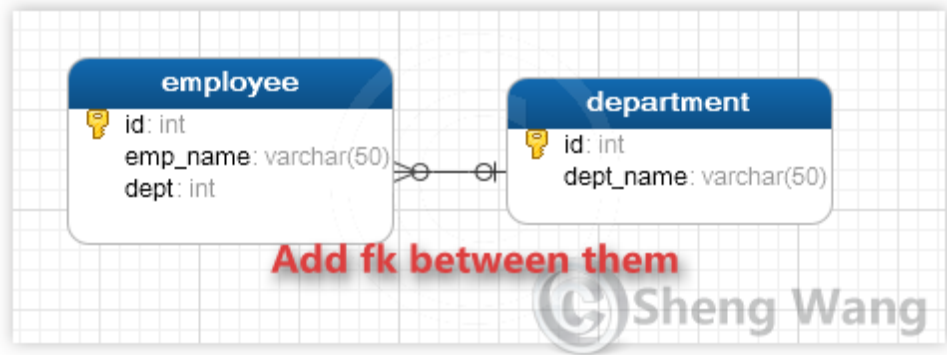
The result up to 1.1 in database is



Later in version 1.2, one index adds to empolyee table, one foreign key adds between employee and department.

```
1 <databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbcha
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangel
4                       http://www.liquibase.org/xml/ns/dbchang
5
6   <changeSet id="add-fk-between-emp-and-dept" author="sheng.w">
7     <addForeignKeyConstraint constraintName="fk_emp_dept"
8       baseTableName="employee" baseColumnNames="dept" reference
9       referencedColumnNames="id" onDelete="CASCADE" onUpdate="C
10  </changeSet>
11
12  <changeSet id="add_index" author="sheng.w">
13    <createIndex tableName="employee" indexName="idx_exp_name">
14      <column name="emp_name"/>
15    </createIndex>
16  </changeSet>
17
18  <changeSet id="tag-1.2" author="sheng.w">
19    <tagDatabase tag="1.2" />
20  </changeSet>
21
22 </databaseChangeLog>
```

Up to version 1.2, the database looks like below.



4. Understand version control of liquibase

Let's now demonstrate the 'version control' function of liquibase. Suppose at beginning we have a clean database with nothing in it. The database change log has 3 versions, 1.0, 1.1 and 1.2 defined in previous chapter. latest version is 1.2.

- version 1.0, create 2 tables
- version 1.1, change column name of table department
- version 1.2, add foreign key and index

4.1 Apply change log to database until latest

First let create database schema to current latest version. Using command line:

```
1 liquibase --defaultsFile=src/main/resources/liquibase/liquibase.
2 --classpath="d:\mysql-connector-java-5.1.6.jar;d:\spring-learning\
3 --changeLogFile=liquibase/db-changelog-master.xml \
4 update
```

The **defaultsFile** specify the location of properties file for database connection. **classpath** specify where to find all necessary java file and xml change log files. Here are 2 jar files, one is mysql jdbc driver, the other is the jar of our demo, from which to read the changelog xml file. **changeLogFile** specify the file name of change log. **update** is the command for liquibase, to update database according to the xml change log file.



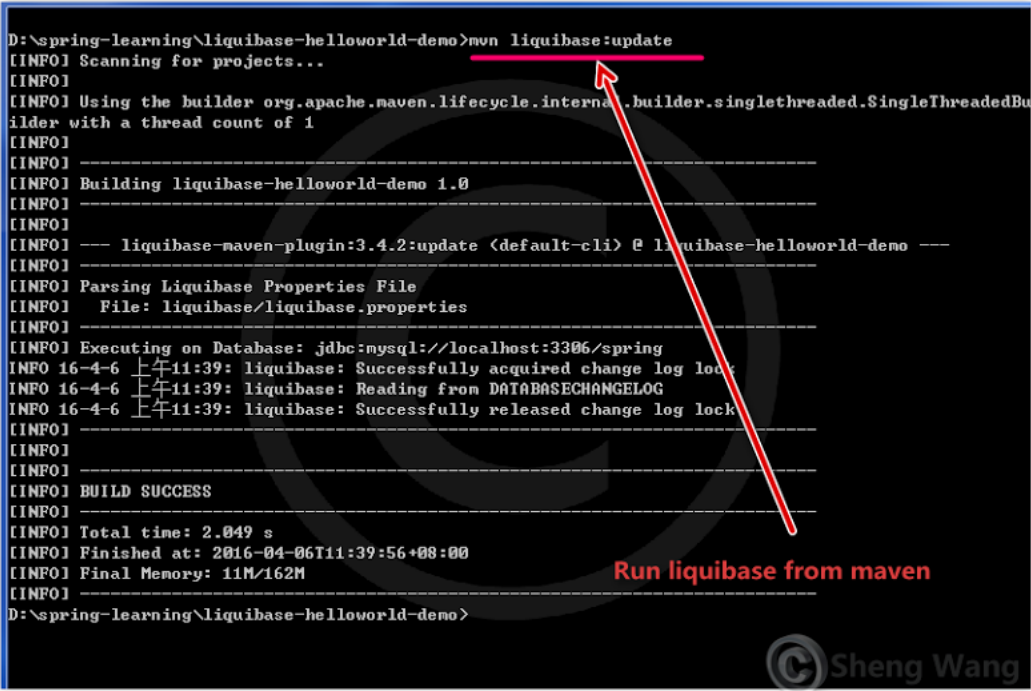
Now check the database, in databasechangelog table, all change set are executed.

The screenshot shows a table named **databasechangelog** in a database. The table has columns: ID, AUTHOR, FILENAME, DATEEXECUTED, TAG, ORDEREXECUTED, EXECUTYPE, and MTIME. The data rows show that all change sets from version 1.0 to 1.2 have been executed successfully. Below the table, the text "All change sets up to latest are applied to database" is written in red. A watermark "Sheng Wang" is visible at the bottom right.

ID	AUTHOR	FILENAME	DATEEXECUTED	TAG	ORDEREXECUTED	EXECUTYPE	MTIME
create_department	sheng.w	liquibase/db-changelog-1.0.xml	2016-04-06 11:34:24	(Null)	1	EXECUTED	7:56
create_employee	sheng.w	liquibase/db-changelog-1.0.xml	2016-04-06 11:34:24	(Null)	2	EXECUTED	7:56
tag-1.0	sheng.w	liquibase/db-changelog-1.0.xml	2016-04-06 11:34:24	1.0	3	EXECUTED	7:56
rename_dept_column	sheng.w	liquibase/db-changelog-1.1.xml	2016-04-06 11:34:24	(Null)	4	EXECUTED	7:56
tag-1.1	sheng.w	liquibase/db-changelog-1.1.xml	2016-04-06 11:34:24	1.1	5	EXECUTED	7:56
add-fk-between-emp-and-dept	sheng.w	liquibase/db-changelog-1.2.xml	2016-04-06 11:34:24	(Null)	6	EXECUTED	7:56
add_index	sheng.w	liquibase/db-changelog-1.2.xml	2016-04-06 11:34:24	(Null)	7	EXECUTED	7:56
tag-1.2	sheng.w	liquibase/db-changelog-1.2.xml	2016-04-06 11:34:24	1.2	8	EXECUTED	7:56

I personally like to run liquibase by maven, because the command is much shorter. The following maven command is equivalent to the previous command line.

```
1 mvn liquibase:update
```



4.2 Rollback database to version 1.0

For some reason you want to roll back you database to version 1.0. You can achieve that by command line

```
1 liquibase --defaultsFile=src/main/resources/liquibase/liquibase
2           --classpath="d:\mysql-connector-java-5.1.6.jar;D:\spr
3           --changeLogFile=liquibase/db-changelog-master.xml \
4           rollback 1.0
```

or by maven command

```
1 mvn liquibase:rollback -Dliquibase.rollbackTag=1.0
```

These 2 ways to are equivalent, just pay attention to how to specify the version tag

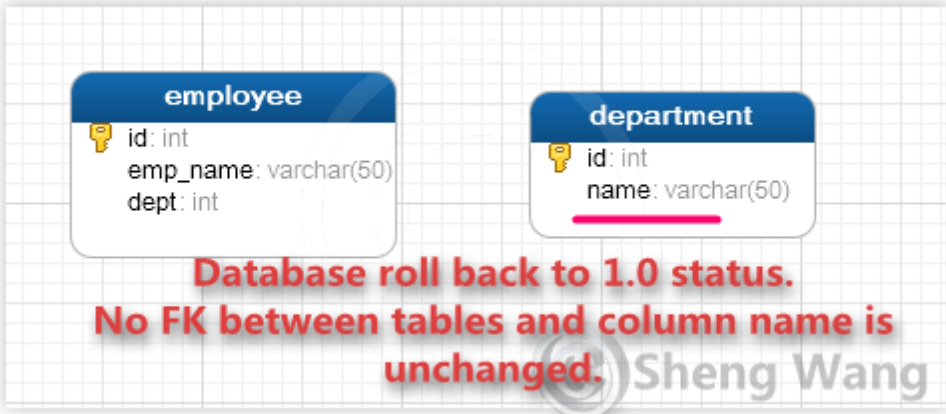
```
D:\spring-learning\liquibase-helloworld-demo>mvn liquibase:rollback -Dliquibase.rollbackTag=1.0
[INFO] Scanning for projects...
[INFO]
[INFO] Using the builder org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuilder with a thread count of 1
[INFO]
[INFO] -----
[INFO] Building liquibase-helloworld-demo 1.0
[INFO] -----
[INFO] --- liquibase-maven-plugin:3.4.2:rollback (default-cli) @ liquibase-helloworld-demo ---
[INFO]
[INFO] Parsing Liquibase Properties File
[INFO]   File: liquibase/liquibase.properties
[INFO] -----
[INFO] Executing on Database: jdbc:mysql://localhost:3306/spring
INFO 16-4-6 上午11:46: liquibase: Successfully acquired change log lock
INFO 16-4-6 上午11:46: liquibase: Reading from DATABASECHANGELOG
INFO 16-4-6 上午11:46: liquibase: null: liquibase/db-changelog-1.2.xml::tag-1.2::sheng.w: Rolling Back Changeset:liquibase/db-changelog-1.2.xml::tag-1.2::sheng.w
INFO 16-4-6 上午11:46: liquibase: null: liquibase/db-changelog-1.2.xml::add_index::sheng.w: Rolling Back Changeset:liquibase/db-changelog-1.2.xml::add_index::sheng.w
INFO 16-4-6 上午11:46: liquibase: null: liquibase/db-changelog-1.2.xml::add-fk-between-emp-and-dept::sheng.w: Rolling Back Changeset:liquibase/db-changelog-1.2.xml::add-fk-between-emp-and-dept::sheng.w
INFO 16-4-6 上午11:46: liquibase: null: liquibase/db-changelog-1.1.xml::tag-1.1::sheng.w: Rolling Back Changeset:liquibase/db-changelog-1.1.xml::tag-1.1::sheng.w
INFO 16-4-6 上午11:46: liquibase: null: liquibase/db-changelog-1.1.xml::rename_dept_column::sheng.w: Rolling Back Changeset:liquibase/db-changelog-1.1.xml::rename_dept_column::sheng.w
INFO 16-4-6 上午11:46: liquibase: Successfully released change log lock
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.179 s
[INFO] Finished at: 2016-04-06T11:46:49+08:00
[INFO] Final Memory: 11M/165M
Now the database rollback to version 1.0
D:\spring-learning\liquibase-helloworld-demo>
```

If you check the databasechangelog table, you will found the changeset after 1.0 are all gone.

ID	AUTHOR	FILENAME	DATEEXECUTED	TAG	ORDEREXECUTED	EXECTYPE
create_department	sheng.w	liquibase/db-changelog-1.0.xml	2016-04-06 11:34:24	(Null)	1	EXECUTED
create_employee	sheng.w	liquibase/db-changelog-1.0.xml	2016-04-06 11:34:24	(Null)	2	EXECUTED
tag-1.0	sheng.w	liquibase/db-changelog-1.0.xml	2016-04-06 11:34:24	1.0	3	EXECUTED

After rollback to 1.0, all change set after 1.0 are all gone

The tables are also revers to what they looks like in verion 1.0, now foreign key and with original column name.



4.3 apply change log to a specified version

Now the database is in status 1.0, and you want to apply 1.1 to it. you can do that by following command.

```
1 liquibase --defaultsFile=src/main/resources/liquibase/liquibase
2 --classpath="d:\mysql-connector-java-5.1.6.jar;D:\spr
3 --changeLogFile=liquibase/db-changelog-master.xml \
4 updateToTag 1.1
```


By using sub command **updateToTag**, you can update database to a certain version tag.



The above command line also equals to following maven command:

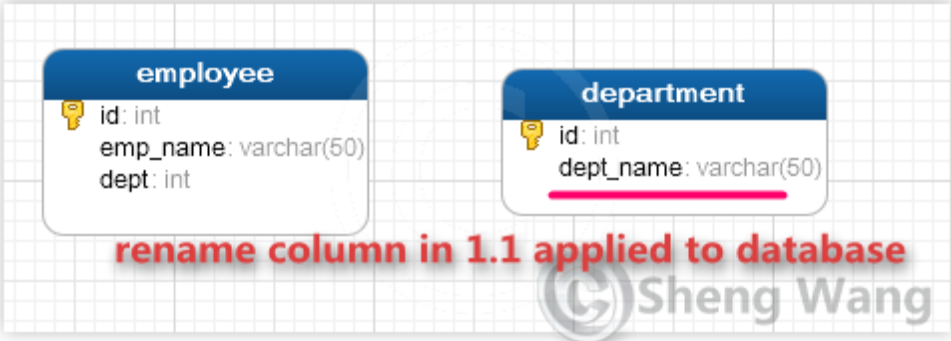
```
1 mvn liquibase:update -Dliquibase.toTag=1.1
```

Let's verify the database.

A screenshot of a database tool interface showing a table named 'databasechangelog'. The table has columns: ID, AUTHOR, FILENAME, DATEEXECUTED, TAG, ORDEREXECUTED, and EXECTYPE. The data rows show five entries, all executed successfully. Below the table, a red text overlay says 'Change sets up to 1.1 are applied to database' and a watermark for 'Sheng Wang' is visible.

ID	AUTHOR	FILENAME	DATEEXECUTED	TAG	ORDEREXECUTED	EXECTYPE
create_department	sheng.w	liquibase/db-changelog-1.0.xml	2016-04-06 11:34:24	(Null)	1	EXECUTED
create_employee	sheng.w	liquibase/db-changelog-1.0.xml	2016-04-06 11:34:24	(Null)	2	EXECUTED
tag-1.0	sheng.w	liquibase/db-changelog-1.0.xml	2016-04-06 11:34:24	1.0	3	EXECUTED
rename_dept_column	sheng.w	liquibase/db-changelog-1.1.xml	2016-04-06 13:09:36	(Null)	4	EXECUTED
tag-1.1	sheng.w	liquibase/db-changelog-1.1.xml	2016-04-06 13:09:36	1.1	5	EXECUTED

Tables ae also in the 1.1 status, no 1.2 foreign key yet, but get 1.1 column rename done.



Now you should have some feelings on how liquibase can do the 'version control'

5. Generate ChangeLog from existent tables

If you already have everything configured in database by hand or sql. You can use liquibase to generate change log file for you, then you can keep working based on the generated xml.

By command line:

```
1 liquibase --defaultsFile=src/main/resources/liquibase/liquibase
2 --classpath="d:\mysql-connector-java-5.1.6.jar;D:\spr
3 --changeLogFile=d:\output.xml \
4 generateChangeLog
```

The changeLogFile is a filename to be created. **The file name must end with ".xml", ".json" or ".yaml".**

By maven plugin:

```
1 mvn liquibase:generateChangeLog -Dliquibase.outputChangeLogFile=
```

The options to specify output name are different in command line and maven plugin.
After running you should be able to find newly create d:\output.xml file

6. Recap

Now you should be able to:

- understand how liquibase workds
- how to create table, pk, fk, index in xml format in change log file
- how to apply change log to datebase
- how to rollback and do version control with liquibase
- how to generate change log from existent tables



[f Facebook](#) [t Twitter](#) [g+ Google+](#) [Stumble](#) [digg Digg](#)

[← Newer Post](#) [Home](#) [Older Post →](#)

7 comments:

Anonymous October 14, 2016 at 3:52 AM

Nice article .

[Reply](#)



Unknown January 12, 2017 at 7:11 AM

Very Good Artikle. Very easy to undestand.. keep it up and thanks..:)

[Reply](#)



devendrareddy m January 20, 2017 at 4:07 AM

Very nice
thanks Sheng W

[Reply](#)

Anonymous April 4, 2017 at 6:15 AM

nice article.
thanks.

[Reply](#)



Unknown June 14, 2017 at 1:43 AM

very nicely written!

[Reply](#)



Unknown June 14, 2017 at 1:44 AM

nice!!!Thanks!

[Reply](#)

Anonymous [August 7, 2017 at 6:03 AM](#)

cool easy to take in. thanks


[Reply](#)




Subscribe to: [Post Comments \(Atom\)](#)

Powered by [Blogger](#).

About The Author





Sheng W

Has been a senior software developer, project manager for 10+ years.
Dedicate himself to Alcatel-Lucent and China Telecom for delivering software solutions.

[View my complete profile](#)