**liquibase学习文档**

# 相关网站

官网：<https://www.liquibase.org>

命令行执行：<https://www.liquibase.org/documentation/command_line.html>

快速开始：

<http://www.cnblogs.com/jimboi/p/8093043.html>

<http://www.liquibase.org/quickstart.html>

spring boot:

<https://blog.csdn.net/xiaojin21cen/article/details/78698308>

生成changelog

<https://www.cnblogs.com/xieshuang/p/8641156.html>

最佳实践：

<https://segmentfault.com/a/1190000016641122>

命令行比较：

<https://www.jianshu.com/p/adbe4eb29c0c>

mvn：

https://blog.csdn.net/xiang\_\_liu/article/details/80800205

# 简介

LiquiBase是一个用于数据库重构和迁移的开源工具，通过日志文件的形式记录数据库的变更，然后执行日志文件中的修改，将数据库更新或回滚到一致的状态。它的目标是提供一种数据库类型无关的解决方案，通过执行schema类型的文件来达到迁移。其有点主要有以下：

支持几乎所有主流的数据库，如MySQL, PostgreSQL, Oracle, Sql Server, DB2等；

支持多开发者的协作维护；

日志文件支持多种格式，如XML, YAML, JSON, SQL等；

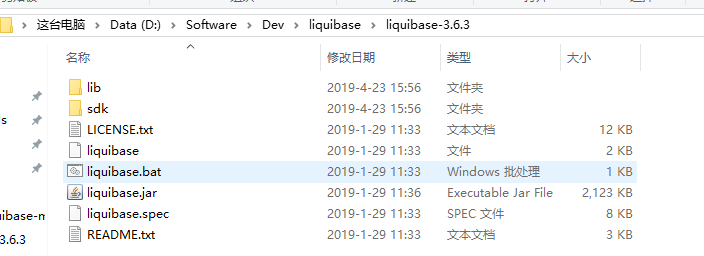
支持多种运行方式，如命令行、Spring集成、Maven插件、Gradle插件等。

更多详情介绍，可以浏览[官网doc](http://www.liquibase.org/documentation/index.html)。

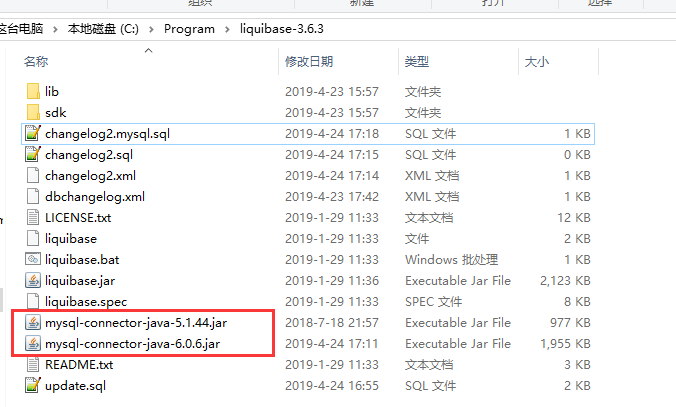
# Hello – 命令行

## 搭建环境

下载liquibase的安装包，下载地址：https://download.liquibase.org，解压到指定目录下



增加数据库连接jar



## 创建Changelog文件dbchangelog.xml

所有的数据库变动都会保存在Changelog文件中

<?xml version="1.0" encoding="UTF-8"?>

<databaseChangeLog

xmlns="http://www.liquibase.org/xml/ns/dbchangelog"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog

http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.1.xsd">

</databaseChangeLog>

## 添加数据库变动changeSet

　　　　每一个ChangeSet元素都是通过id，author，文件名，包全限定名来唯一标识。如果只指定了id属性，那么在有多个人协同开发并且存在多个分支的时候很容易造成ChangeSet重复。

　　　　将每一个changeSet都看做你想要应用到数据库的一个原子操作，通常在一个ChangeSet中最好只包含一个变化，但是如果你想在一个事务中插入多条记录，将它们放在一个ChangeSet中效果会非常好。Liquibase会尝试将每个changeSet放在一个事务中执行，但是很多数据库可能会静默的提交事务，或者为某些命令如（create table，drop table，等）开启新的事物。

<?xml version="1.0" encoding="UTF-8"?>

<databaseChangeLog

xmlns="http://www.liquibase.org/xml/ns/dbchangelog"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog

http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.1.xsd">

<changeSet id="1" author="bob">

<createTable tableName="department">

<column name="id" type="int">

<constraints primaryKey="true" nullable="false"/>

</column>

<column name="name" type="varchar(50)">

<constraints nullable="false"/>

</column>

<column name="active" type="boolean" defaultValueBoolean="true"/>

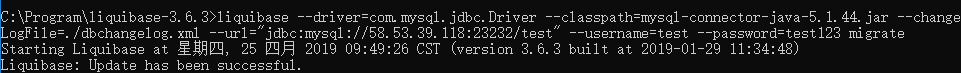
</createTable>

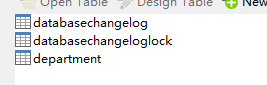
</changeSet>

</databaseChangeLog>

## 运行ChangeSet

liquibase --driver=com.mysql.jdbc.Driver --classpath=mysql-connector-java-5.1.44.jar --changeLogFile=./dbchangelog.xml --url="jdbc:mysql://58.53.39.118:23232/test" --username=test --password=test123 migrate



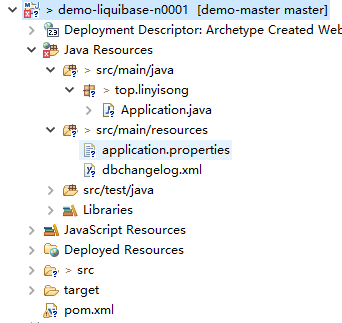


检查你的数据库

　　你会发现你的数据库中多了一个"department"表。而且还创建了两张其它的表“databasechangelog” 和 “databasechangeloglock”。“databasechangelog”表包含了在这个数据库中已经执行的所有的语法的列表。“databasechangeloglock”用了确保两个机器不会尝试在同一时刻修改数据库

# Spring boot中集成

## 项目结构



## pom.xml文件

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>org.lys.demo</groupId>

<artifactId>demo-liquibase-n0001</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>demo-liquibase-n0001 Maven Webapp</name>

<url>http://maven.apache.org</url>

<parent >

<groupId> org.springframework.boot </groupId >

<artifactId> spring-boot-starter-parent </artifactId>

<version>1.4.1.RELEASE</version>

</parent >

<dependencies>

<!--liquibase依赖-->

<dependency>

<groupId>org.liquibase</groupId>

<artifactId>liquibase-core</artifactId>

<version>3.5.3</version>

</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>1.0.24</version>

</dependency>

<!--测试依赖-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

</dependency>

<dependency>

<groupId> org.springframework.boot </groupId >

<artifactId> spring-boot-starter-web </artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

</dependencies>

<build>

<finalName>demo-liquibase-n0001</finalName>

</build>

</project>

## application.properties

spring.datasource.type=com.alibaba.druid.pool.DruidDataSource

spring.datasource.url=jdbc:mysql://58.53.39.118:23232/test?useUnicode=true&characterEncoding=utf-8&allowMultiQueries=true

spring.datasource.username=test

spring.datasource.password=test123

liquibase.change-log= classpath:dbchangelog.xml

liquibase.enabled= true

liquibase.drop-first= false

## Application.java

package top.linyisong;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Application {

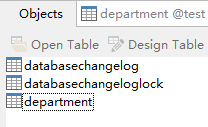
public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

## 启动项目

同样效果

# spring boot 代码配置

package com.aop8.config;

import javax.sql.DataSource;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import liquibase.integration.spring.SpringLiquibase;

@Configuration

public class LiquibaseConfig {

@Bean

public SpringLiquibase liquibase(DataSource dataSource) {

SpringLiquibase liquibase = new SpringLiquibase();

liquibase.setDataSource(dataSource);

liquibase.setChangeLog("classpath:config/liquibase/master.xml");

liquibase.setContexts("development,test,production");

liquibase.setShouldRun(true);

return liquibase;

}

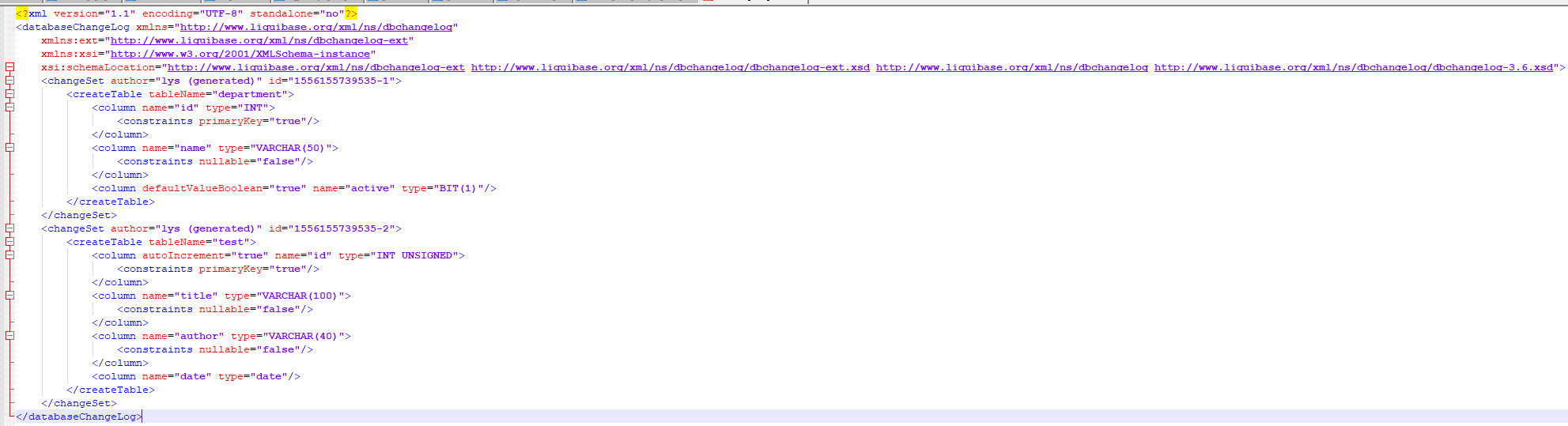
}

# 生成changeLog

通过generateChangeLog生成现有数据库的changeLog文件；

liquibase --driver=com.mysql.jdbc.Driver --classpath=mysql-connector-java-5.1.44.jar --changeLogFile=./dbchangelog.xml --url="jdbc:mysql://58.53.39.118:23232/test" --username=test --password=test123 generateChangeLog





可以看到xml的大致结构为

<databaseChangeLog>

<changeSet>

</changSet>

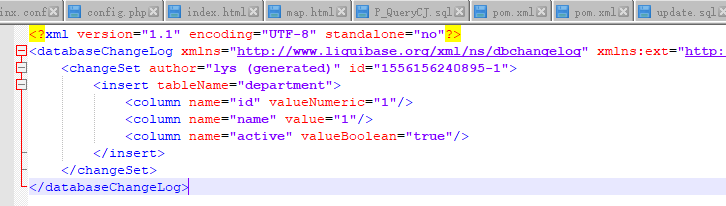
</databaseChangeLog>

changeSet中记录变化内容

generateChangeLog命令默认只会创建数据库结构的变更日志文件，如果希望创建插入数据的变更日志文件，可以使用参数diffTypes，该参数包括如下可选项：

* tables [DEFAULT]表
* columns [DEFAULT] 列
* views [DEFAULT] 视图
* primaryKeys [DEFAULT] 主键
* indexes [DEFAULT] 索引
* foreignKeys [DEFAULT]
* sequences [DEFAULT]
* data  
  例如生成数据：

liquibase --driver=com.mysql.jdbc.Driver --classpath=mysql-connector-java-5.1.44.jar --changeLogFile=./dbchangelog.xml --url="jdbc:mysql://58.53.39.118:23232/test" --username=test --password=test123 --diffTypes=data generateChangeLog



# 命令行命令

详见：<https://www.liquibase.org/documentation/command_line.html>

## 生成

liquibase --driver=com.mysql.jdbc.Driver --classpath=mysql-connector-java-5.1.44.jar --changeLogFile=./dbchangelog.xml --url=" jdbc:mysql://58.53.39.118:23232/test " --username=test --password=test123 generateChangeLog

## 使用changelog生成update sql

liquibase --driver=com.mysql.jdbc.Driver --classpath=mysql-connector-java-5.1.44.jar --changeLogFile=./dbchangelog.xml --url="jdbc:mysql://58.53.39.118:23232/test" --username=test --password=test123 --outputFIle=./update.sql updateSQL

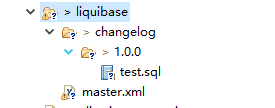
## 比较数据库差异生成changelog,另存为changelog.xml

liquibase --classpath=mysql-connector-java-6.0.6.jar --url=jdbc:mysql://58.53.39.118:23232/test --username=test --password=test123 --changeLogFile=changelog2.msql.sql diffChangeLog --referenceUrl=jdbc:mysql://58.53.39.118:23232/test2 --referenceUsername=test --referencePassword= test123

test为修改后的版本 tets2为修改前的版本

# 其他

## Master.xml组织



<includeAll path=*"changelog/"* relativeToChangelogFile=*"true"*/>