Liquibase

Version control for Database

## Integration with Spring Boot

#### 1.1 Dependencies

Below dependency must be added to spring boot project pom

|  |
| --- |
| <dependency>  <groupId>org.liquibase</groupId>  <artifactId>liquibase-core</artifactId>  <version>${liquibase-core.version}</version> </dependency> |

#### 1.2 Enable Liquibase at Startup

Add the below property to application properties/yml file

|  |
| --- |
| *# Liquibase properties* liquibase:  enabled: true  change-log: /db/db.changelog-dev.xml |

*Change Log File* – The root of all Liquibase changes is the changelog file. Liquibase uses a changelog to sequentially list all changes made to your database. It is a file that contains a record of all your database changes (changesets). Liquibase uses this changelog record to audit your database and execute any changes that are not yet applied to your database.

#### 1.3 Change Log

db.changelog-dev.xml – This is master file, we can include all file under a folder or explicitly mention which changelog should be part of master changelog.

|  |
| --- |
| <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-3.8.xsd">  *<!--<includeAll path="db/changelog/dev"/>-->* <include file="db/changelog/dev/V\_1.0.xml"/>  <include file="db/changelog/dev/V\_2.0.xml"/>  <include file="db/changelog/dev/V\_3.0.xml"/> </databaseChangeLog> |

#### 1.4 JPA config

Application.yml

|  |
| --- |
| *# Application Properties* spring:  application:  name: Liquibase POC  *# Jpa Properties* datasource:  url: jdbc:mysql://localhost:3306/lqpoc  username: root  password: Chandu!1  driver-class-name: com.mysql.cj.jdbc.Driver   jpa:  hibernate:  ddl-auto: none |

#### 1.5 Plugin Configuration

Add below plugin to run Liquibase maven commands

|  |
| --- |
| <plugin>  <groupId>org.liquibase</groupId>  <artifactId>liquibase-maven-plugin</artifactId>  <version>${liquibase-maven-plugin.version}</version>  <configuration>  <propertyFile>src/main/resources/application-${activeProfile}.yml</propertyFile>  <outputChangeLogFile>src/main/resources/db/db.changelog-${activeProfile}.xml</outputChangeLogFile>  <changeLogFile>src/main/resources/db/db.changelog-${activeProfile}.xml</changeLogFile>  <diffChangeLogFile>src/main/resources/db/changelog/${activeProfile}/${maven.build.timestamp}\_changelog.xml</diffChangeLogFile>  <logging>info</logging>  </configuration> </plugin> |

**Example1:** Change log can also be generated using this plugin from an existing database with below command:

mvn liquibase:generateChangeLog

**Example2:** Change log can also be generated for the differences between two databases (like dev vs test) with below command:

mvn liquibase:diff

## DDL

Change Types are the kinds of changes Liquibase does to database schema. Below is the example for **table creation** change type. There are other change types like createIndex, createView, createSequence etc.

|  |
| --- |
| <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-3.8.xsd">  <changeSet author="saichandu" id="changelog-1.0">  <createTable tableName="PERSON">  <column name="NAME" type="VARCHAR2(50)">  <constraints nullable="false" primaryKey="true" primaryKeyName="PERSON\_PK1"/>  </column>  <column name="PHONE" type="VARCHAR2(10)">  <constraints nullable="true" primaryKey="false" unique="false"/>  </column>  </createTable>  </changeSet>  </databaseChangeLog> |

Below logs would be observed:

2021-liquibase.changelog : Table PERSON created

2021-liquibase.changelog : ChangeSet db/changelog/dev/V\_1.0.xml::changelog-1.0::saichandu ran successfully in 30ms

## DML

Below is the example of adding a column, modifying primary key. There are other dml change types like addAutoIncrement, addPrimaryKey, addForeignKeyConstraint, renameColumn, renameTable, modifyDataType etc.

|  |
| --- |
| <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-3.8.xsd">  <changeSet id="changelog-2.0" author="saichandu">  <addColumn tableName="PERSON">  <column name="CREATED\_DT" type="datetime" valueDate="current\_datetime" defaultValueDate="current\_datetime">  <constraints nullable="false"/>  </column>  </addColumn>  </changeSet>  <changeSet id="changelog-2.1" author="saichandu">  <dropPrimaryKey tableName="PERSON" constraintName="PERSON\_PK1"/>  <addPrimaryKey tableName="PERSON" columnNames="NAME,PHONE" constraintName="PERSON\_PK1"/>  </changeSet>  <changeSet id="changelog-2.2" author="saichandu">  <createTable tableName="ADDRESS">  <column name="ID" type="NUMERIC">  <constraints nullable="false" primaryKey="true"/>  </column>  <column name="CITY" type="TEXT"/>  <column name="STATE" type="TEXT"/>  <column name="COUNTRY" type="TEXT"/>  </createTable>  </changeSet> </databaseChangeLog> |

Below logs would be observed:

2021-liquibase.changelog : Columns CREATED\_DT(datetime) added to PERSON

2021-liquibase.changelog : ChangeSet db/changelog/dev/V\_2.0.xml::changelog-2.0::saichandu ran successfully in 31ms

2021-liquibase.changelog : Primary key dropped from PERSON

2021-liquibase.changelog : Primary key added to PERSON (NAME,PHONE)

2021-liquibase.changelog : ChangeSet db/changelog/dev/V\_2.0.xml::changelog-2.1::saichandu ran successfully in 64ms

2021-liquibase.changelog : Table ADDRESS created

2021-liquibase.changelog : ChangeSet db/changelog/dev/V\_2.0.xml::changelog-2.2::saichandu ran successfully in 15ms

## Executing procedures

The createProcedure Change Type is better to use for creating procedures than the raw SQL command. However, we can call external file using change type ‘sqlFile’. We can call procedure using ‘sql’ change type as shown below. Using ‘sql’ change type we can write and execute any sql.

|  |
| --- |
| <changeSet id="changelog-3.0" author="saichandu" dbms="mysql" runInTransaction="true">   <createProcedure dbms="mysql" encoding="UTF-8"  path="procedures/procedure-1.sql"  procedureName="person\_alter\_column" relativeToChangelogFile="true">  </createProcedure>  <sql>call person\_alter\_column() </sql> </changeSet> |

|  |
| --- |
| <changeSet id="changelog-3.0" author="saichandu" runOnChange="true" runInTransaction="true">  <sqlFile path="procedures/procedure-1.sql"  encoding="UTF-8"  relativeToChangelogFile="true"  endDelimiter=";"  splitStatements="true"/>  </changeSet> |

Below logs would be observed:

2021-liquibase.changelog : Stored procedure created

2021-liquibase.changelog : Custom SQL executed

2021-liquibase.changelog : ChangeSet db/changelog/dev/V\_3.0.xml::changelog-3.0::saichandu ran successfully in 58ms

## Data Loading

We can load metadata or initial data using below change set. We can specify the context/environment in which the change log has to be executed. This way we can achieve selective deployments to specific environments.

|  |
| --- |
| <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-3.8.xsd">  <changeSet author="saichandu" id="changelog-1.1" context=”dev”>  <insert tableName="PERSON">  <column name="NAME" value="John"/>  <column name="PHONE" value="3145577889"/>  </insert>  </changeSet> </databaseChangeLog> |

Below logs would be observed:

2021-liquibase.changelog : New row inserted into PERSON

2021-liquibase.changelog : ChangeSet db/changelog/dev/V\_1.1.xml::changelog-1.1::saichandu ran successfully in 0ms

## Rollbacks

#### 6.1 Multistatement rollback

We can enclose more than one instruction as shown below:

|  |
| --- |
| <changeSet id="changelog-4.0" author="saichandu">  <rollback>  <dropTable tableName="PERSON"/>  <dropTable tableName="ADDRESS"/>  </rollback>  </changeSet> |

#### 6.2 Multiple rollback tags

We can split the rollback into multiple parts as shown below:

|  |
| --- |
| <changeSet id="changelog-4.0" author="saichandu">  <rollback>  <dropTable tableName="PERSON"/>  </rollback>  <rollback>  <dropTable tableName="ADDRESS"/>  </rollback>  </changeSet> |

#### 6.3 Reference another changeset

Different change set can be given as reference as shown below. This will reduce the code duplication and can correctly revert the done changes

|  |
| --- |
| <changeSet id="changelog-4.0" author="saichandu">  <dropTable tableName="PERSON"/>  <dropTable tableName="ADDRESS"/>  <rollback changeSetId="changelog-2.0" changeSetAuthor="saichandu"/>  </changeSet> |

## Best practices

#### 7.1 Precondition

It helps in:

* + Document what assumptions the author of the changelog had when creating it
  + Enforce that those assumptions are not violated by users running the changelog
  + Perform data checks before performing an unrecoverable change such as dropTable
  + Control what changesets are run and not run based on the state of the database

We can add multiple precondition statements combining with logical operators. Default operator is AND.

Example1:

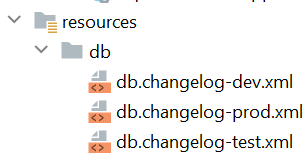
|  |
| --- |
| <preConditions onFail="WARN">  <and>  <dbms type="mysql"/>  <runningAs username="root"/>  </and> </preConditions> |

Example2:

|  |
| --- |
| <changeSet id="changelog-4.2" author="saichandu">  <preConditions onFail="WARN">  <sqlCheck expectedResult="2">select count(\*) from PERSON</sqlCheck>  </preConditions>  <comment>Verify if no data loaded in PERSON table  </comment> </changeSet> |

#### 7.2 Master file per environment

Maintain master file per environment so that the changesets can be included as per environment.



#### 7.3 Managing stored procedures

Always keep separate changelog for Stored Procedures and add runOnChange=”true”. This flag forces Liquibase to check if the changeset was modified.

#### 7.4 One change per changeset

Create multiple changesets instead of combining them in single changeset to avoid failed auto-commit statements that can leave the database in an unexpected state.