|  |  |  |  |
| --- | --- | --- | --- |
|  | **TABLE OF THE CONTENT** | | |
|  |
| **1** | | **INTRODUCTION** |
| **2** | | **WHAT IS LIQUIBASE** |
| **3** | | **HOW LIQUIBASE IS USED?** |
| **4** | | **CORE CONCEPT OF LIQUIBASE** |
| **5** | | **SETTING UP LIQUIBASE IN SPRING BOOT** |
| **6** | | **EXAMPLE CODE** |
| **7** | | **CONCLUSION** |

**1.INTRODUCTION**

**Liquibase** is an open-source database schema change management tool. It provides an automated way to track, manage, and apply database schema changes. Liquibase helps developers and database administrators ensure that database changes are consistently and reliably applied across different environments.

**2.WHAT IS LIQUIBASE?**

Liquibase is a data base change management tool.. Liquibase is widely used in environments where database changes need to be managed consistently and reliably, particularly in agile development and DevOps practices.

**3. HOW LIQUIBASE IS USED?**

In an application if you modify any tables for any requirements and if you want to revert to the old versions, liquibase can be used safely. Also, liquibase provides database neutral way to perform create, update, and delete operations.

We can write the changes in xml files and can be applied to any database. Suppose if five instances of same application are running in five different data bases then to apply data base changes writing different SQL statements for different database is time consuming.

Liquibase provides xml tags and attributes to apply data base changes. Rather than writing SQL directly against the database to create, update or drop database objects, we can define database changes in XML/JSON/YAML files.

## 4.CORE CONCEPT OF LIQUIBASE

* **Changelog**: The file which has the list of database changeSets that needs to be applied is called a [changelog](https://www.liquibase.org/documentation/databasechangelog.html). These changelog files can be in either SQL, YAML, XML, or JSON format.
* **ChangeSet**: A [changeSet](https://www.liquibase.org/documentation/changeset.html) is a set of changes that need to be applied to a database. Liquibase tracks the execution of changes at a ChangeSet level.
* **Change**: A [change](https://www.liquibase.org/documentation/changes/index.html) describes a single change that needs to be applied to the database. Liquibase provides several change types like “create table” or “drop column” out of the box which are each an abstraction over a piece of SQL.
* **Preconditions**: [Preconditions](https://www.liquibase.org/documentation/preconditions.html) are used to control the execution of changelogs or changeSets. They are used to define the state of the database under which the changeSets or changes logs need to be executed.
* **Changelog Parameters**: Liquibase allows us to have [placeholders](https://www.liquibase.org/documentation/changelog_parameters.html) in changelogs, which it dynamically substitutes during runtime.

Liquibase creates the two tables databasechangelog and databasechangeloglock when it runs in a database for the first time. It uses the databasechangelog table to keep track of the status of the execution of changeSets and databasechangeloglock to prevent concurrent executions of Liquibase. Refer to the [docs](https://www.liquibase.org/get_started/how-lb-works.html) for more details.

### 5.SETTING UP LIQUIBASE IN SPRING BOOT

1.ADDING DEPENDENCY

By default Spring Boot auto-configures Liquibase when we add the [Liquibase dependency](https://search.maven.org/artifact/org.liquibase/liquibase-core) to our build file.

<dependency>

<groupId>org.liquibase</groupId>

<artifactId>liquibase-core</artifactId>

<version>4.28.0</version>

</dependency>

2.SETTING FILES IN CORRECT MANNER

**By default, Spring Boot runs Liquibase database migrations automatically on application startup.**

It looks for a master changelog file in the folder db/migration within the classpath with the name db.changelog-master.xml. If we want to use other Liquibase changelog formats or use different file naming convention, we can configure the spring.liquibase.change-log application property to point to a different master changelog file.

3.APPLICATION PROPERTIES FOR LIQUIBASE

ChangeLog file location spring.liquibase.

change-log=classpath:db/changelog/db.changelog/master.xml

This configurations help ensure that Liquibase operates correctly within the context of your Spring Boot application

## 6.EXAMPLE CODE

## Changelog file

<?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-4.9.xsd">  
  
 <changeSet id="b661266e-c090-4f14-93b6-1db4adbd4f42" author="shiny">  
 <createTable tableName="product">  
 <column name="id" type="bigint" autoIncrement="true">  
 <constraints primaryKey="true" nullable="false"/>  
 </column>  
 <column name="name" type="varchar(255)">  
 <constraints nullable="false"/>  
 </column>  
 <column name="price" type="decimal(10,2)">  
 <constraints nullable="false"/>  
 </column>  
 <column name="description" type="varchar(255)">  
 <constraints nullable="false"/>  
 </column>  
  
 </createTable>  
 </changeSet>  
</databaseChangeLog>

## Master.xml file

<?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-4.9.xsd">  
  
 <include file="db/changelog/202407091209\_created\_product\_entity.xml"/>  
 <include file="db/changelog/202407091633\_updated\_product\_entity.xml"/>  
 <include file="db/changelog/202407091444\_created\_customer\_entity.xml"/>  
<!-- <include file="db/changelog/202407091751\_foreignkeyadded\_customer\_entity.xml"/>-->  
<!-- <include file="db/changelog/202407091822\_rollbackcustomer\_customer\_entity.xml"/>-->  
  
  
</databaseChangeLog>

## 7.CONCLUSION

## Liquibase helps to automate database migrations, and Spring Boot makes it easier to use Liquibase. Using Liquibase with Spring Boot offers a robust solution for managing database changes in a controlled and efficient manner. It enables developers to focus on delivering features without worrying about the complexities of database migrations, making it an essential tool for any Spring Boot-based project