MyBatis 逆向工程

- 什么是逆向工程?

可以针对数据表自动生成 MyBatis 执行所需要的代码，包括：Mapper.java（***Repository），Mapper.xml，实体类。

- 为什么要使用 MyBatis 逆向工程?

MyBatis 框架是一个“半自动”的 ORM 框架，SQL 语句需要开发者自定义，这样做的好处是代码更加灵活，缺点是如果参与业务的表太多，每张表的业务都需要自定义 SQL，创建实体类，Repository 接口，工作量会很大，所以需要使用 MyBatis 逆向工程来自动生成需要的资源，减少开发者的工作量，让开发者可以将精力集中在业务逻辑上，而非是创建实体类，Repository 接口等工作。

- MyBatis 逆向工程的缺点?

逆向工程有其自身的局限性，逆向工程方法只能执行一次，如果再次执行就会重复生成 Repositpory 接口，实体类等资源，如果需要对表结构进行修改，那么就必须删除已经生成的所有资源，重新生成。

- 如果使用 MyBatis 逆向工程?

MyBatis Generator，简称 MBG：是一个专门为 MyBatis 框架定制的代码生成器，可以根据数据表结构快速生成对应的 Mapper.xml，Mapper 接口以及实体类，支持基本的 CRUD，复杂的 SQL 语句需要开发者手动编写。

- pom.xml

```
<dependencies>
    <!-- MySQL驱动 -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.11</version>
    </dependency>

    <!-- MyBatis -->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.4.5</version>
    </dependency>

    <!-- MBG -->
    <dependency>
        <groupId>org.mybatis.generator</groupId>
        <artifactId>mybatis-generator-core</artifactId>
        <version>1.3.2</version>
    </dependency>
</dependencies>
```

- 创建 MBG 配置文件 generatorConfig.xml
  - jdbcConnection：配置数据库连接信息。
  - javaModelGenerator：配置 JavaBean 的生成策略。
  - sqlMapGenerator：配置 SQL 映射文件的生成策略。
  - javaClientGenerator：配置 Mapper 接口的生成策略。
  - table：配置需要逆向解析的数据表（tableName：数据表名，domainObjectName：对应 JavaBean 的类名）

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration PUBLIC "-//mybatis.org//DTD MyBatis
Generator Configuration 1.0//EN" "http://mybatis.org/dtd/mybatis-
generator-config_1_0.dtd">

<generatorConfiguration>
    <context id="testTables" targetRuntime="MyBatis3">
        <!-- 配置数据库连接信息 -->
        <jdbcConnection
                driverClass="com.mysql.cj.jdbc.Driver"
                connectionURL="jdbc:mysql://localhost:3306/mybatis?
useUnicode=true&amp;characterEncoding=UTF-8"
                userId="root"
                password="root"
        ></jdbcConnection>

        <!-- 实体类 -->
        <javaModelGenerator targetPackage="com.southwind.entity"
targetProject="./src/main/java"></javaModelGenerator>

        <!-- Mapper.xml -->
        <sqlMapGenerator targetPackage="com.southwind.repository"
targetProject="./src/main/java"></sqlMapGenerator>

        <!-- Mapper.java -->
        <javaClientGenerator type="XMLMAPPER"
targetPackage="com.southwind.repository"
targetProject="./src/main/java"></javaClientGenerator>

        <!-- 数据表 -->
        <table tableName="t_user" domainObjectName="User"></table>
    </context>
</generatorConfiguration>
```

GeneratorMain

```
package com.southwind;

import org.mybatis.generator.api.MyBatisGenerator;
import org.mybatis.generator.config.Configuration;
```

```
import org.mybatis.generator.config.xml.ConfigurationParser;
import org.mybatis.generator.exception.InvalidConfigurationException;
import org.mybatis.generator.exception.XMLParserException;
import org.mybatis.generator.internal.DefaultShellCallback;

import java.io.File;
import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class GeneratorMain {
    public static void main(String[] args) {
        List<String> warings = new ArrayList<String>();
        boolean overwrite = true;
        String genCig = "/generatorConfig.xml";
        File configFile = new
File(GeneratorMain.class.getResource(genCig).getFile());
        ConfigurationParser cp = new ConfigurationParser(warings);
        Configuration config = null;
        try {
            config = cp.parseConfiguration(configFile);
        } catch (IOException e) {
            e.printStackTrace();
        } catch (XMLParserException e) {
            e.printStackTrace();
        }

        DefaultShellCallback callback = new
DefaultShellCallback(overwrite);
        MyBatisGenerator myBatisGenerator = null;
        try {
            myBatisGenerator = new
MyBatisGenerator(config,callback,warings);
        } catch (InvalidConfigurationException e) {
            e.printStackTrace();
        }

        try {
            myBatisGenerator.generate(null);
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

直接运行 GeneratorMain，即可自动生成相关的资源。