8、MBG-逆向工程

笔记本： spring
创建时间： 2022/4/4 17:26
作者： 雷丰阳

正向：
table----javaBean---BookDao---dao.xml---xxx
逆向工程：
根据数据表table，逆向分析数据表，自动生成javaBean---BookDao---dao.xml---xxx

MBG：MyBatis Generator：代码生成器；
MyBatis官方提供的代码生成器；帮我们逆向生成；

1、导包：mbg的核心包
2、编写mbg.xml配置文件

```
<?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>

    <!--
    MyBatis3Simple：基础班CRUD
    MyBatis3：复杂版CRUD
     -->
    <context id="DB2Tables" targetRuntime="MyBatis3">
        <commentGenerator>
            <property name="suppressAllComments" value="true"/>
        </commentGenerator>
        <!-- jdbcConnection:指导连接到哪个数据库 -->
        <jdbcConnection

            driverClass="com.mysql.jdbc.Driver"
            connectionURL="jdbc:mysql://localhost:3306/mybatis_0325"

            userId="root"

            password="123456">
        </jdbcConnection>

        <javaTypeResolver>
            <property name="forceBigDecimals" value="false" />
        </javaTypeResolver>

        <!-- javaModelGenerator：生成pojo

        targetPackage：生成的pojo放在哪个包
        targetProject：放在哪个工程下
        -->
        <javaModelGenerator targetPackage="com.atguigu.bean"
            targetProject=".\src">
            <property name="enableSubPackages" value="true" />
            <property name="trimStrings" value="true" />
        </javaModelGenerator>

        <!--sqlMapGenerator：sql映射文件生成器；指定xml生成的地方  -->
        <sqlMapGenerator targetPackage="com.atguigu.dao"
            targetProject=".\conf">
```

```
            <property name="enableSubPackages" value="true" />
        </sqlMapGenerator>

        <!-- javaClientGenerator：dao接口生成的地方 -->
        <javaClientGenerator type="XMLMAPPER"
            targetPackage="com.atguigu.dao"

            targetProject=".\src">
            <property name="enableSubPackages" value="true" />
        </javaClientGenerator>

        <!-- table：指定要逆向生成哪个数据表
        tableName="t_cat"：表名
        domainObjectName=""：这个表对应的对象名
         -->
        <table tableName="t_cat" domainObjectName="Cat"></table>
        <table tableName="t_employee" domainObjectName="Employee"></table>
        <table tableName="t_teacher" domainObjectName="Teacher"></table>

    </context>
</generatorConfiguration>
```

3、运行代码生成

```
package com.atguigu.test;

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

import org.mybatis.generator.api.MyBatisGenerator;
import org.mybatis.generator.config.Configuration;
import org.mybatis.generator.config.xml.ConfigurationParser;
import org.mybatis.generator.exception.XMLParserException;
import org.mybatis.generator.internal.DefaultShellCallback;

public class MBGTest {

    public static void main(String[] args) throws Exception {
        List<String> warnings = new ArrayList<String>();
        boolean overwrite = true;
        File configFile = new File("mbg.xml");
        ConfigurationParser cp = new ConfigurationParser(warnings);
        Configuration config = cp.parseConfiguration(configFile);
        DefaultShellCallback callback = new DefaultShellCallback(overwrite);
        MyBatisGenerator myBatisGenerator = new MyBatisGenerator(config,
                callback, warnings);
        //代码生成
        myBatisGenerator.generate(null);
        System.out.println("生成ok了！");
    }

}
```

4、测试复杂查询

```
package com.atguigu.test;

import static org.junit.Assert.*;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
```

```
import java.util.Map;
import java.util.UUID;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
import org.junit.Before;
import org.junit.Test;

import com.atguigu.bean.Teacher;
import com.atguigu.bean.TeacherExample;
import com.atguigu.bean.TeacherExample.Criteria;
import com.atguigu.dao.TeacherMapper;



public class MyBatisTest {

    // 工厂一个
    SqlSessionFactory sqlSessionFactory;


    @Test
    public void test02(){
        SqlSession sqlSession = sqlSessionFactory.openSession();
        //1、测试
        TeacherMapper mapper = sqlSession.getMapper(TeacherMapper.class);
        List<Teacher> teachers = new ArrayList<Teacher>();
        for (int i = 0; i < 1000; i++) {
            Teacher teacher = new Teacher();
            teacher.setTeachername(UUID.randomUUID().toString().substring(0,
5));
            teacher.setClassName(UUID.randomUUID().toString().substring(0,
5));
            teachers.add(teacher);
        }
        System.out.println("批量保存.....");
        mapper.insertBatch(teachers);
        sqlSession.commit();
        sqlSession.close();


    }

    /**
     * 测试代码生成器
     * @throws IOException
     */
    @Test
    public void test01(){
        SqlSession sqlSession = sqlSessionFactory.openSession();
        //1、测试
        TeacherMapper mapper = sqlSession.getMapper(TeacherMapper.class);
        //2、测试查询所有teacher
        List<Teacher> list = mapper.selectByExample(null);
        for (Teacher teacher : list) {
            System.out.println(teacher);
        }

        //3、带复杂条件的查询
        //select * from t_teacher id=? and teacherName like ?
        //封装查询条件的
        TeacherExample example = new TeacherExample();
        example.setOrderByClause("id DESC");
        //1、使用example创建一个Criteria（查询准则）
        Criteria criteria = example.createCriteria();
        criteria.andIdEqualTo(1);
        criteria.andTeachernameLike("%a%");

        System.out.println("======================");
        List<Teacher> list2 = mapper.selectByExample(example);
        for (Teacher teacher : list2) {
                System.out.println(teacher);
```

```
        }

        /**
         * 多个复杂条件
         * select * from t_teacher where  (id=? and teacherName like ?) or
(address like ? and birth bet)
         */
        TeacherExample example2 = new TeacherExample();


        //一个Criteria能封装一整个条件
        Criteria criteria2 = example2.createCriteria();
        criteria2.andIdGreaterThan(1);
        criteria2.andTeachernameLike("%a%");

        //创建第二个查询条件
        Criteria criteria3 = example2.createCriteria();
        criteria3.andAddressLike("%%");
        criteria3.andBirthDateBetween(new Date(), new Date());

        example2.or(criteria3);
        System.out.println("=======-=-=-=-=-=-=-");
        mapper.selectByExample(example2);

    }

    @Before
    public void initSqlSessionFactory() throws IOException {
        String resource = "mybatis-config.xml";
        InputStream inputStream = Resources.getResourceAsStream(resource);
        sqlSessionFactory = new
SqlSessionFactoryBuilder().build(inputStream);
    }

}
```