Mybatis动态sql

课程目标

1. 回顾使用Mybatis查询所有学生信息

理解Dao层的实现过程

2. 介绍Mybatis动态sql能力 重点

3. 实现学生信息的搜索功能

4. 实现年级的批量删除

回顾:

功能:使用Mybatis查询所有学生信息

理解Dao层的实现过程

1. 创建学生实体

|  |
| --- |
| **public class** Student {  *//推荐属性的数据类型使用类类型* **private** Integer **xh**;  **private** String **name**;  **private** String **sex**;  **private** Integer **age**;  **private** Date **birthday**;  **private** String **address**;  **private** Integer **state**;  **private** Integer **gid**;  setter和getter略  } |

1. 编写dao层接口

|  |
| --- |
| *//一个表一个实体一个接口一个sql映射文件* **public interface** StudentDao {  *//一个功能一个方法一个持久化操作* **public** List<Student> getAllStudent(); } |

1. 编写持久化操作

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* **<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd"*>*** <**mapper namespace="com.team.project.dao.StudentDao"**>  *<!--持久化操作-->* <**select id="getAllStudent" resultType="com.team.project.entity.Student"**>  select *\** from student  </**select**> </**mapper**> |

注意:编写了sql映射文件，一定要在mybatis配置文件中添加sql映射文件

1. 测试dao层

|  |
| --- |
| **public class** SelectStudent {  **public static void** main(String[] args) {  *//显示所有的学生信息* SqlSession sqlSession=MyBatisUtil.*getSession*();  StudentDao studentDao=sqlSession.getMapper(StudentDao.**class**);  *//调用方法* List<Student> list=studentDao.getAllStudent();  *//显示* System.***out***.println(**"学号\t姓名\t年龄\t性别\t生日"**);  list.forEach(s -> {  System.***out***.println(s.getXh()+**"\t"**+s.getName()+**"\t"**+s.getAge()+**"\t"**+s.getSex()+**"\t"**+s.getBirthday());  });  MyBatisUtil.*closeSession*();  } } |

Mybatis总结:

一个表一个实体一个Dao接口一个sql映射文件(一系列的持久化操作)

编写功能时:一个功能一个dao方法一个持久化操作.

1. Mybatis动态sql能力(拼接sql语句)

MyBatis 的一个强大的特性之一通常是它的动态 SQL 能力。 如果你有使用 JDBC 或其他 相似框架的经验,你就明白条件地串联 SQL 字符串在一起是多么的痛苦,确保不能忘了空 格或在列表的最后省略逗号。动态 SQL 可以彻底处理这种痛苦。

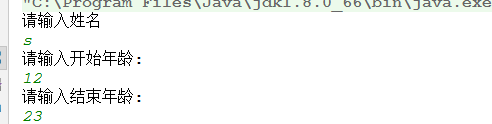
通常使用动态 SQL 不可能是独立的一部分,MyBatis 当然使用一种强大的动态 SQL 语 言来改进这种情形,这种语言可以被用在任意映射的 SQL 语句中。

动态 SQL 元素和使用 JSTL 或其他相似的基于 XML 的文本处理器相似。在 MyBatis 之 前的版本中,有很多的元素需要来了解。MyBatis 3 大大提升了它们,现在用不到原先一半 的元素就能工作了。MyBatis 采用功能强大的基于 OGNL 的表达式来消除其他元素。

* if
* choose (when, otherwise)
* trim (where, set)
* Foreach

三、实现学生信息的搜索功能

需求:



1. 创建条件实体类

作用:封装查询条件,一个搜索条件为一个属性

|  |
| --- |
| *//搜索条件的实体* **public class** StudentCondition {  *//一个查询条件就一个属性* **private** String **name**; *//姓名* **private** Integer **startAge**; *//开始年龄* **private** Integer **endAge**; *//结束年龄*  *Setter和getter方法略*  } |

1. 编写dao层
   1. 编写dao层接口的方法

|  |
| --- |
| *//通过条件查询学生* **public** List<Student> searchStudent(StudentCondition studentCondition); |

* 1. 编写持久化操作:根据条件动态生成sql

|  |
| --- |
| <**select id="searchStudent"  parameterType="com.kgc.entity.StudentCondition"**  **resultType="com.kgc.entity.Student"**> *<!--动态拼接sql-->* SELECT *\** FROM students where 1=1  <**if test="name!=null"**> and name like concat('%',#{name},'%')  </**if**>  <**if test="startAge!=null"**>  and age > #{startAge}  </**if**>  <**if test="endAge!=null"**>  and age **&lt;** #{endAge}  </**if**> </**select**>  <!--注意:不能出现特殊字符 如:<(&lt;) 🡪  <![CDATA[ 特殊字符 ]>  *<!-- SELECT \* FROM students  <where>  <if test="name!=null">  and name like concat('%',#{name},'%')  </if>  <if test="startAge!=null">  and age > #{startAge}  </if>  <if test="endAge!=null">  and age &lt; #{endAge}  </if>  </where>-->*  <select> |

1. 测试调用dao层

|  |
| --- |
| */\*实现搜索学生 测方式dao层\*/* **public static void** main(String[] args) **throws** Exception {  *//一、用户输入信息 jsp表单* Scanner input=**new** Scanner(System.***in***);  System.***out***.println(**"请输入姓名:"**);  String name=input.nextLine(); *//没有输入内容""* System.***out***.println(**"请输入开始年龄:"**);  String startAge=input.nextLine();  System.***out***.println(**"请输入结束年龄:"**);  String endAge=input.nextLine();   *//二、将用户输入的信息封装查询条件里 Servlet* StudentCondition studentCondition=**new** StudentCondition();  **if**(!name.equals(**""**)){  studentCondition.setName(name);  }  **if**(!startAge.equals(**""**)){  studentCondition.setStartAge(**new** Integer(startAge));  }  **if**(!endAge.equals(**""**)){  studentCondition.setEndAge(**new** Integer(endAge));  }    *///1.获得sqlSession对象* Reader reader=Resources.*getResourceAsReader*(**"mybatis-config.xml"**);  SqlSessionFactoryBuilder sqlSessionFactoryBuilder=**new** SqlSessionFactoryBuilder();  SqlSessionFactory sqlSessionFactory=sqlSessionFactoryBuilder.build(reader);  SqlSession sqlSession=sqlSessionFactory.openSession();  *//2.利用sqlSession生成接口的动态代理实现* StudentDao studentDao=sqlSession.getMapper(StudentDao.**class**);  *//3.调用dao层搜索的方法 传条件* List<Student> list=studentDao.searchStudent(studentCondition);  *//显示* System.***out***.println(**"学号\t姓名\t年龄\t性别\t生日"**);  list.forEach(s -> {  System.***out***.println(s.getXh()+**"\t"**+s.getName()+**"\t"**+s.getAge()+**"\t"**+s.getSex()+**"\t"**+s.getBirthday());  });  *//关闭sqlSession* sqlSession.close(); } |

四、实现批量(多项)删除功能：同时删除多个学生

1. 编写dao层

接口方法:

|  |
| --- |
| *//批量删除 传递list集合或者数组 //public int deleteMoreGrade(List<Integer> gids);* **public int** deleteMoreGrade(Integer [] gids); |

持久化操作:

|  |
| --- |
| *<!--删除多项的持久化操作  你可以传递一个 List 实例或者数组作为参数对象传给 MyBatis。  当你这么做的时 候,MyBatis 会自动将它包装在一个 Map 中,用名称在作为键。  List 实例将会以“list” 作为键,而数组实例将会以“array”作为键。 -->* <**delete id="deleteMoreGrade"**>  DELETE FROM grade WHERE gid IN  <**foreach collection="array" item="gid" open="(" close=")" separator=","**>  #{gid}   </**foreach**> </**delete**> |

2.测试 dao层

|  |
| --- |
| */\*删除多个年级\*/* **public static void** main(String[] args) **throws** Exception {  *//使用数组存放要删除的编号* Integer [] gids={45,46,47};  *//测试调用dao层删除多项  //1.获得sqlSession对象* Reader reader=Resources.*getResourceAsReader*(**"mybatis-config.xml"**);  SqlSessionFactoryBuilder sqlSessionFactoryBuilder=**new** SqlSessionFactoryBuilder();  SqlSessionFactory sqlSessionFactory=sqlSessionFactoryBuilder.build(reader);  SqlSession sqlSession=sqlSessionFactory.openSession();  *//2.利用sqlSession生成接口的动态代理实现* GradeDao gradeDao=sqlSession.getMapper(GradeDao.**class**);  *//调用方法* **int** temp=gradeDao.deleteMoreGrade(gids);  sqlSession.commit(); *//删除需要提交* sqlSession.close();   **if**(temp>0)  System.***out***.println(**"成功删除"**+temp+**"行"**);  **else** System.***out***.println(**"删除失败"**); } |