**作业名称：**课堂作业 2024.6.6

**文档命名：**课堂作业-2021214975-黑昀熙-2024.6.6

**作业内容：**

一、开发接口实现从分数表scores查询考试成绩最多的同学的科目名称，通过POSTMAN 展示最终结果，以下是具体内容:

1、构建分数实体类，内容结构参考



参考：

|  |
| --- |
| package com.example.entity;  /\*\*  \* SpringBootProj - Score  \* 分数实体类  \* @author HeiYunxi 黑昀熙  \* @version 1.0  \*/  public class Scores {  private int id;  private String course;  private String score;  private int studentid;  public Scores() {  }  public Scores(int id, String course, String score, int studentid) {  this.id = id;  this.course = course;  this.score = score;  this.studentid = studentid;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getCourse() {  return course;  }  public void setCourse(String course) {  this.course = course;  }  public String getScore() {  return score;  }  public void setScore(String score) {  this.score = score;  }  public int getStudentid() {  return studentid;  }  public void setStudentid(int studentid) {  this.studentid = studentid;  }  @Override  public String toString() {  return "Score{" +  "id=" + id +  ", course='" + course + '\'' +  ", score='" + score + '\'' +  ", studentid=" + studentid +  '}';  }  public String toString(User user) {  return "Score{" +  "user=" + user.getUsername() +  ", course='" + course + '\'' +  ", score='" + score +  '}';  }  } |

1. 映射文件，包含数据库查询语句

参考：

|  |
| --- |
| <?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.example.mapper.ScoresMapper">  <select id="ScoersInfo" resultType="com.example.entity.Scores">  select course from scores  </select>  <select id="moreCourse" resultType="String">  select course from scores  where studentid = (  SELECT studentid FROM scores  GROUP BY studentid  ORDER BY count(\*) DESC  LIMIT 1  )  </select>  </mapper> |

1. 构建数据库持久化层，模拟数据库查找操作（定义其查找数据的函数操作）

参考：

|  |
| --- |
| package com.example.mapper;  import com.example.entity.UserScores;  import org.springframework.stereotype.Component;  import java.util.List;  /\*\*  \* SpringBootProj - UserScores  \* @author HeiYunxi 黑昀熙  \* @version 1.0  \*/  @Component  public interface UserScoresMapper {  List<UserScores> SelectUserScoers();  } |

1. 构建业务服务层，完成数据查找的操作业务逻辑（定义其数据查找的函数操作，并完成内部逻辑）

参考：

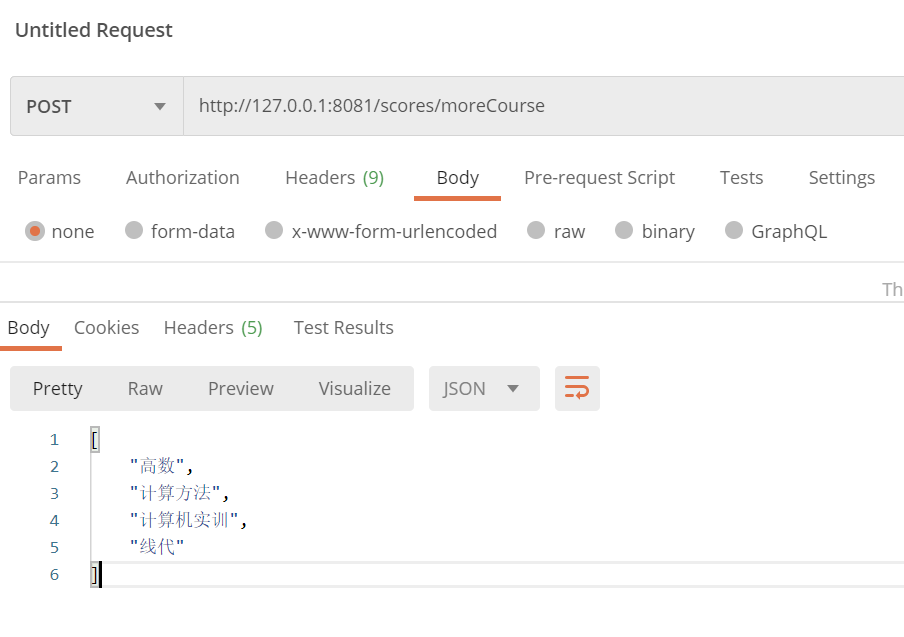
|  |
| --- |
| package com.example.service.impl;  import com.example.entity.Scores;  import com.example.mapper.ScoresMapper;  import com.example.service.IScoresService;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import org.springframework.transaction.annotation.Transactional;  import java.util.List;  /\*\*  \* SpringBootProj - ScoreServiceImpl  \* 业务服务层，完成数据的增删改查业务逻辑  \* @author HeiYunxi 黑昀熙  \* @version 1.0  \*/  @Service  @Transactional  public class ScoresServiceImpl implements IScoresService {  @Autowired  ScoresMapper scoresMapper;  @Override  public List<Scores> ScoresInfo(Scores scores) {  return scoresMapper.ScoersInfo(scores);  }  @Override  public List<String> moreCourse(){  return scoresMapper.moreCourse();  }  } |

1. 构建前端控制层，完成对前端请求的响应（定义其请求的响应函数，并完成响应逻辑）

参考：

|  |
| --- |
| package com.example.controller;  import com.example.entity.Scores;  import com.example.service.IScoresService;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.\*;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  /\*\*  \* SpringBootProj - ScoreController  \* @author HeiYunxi 黑昀熙  \* @version 1.0  \*/  @Controller  @RequestMapping("scores")  public class ScoresController {  @Autowired  IScoresService iScoresService;  /\*\*  \* @param map  \* @return dataMap  \*/  @RequestMapping(value = "/ScoresInfo")  @ResponseBody  public Map<String, Object> ScoresInfo(@RequestBody Map<String, String> map) {  if (null != map && map.containsKey("course")) {  String score = map.get("course").toString();  Scores scores = new Scores();  scores.setScore(score);  List<Scores> scoreslist = iScoresService.ScoresInfo(scores);  Map<String, Object> dataMap = new HashMap<>();  dataMap.put("Scores data", scoreslist);  return dataMap;  }  return null;  }  /\*\*  \* @return courselist - 课程列表：考试成绩最多的同学的科目名称  \*/  @RequestMapping(value = "/moreCourse")  @ResponseBody  public List<String> moreCourse() {  List<String> courselist = iScoresService.moreCourse();  return courselist;  }  } |

6、POSTMAN 展示最终结果截图



二、开发用户表users的插入接口，数据库查询记录数增加为准（POSTMAN发送请求）

1、构建分数实体类，内容结构参考



参考：

|  |
| --- |
| package com.example.entity;  import java.util.Date;  /\*\*  \* User  \* 用户实体类  \* @author HeiYunxi 黑昀熙  \* @version 1.0  \*/  public class User {  private int id;  private String username;  private Date birthday;  private String sex;  private String address;  public User() {  }  public User(int id, String username, Date birthday, String sex, String address) {  this.id = id;  this.username = username;  this.birthday = birthday;  this.sex = sex;  this.address = address;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public Date getBirthday() {  return birthday;  }  public void setBirthday(Date birthday) {  this.birthday = birthday;  }  public String getSex() {  return sex;  }  public void setSex(String sex) {  this.sex = sex;  }  public String getAddress() {  return address;  }  public void setAddress(String address) {  this.address = address;  }  @Override  public String toString() {  return "User{" +  "id=" + id +  ", username='" + username + '\'' +  ", birthday=" + birthday +  ", sex='" + sex + '\'' +  ", address='" + address + '\'' +  '}';  }  } |

1. 映射文件，包含数据库查询语句

参考：

|  |
| --- |
| <insert id="insertUser" parameterType="com.example.entity.User">  insert into users(username, birthday, sex, address) values(#{username}, #{birthday}, #{sex}, #{address});  </insert> |

1. 构建数据库持久化层，模拟数据库数据增加的操作（定义其数据增加的函数操作）

参考：

|  |
| --- |
| package com.example.mapper;  import com.example.entity.User;  import org.apache.ibatis.annotations.Param;  import org.springframework.stereotype.Component;  import java.util.Date;  import java.util.List;  /\*\*  \* SpringBootProj - UserMapper  \* 数据库持久化层，模拟数据库增删改查操作  \* @author HeiYunxi 黑昀熙  \* @version 1.0  \*/  @Component  public interface UserMapper {  //插入  void insertUser(@Param("username")String name, @Param("birthday")Date birthday, @Param("sex")String sex, @Param("address")String address);  } |

1. 构建业务服务层，完成数据增加的操作业务逻辑（定义其数据插入操作的函数操作，并完成内部逻辑）

参考：

|  |
| --- |
| @Service  @Transactional  public class UserServiceImpl implements IUserService {  @Autowired  UserMapper userMapper;  @Override  public void insertUser(String name, Date birthday, String sex, String address) {  userMapper.insertUser(name, birthday, sex, address);  }  } |

1. 构建前端控制层，完成对前端请求的响应（定义其请求的响应函数，并完成响应逻辑）

参考：

|  |
| --- |
| /\*\*  \* 插入用户信息  \* @param  \*/  @RequestMapping("/insert")  @ResponseBody  public boolean insert(@RequestBody Map<String, String> map) throws ParseException {  String name=map.get("username")==null?"":map.get("username").toString();  DateFormat df = new SimpleDateFormat("yyyy-MM-dd");  Date birthday =map.get("birthday").toString()==null?null:df.parse(map.get("birthday").toString());  String sex=map.get("sex").toString()==null?"":map.get("sex").toString();  String address=map.get("address").toString()==null?"":map.get("address").toString()  iUserService.insertUser(name,birthday,sex,address);  return true;  } |

6、POSTMAN 及数据库展示最终结果截图

