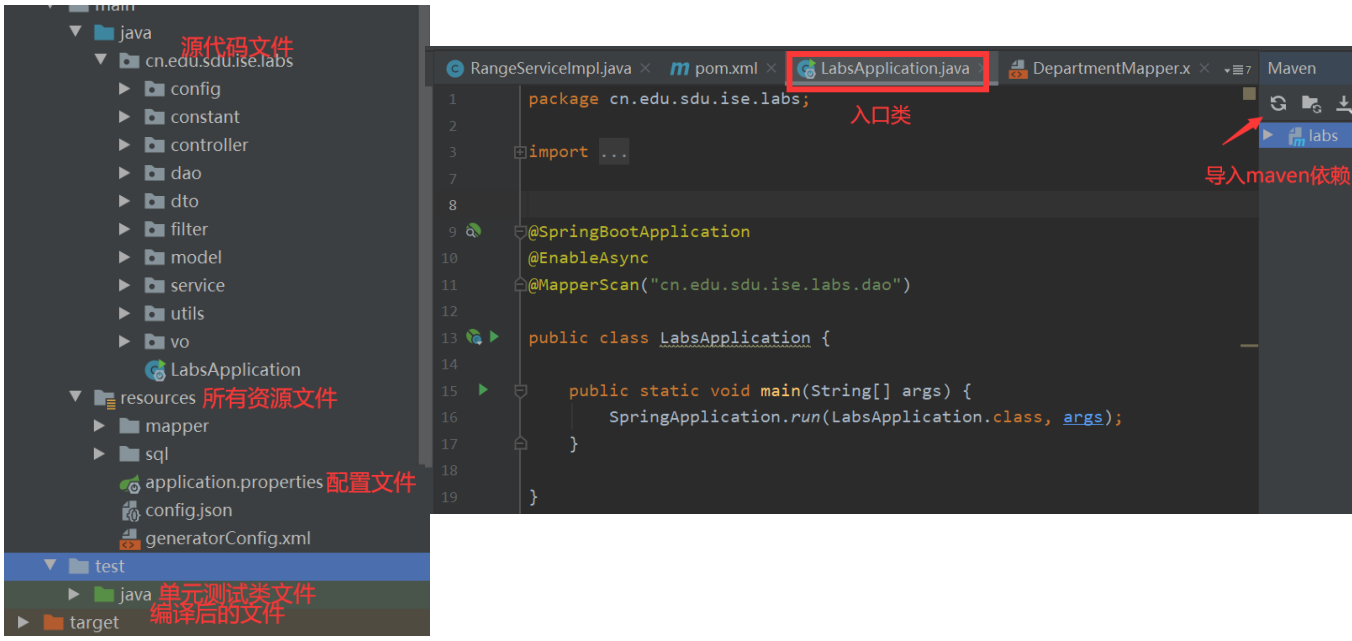
1 前期学习

1.1. Spring Boot 开发环境



基本结构:

1.controllor 层

controllor 属于控制层，多数用来实现和前端请求的对接，并判断，前端发出的请求，传入的参数是否符合条件。controllor 中的方法将会通过调用 service 接口来实现。

2.service 层

service 层为服务层，一些需要完成的动作，操作需要在 service 层完成

service 层分为 service 接口类和 serviceImpl 接口实现类

1）service 接口类，用来完成 controllor 方法的请求

2）serviceImpl 接口实现类，通过 implements service 接口类 实现接口方法。同时会调用 mapper 层的方法。

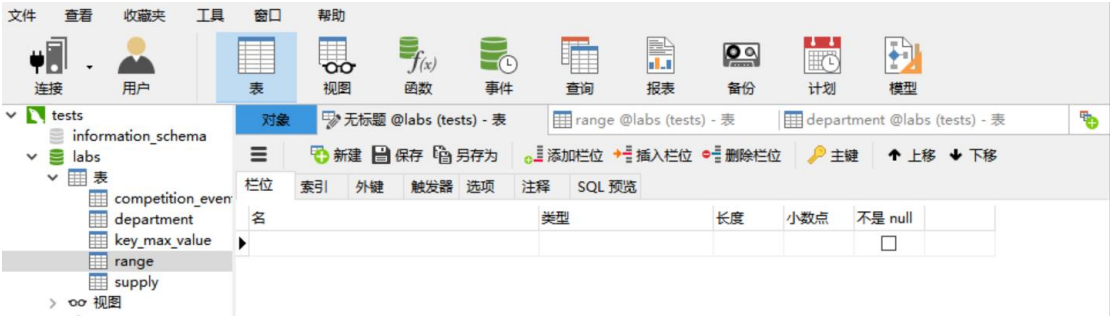
3.mapper 层

数据层，进行数据操作，完成 sql 语句

1.2. MySQL 数据库以及基本 SQL 语句的使用

1. 利用 SQL 面向数据库执行查询
2. 利用 SQL 从数据库取回数据
3. 利用 SQL 在数据库中插入新的记录
4. 利用 SQL 更新数据库中的数据
5. 利用 SQL 从数据库删除记录

1.3. Navicat 创建数据库表.添加数据等



1.4. Spring data JPA 使用

作用: 可以省略实现持久成业务逻辑的工作,只需要声明持久层接口.

1.服务层接口类 cn/edu/sdu/ise/labs/service/RangeService.java

接口类定义接口来实现文档中所要求的增删改查的目的.

2.服务层实现类 cn/edu/sdu/ise/labs/service/impl/RangeServiceImpl.java

实现类中注解:

- @Service -- service 标注业务层组件,这个注解是写在类上面的，标注将这个类交给 Spring 容器管理
- @autowired --这个注解是用来修饰变量的，写在变量上面，并且由系统底层代理创建这个变量的实例，并注入到这个类中，就不用自己手动去创建对象了
- @Override --验证@Override 下面的方法名是否是你父类中所有的，如果没有则报错。

1.5. MyBatis 配置与使用

配置

向 application.properties 添加 mybatis 配置

mybatis.mapperLocations=classpath:mapper/*.xml

mybatis.typeAliasesPackage=cn.edu.sdu.ise.labs.model

DAO 层接口

目标文件:cn/edu/sdu/ise/labs/dao/RangeMapper.java

@Param:注解参数,在 Mapper.xml 中可以采用#{ }的方式对注解括号内的参数进行引用.

Mapper.xml 文件

MyBatis 中在查询进行 select 映射的时候，返回类型可以用 resultType，也可以用 resultMap，resultType 是直接表示返回类型的，而 resultMap 则是对外部 ResultMap 的引用，但是 resultType 跟 resultMap 不能同时存在。

<mapper> : 该标签的 namespace 属性用于绑定 DAO 接口, namespace 中的包名要和 Dao/mapper 接口的包名一致!

<resultMap> : 是 Maybatis 的结果集封装，搭配<select><association>等标签的 resultMap 属性使用,管理结果与实体类之间的映射关系.

*<id>*和 *<result>*都映射一个单独列的值到简单数据类型

属性:

- Property : 映射到列结果的字段或属性。
- column 从数据库中得到的列名，

< select >: 选择，查询语句;

属性:

- - id : 就是对应的 namespace 中的方法名;
- - resultType：Sql 语句执行的返回值!

- - resultMap：它是映射器的引用
- - parameterType： 参数类型

< insert > – 映射插入语句

< update > – 映射更新语句

< delete > – 映射删除语句

Model 中的实体类(lombok 插件)

目录:cn/edu/sdu/ise/labs/model/Range.java

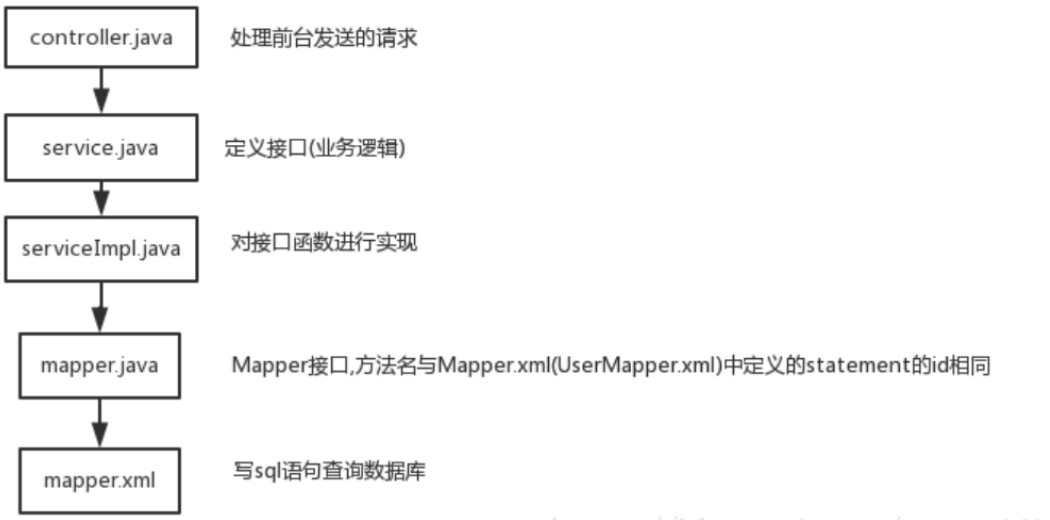
使用步骤:

1. 在 IDEA 中安装 Lombok 插件

2. 在项目中导入 lombok 的 jar 包

@Data：无参构造，get、set、tostring、hashcode，equals

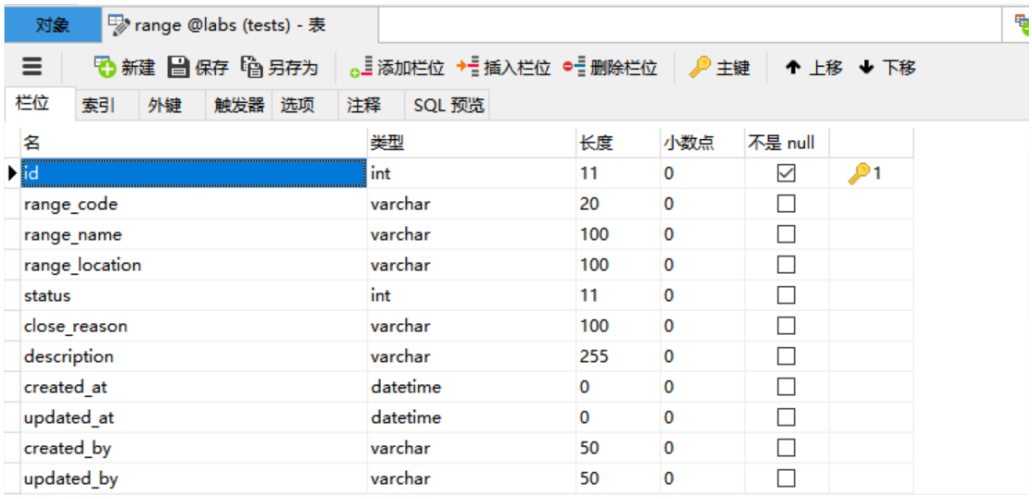
2 开发工作



相关功能的调用和代码解释在文件中进行注释,为避免报告冗长,仅以实现添加数据为例

1. 数据库表的创建

根据设计文档要求，在 Navicat 中新建 range 和 competition_event 表，相关参数如图所示:



名	类型	长度	小数点	不是 null	
id	int	11	0	☑	1
range_code	varchar	20	0	☐	
range_name	varchar	100	0	☐	
range_location	varchar	100	0	☐	
status	int	11	0	☐	
close_reason	varchar	100	0	☐	
description	varchar	255	0	☐	
created_at	datetime	0	0	☐	
updated_at	datetime	0	0	☐	
created_by	varchar	50	0	☐	
updated_by	varchar	50	0	☐	

名	类型	长度	小数点	不是 null	
id	int	11	0	☑	1
competition_event_code	varchar	20	0	☐	
competition_event_name	varchar	20	0	☐	
suite_type	int	10	0	☐	
range_code	varchar	20	0	☐	
plan_start_at	date	0	0	☐	
plan_end_at	date	0	0	☐	
status	int	10	0	☐	
created_at	datetime	0	0	☐	
updated_at	datetime	0	0	☐	
created_by	varchar	50	0	☐	

2. Mybatis-Generator 生成 Model,Dao,Mapper 代码

根据 generatorConfig.xml 配置文件中的相关配置,自动生成 Model,Dao,Mapper 代码

Model 中使用 lombok

在 idea 中安装 lombok 插件

将 model 中的相关函数删除,添加@Data 注解.

3. Controller 层设计

@RestController : @Controller+@ResponseBody 两个注解的结合，返回 json 数据不需要在方法前面加@ResponseBody 注解, 表示 RangeController 的实例是一个控制器

@RequestMapping 注解语句，作用在处理方法上。注解的将请求 URI 映射到方法

@GetMapping 用于处理请求方法的 GET 类型

@PostMapping，处理 post 请求

@RequestBody 主要用来接收前端传递给后端的 json 字符串中的数据的(请求体中的数据的)；GET 方式无请求体，所以使用@RequestBody 接收数据时，前端不能使用 GET 方式提交数据，而是用 POST 方式进行提交。

在 controller 中设计了五个接口,用于实现文档中要求的增删改查获取详细等功能，

自动装载 rangeService，在不同的接口中分别调用业务层中的相关方法

```
@PostMapping("add")
public ResultContext add(@RequestBody RangeDTO rangeDTO) {
    return
ResultContext.returnSuccess(rangeService.addRange(rangeDTO));
}
```

4. Service 层接口

在这层中,设计了五个功能的接口,” addRange”,” updateRange”,” listRange”,” deleteRange”,” getRange”,并在实现类中,编写相关代码将其实现.

```
/**
 * 添加场地
 * @param rangeDTO 场地对象
 * @return 场地编码
 */
String addRange(RangeDTO rangeDTO);
```

5. Service 层实现类

存放业务逻辑处理，也是一些关于数据库处理的操作，但不是直接和数据库打交道，通过导入 mapper 层， service 是供我们使用的方法。

在编写代码时导入了，“rangeMapper”和” competitionEventMapper”两个接口,用于实现相关功能.

```
@Override
public String addRange(RangeDTO rangeDTO) {
    // 校验输入数据正确性
  RangeUtils.validateRange(rangeDTO);
    // 创建实体对象，用以保存到数据库
  Range range = new Range();
    // 将输入的字段全部复制到实体对象中
  BeanUtils.copyProperties(rangeDTO,range);
    // 生成业务代码

range.setRangeCode(keyMaxValueService.generateBusinessCode(PrefixConsta
nt.RANGE));
    // 将 token 相关信息填入 range 对象
  TokenContextHolder.formatInsert(range);
    //设置创建和更新时间
  range.setCreatedAt(new Date());
    range.setUpdatedAt(range.getCreatedAt());
    //检测是否重名
  Integer number = rangeMapper.countByRangeName(range.getRangeName());
    //重名则提供警告
  Assert.isTrue(number == 0,"警告:场地名称已经存在");
    //调用 mapper 中的 insert 方法将其插入
  rangeMapper.insert(range);
    return range.getRangeCode();
}
```

6. Dao 层设计

作为 mapper 的接口

```
/**
 * 新建场地
 * @param record 场地参数
 * @return
 */
int insert(Range record);
```

mapper 层

数据层，进行数据操作，完成 sql 语句, 对数据库进行数据持久化操作，他的方法语句是直接针对数据库操作的。

```
<insert id="insert" parameterType="cn.edu.sdu.ise.labs.model.Range" >
  insert into `range` (id,range_code, range_name,range_location,
status, close_reason,description,created_at,updated_at,created_by,
updated_by)
  values (#{id,jdbcType=INTEGER}, #{rangeCode,jdbcType=VARCHAR},
#{rangeName,jdbcType=VARCHAR},
    #{rangeLocation,jdbcType=VARCHAR}, #{status,jdbcType=INTEGER},
#{closeReason,jdbcType=VARCHAR},
    #{description,jdbcType=VARCHAR}, #{createdAt,jdbcType=TIMESTAMP},
#{updatedAt,jdbcType=TIMESTAMP},
    #{createdBy,jdbcType=VARCHAR}, #{updatedBy,jdbcType=VARCHAR})
</insert>
```

7. Model 层。

存放实体类，与数据库中的属性值基本保持一致。

8. DTO（Data Transfer Object）和 VO（View Object）

DTO 代表服务层需要接收的数据,或者说是前端返回的数据，而 VO 代表返回到前端的数据.

3. 文档详细描述的实现

添加场地时,场地名称存在时警告

接口描述：
1、校验场地名称的唯一性，若存在报错“场地名称已经存在”

```
//检测是否重名
Integer number = rangeMapper.countByRangeName(range.getRangeName());
//重名则提供警告
Assert.isTrue( expression: number == 0, message: "警告:场地名称已经存在");
```

通过 countByRangeName 方法到数据库查询有无相同名称的场地,若存在则提示警告,若不存在则添加场地.

修改场地时,查询场地编码是否存在,和场地名称是否唯一

接口描述：↵
1、 根据场地编码查询场地记录，若不存在则报错↵
2、 校验场地名称的唯一性，若存在报错“场地名称已经存在”↵

```
//从数据库中获取需要更新的实体类
Range range = rangeMapper.getByCode(rangeDTO.getRangeCode());
//警告
Assert.notNull(range, message: "未找到场地，场地编码为：" + rangeDTO.getRangeCode(
//如果修改的名称和指向的名称不同则要检查是否重名,如果名称没改则不需要检查
if (!rangeDTO.getRangeName().equals(range.getRangeName()))
{//检测是否重名
    Integer number = rangeMapper.countByRangeName(rangeDTO.getRangeName());
    //重名则提供警告
    Assert.isTrue( expression: number == 0, message: "警告:场地名称已经存在");}
```

1. 通过 getByCode 方法查询,场地编码是否存在
2. 先用 if 判断是否修改名称,若没有修改名称则不需要查询,如果修改名称,则要查询是否重名

问题 1:

这里判断用的表达式不能用

rangeDTO.getRangeName() != range.getRangeName()

原因: java 中字符串的比较是==比较引用，equals 比较值

解决方法:

rangeDTO.getRangeName().equals(range.getRangeName()

删除场地,查询是否被占用

接口描述：↵
1、 根据场地编码查询 competition_event 表中 status!=3 的记录，如果有记录，则报错“该场地已经被比赛使用，不能删除”↵

```java
//将codeString循环删除
for(String rangecode : rangeCodes) {
    //查询该场地是否在competition中使用,如果正在使用不能删除
    Integer number = competitionEventMapper.countByRangeCode(rangecode);
    Assert.isTrue( expression: number == 0, message: "警告:该场地已经被比赛使用，不能删除");
    rangeMapper.deleteByCode(rangecode);
}
```

```xml
<select id="countByRangeCode" resultType="integer">
  select count(*)
  from `competition_event`
  <where>
    range_code = #{rangeCode}
    and status != 3
  </where>
```

通过 for 循环遍历,rangeCodes 列表,对每一个 code 都查询是否在 CompetitionEvent 表中存在,且状态不等于 3,如果有存在那么提示警告

查询场地时,使用 mybatis 动态 SQL

```xml
<select id="list" resultMap="BaseResultMap">
  select
  <include refid="Base_Column_List"/>
  from `range`
  <where>
  <if test="queryDTO.rangeName != null">
    AND range_name like #{queryDTO.rangeName}
  </if>
  <if test="queryDTO.rangeLocation != null">
    AND range_location like #{queryDTO.rangeLocation}
  </if>
  <if test="queryDTO.status != null">
    AND status = #{queryDTO.status}
  </if>
  </where>
  limit #{offset}, #{limit}
</select>
```

这条语句提供了一个可选的文本查找类型的功能。如果没有传入"title"，那么所有结果都会返回；反之若传入了"title"，那么就会把模糊查找"title"内容的结果返回

Status 没有使用模糊查询,直接查询相等的结果.

VO 的 statusDesc 实现

statusDesc	状态描述	String	

```
public static RangeVO convertToVO(Range range) {
    RangeVO rangeVO = new RangeVO();
    BeanUtils.copyProperties(range, rangeVO);
    //通过 swich 语句给statusDesc赋值
    switch (rangeVO.getStatus()){
        case 1:
            rangeVO.setStatusDesc("未开始");
            break;
        case 2:
            rangeVO.setStatusDesc("进行中");
            break;
        default:
            rangeVO.setStatusDesc("已结束");
            break;
    }
    rangeVO.setCreatedAt(FormatUtils.formatFullDate(range.getCreatedAt
    rangeVO.setUpdatedAt(FormatUtils.formatFullDate(range.getUpdatedAt
    //将VO对象返回
    return rangeVO;
```

问题 2:

通过修改 cn/edu/sdu/ise/labs/service/utils/RangeUtils.java 中的 convertToVO 方法来实现,每次讲 range 转为 VO 时为其赋值

在 vo 对象中添加 statusDesc 属性,并根据 status 的值通过 switch 语句对其赋值

Mapper.xml 的问题

问题 3

在 swagger 测试时会报错:

{"code":1000,"message":"\r\n### Error updating database. Cause: com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'range (id, range_code, range_name, \n range_location, status, close_reason, ' at line 1\r\n### The error may involve cn.edu.sdu.ise.labs.dao.RangeMapper.insert-Inline\r\n### The error occurred while setting parameters\r\n### SQL: insert into range (id, range_code, range_name, range_location, status, close_reason, description, created_at, updated_at, created_by, updated_by) values

解决方法: 将 range 用` `包裹

```
from `range`
```