**智慧社区项目组总结**

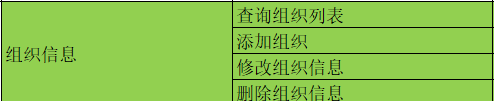
**——**2021/12 — 2022/06

1. **Eclipse**
2. **业务逻辑**

完成涉及的需求有很多，不可能一一的都拿出来讲，选了两个最具有代表性的需求来讲，目的不是为了讲需求的实现，而是讲这类似的需求的一个逻辑思维。

**1.1、组织服务**

这一需求首先要先确立一点，就是这个功能是一对多，还是一对一的关系。需不需要涉及关联表？需不需要考虑权限问题。侧重点是什么？侧重点是部门，也就是说的组织，部门就是部门，在它管辖的范围里会有一个组织信息的大菜单，它就是最大的，没有父级菜单，在这个菜单下会有一些小的功能，所以针对这个功能模块属于是一对一的关系，无需建立关联表。



也就是说我们所要实现的只是对部门的一个简单的增删改查的操作，但是我们是小区管理系统，会有员工，会有物业的员工，查询部门信息不可能只查一个部门名字就可以了，这一定会有相应员工的名字。

那么我们建表首先部门肯定有个部门id，然后我们从物业公司的角度出发他在查询的时候肯定会是在自己所在的小区查询，不能出现小区1的工作人员去查询到了小区2的部门信息，这是相当于盗窃机密，所以在对这个服务进行的任何操作都一定要传一个小区id，必须在知道你是哪一个小区的前提下咱们再慢慢说其他的，在这里不用考虑业主或者是员工的权限问题，因为在使用账号登录的时候拦截器已经校验了权限归属问题，部门id有了，小区id有了，那么该部门下一定得有员工的名字，查部门查部门肯定是查某个员工是哪个部门的啊。其次我们的部门会有一个上下级关系，就像一个公司他有总公司、分公司这样的一个上下级关系，我们称之为部门树，一个树状结构，所以还会有一个父级id。我们在页面上看到这些信息的时候是不是还会有一个顺序，就像是一个编号，1：部门.....2：部门......这样的，所以还要有一个编号。最后再加上一个数据状态，这样一个部门信息表就建好了。

**1.2、车场车位管理**

停车场、停车位的管理必然是连在一起的。

车场{parkLots\_id,num,name,community\_id,......}

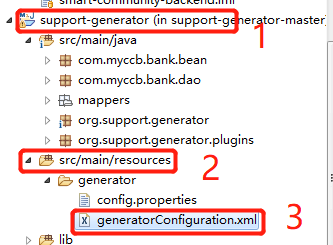
车位{ps\_id,num,community\_id,......}

在我们对这两张表进行操作的时候，首先是增添停车位的问题，联系到现实生活，虽然我们设置的是两张不同的表，但是在作车位增添的时候我们需要考虑需要添加的车场所属的车场num是否存在，如果连车场都不存在还怎么添加车位，所以在这之前我们一定要拿着传来的parkLots\_id去车场表里查询一遍看看有没有该条数据，有则可以添加，没有则抛出异常。到这里我们才做了第一点，其二我们要考虑添加的这个num是否已经存在，因为我们不能让同一小区下的同一车场里有相同num的车位，这肯定是不行的，所以同样的我们需要拿着传来的num在我们的车位表里联合小区id和车场id先去查一遍看是否存在，不存在才能进行添加否则抛出异常。在这里我们不能去对num设置唯一约束，因为我们要保证的是同一小区下的同一车场的num不能重复，但是我们所有的车位信息都是在一张表里的，会出现同一小区的不同车场会有相同num或者不同小区的不同车场的num也是可以重复的，如果设置约束的话它只要查询到有相同num都会抛异常的，这样是不行的，最好是在Service里面使用代码逻辑来实现。同样的，车位表里的会有一个车场的id，我们为了维护和区分这两张表，在车位表里只会显示一个车场id而不是直接显示车场的名字，想要有名字理论上还得在拿着我们的这个车场id去车场表里查询一次才能有结果，但是这样一来的话如果我们后端不做处理，前端拿到这个id再去查会很麻烦的，最好的方法时我们后端设置一个方法去让查询车位这一操作的时候顺便也把车场给查了，直接返回一个车场的name，这里就不用在service里使用代码逻辑，只需要在xml里面对查询方法做一个左连接查询的sql语句就可以了。在不增加代码量和改变表结构的情况下，也为前端减轻了一定的工作量。

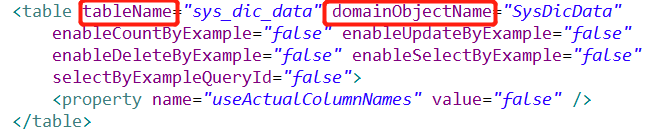
1. **代码分析**

**2.1、框架的基本使用方法**

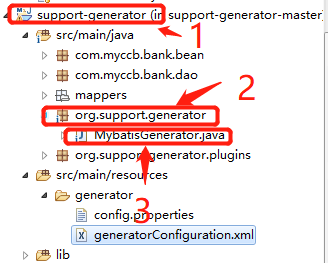
这里不讲配置文件原理，只讲在实际开发中的一个大体流程，取消了controller层、serviceImpl层的代码编写，重新定义了@Service注解的用法，保留bean,dao,mapper,service。加入了代码生成器support-generator，在连接我们的Mysql的同时只需要：



在generatorConfiguration.xml中的：



替换成我们想要生成的数据库表的表名，再运行MybatisGenerator.java后刷新项目



会得到相应的bean、dao、mapper的java类。将其复制粘贴至我们社区项目的对应目录下即可，那么剩下的就只有service的编写，注意每一个服务都需要使用框架封装好的注释@Service+自定义服务，例如新增车场@Service(“addParkLots”)，一个服务号代表一个服务的接口，同一个Service下可以有多个服务接口，每一个接口方法根据实际情况编写。在这里因为我们用代码生成器自动生成的有\*mapper.xml，在这里最基本的有关增删改查的sql语句已经直接写好，所以不用考虑sql。

**2.2、查询和分页查询**

查询使用list来查集合，自定义一个分页查询类RspPage<>，自定义在使用分页查询的时候要先通过AppMidRequestHeader 的header拿到pageNo和pageSize，也就是说在传参数的时候一定要有这个：



在知道页数和条数的情况下再执行：



PageMethod是一个定义的基础分页方法，只用最基础的sqlSession方法查询想要查询的类表，将结果放在一个List集合里，最后将这个List集合打包强转为Page<>的集合，返回的东西就是这个Page<>集合的内容，分页查询的效果就达到了。

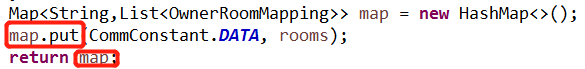
分页，和不分页的区别用法，在于对业务量的需求，查询结果很多时使用分页查询来美化界面让人一目了然，数据量很少时也可以使用分页查询，这时的分页与否相差不大。

那是效果。不使用分页查询的时候我们不能够直接返回查询的结果集，那样的话对于前端处理查询结果的时候会很麻烦，这时候最好的处理是将其放在Map集合里，返回一个map。

直接在方法头上将List集合放在Map里面



new出基本对象，将查询的list结果集put进这个map中，最后返回map



每一种的查询方法在返回的时候都要考虑到接下来的处理是否便捷，是否能够直观的处理，返回一个String状态码也好，集合也好，都是遵循这一个原则。

**2.3、同时添加数据到两张表中**

这里有一个业务需求，我们在添加员工信息的时候，该员工信息在我们的员工表中同时该员工也是我们的用户，也应当同时作为用户添加到我们的用户表中。

一个简单的逻辑思路，在添加员工这个方法类中传员工实体类，在方法中new一个用户对象，将我们要添加的员工信息中同时在用户表中有的字段值给set进用户new出来的对象中，在分别对员工实体和用户实体进行插入到表操作。

**2.4、添加数据到表时校验数据是否重复**

**项目需求：**

现有一张==u\_ps==的车位信息表，对应每个小区的车位id ==ps\_id==自增长，每个车位又对应==车位编号num==，车位id由自增长，车位编号可以根据自己小区情况手动添加设置，但是不能重复，车位编号不能重复。这就需要在新增车位信息的时候代码检验新增的该==num==在==u\_ps==表中是否存在，存在则抛出异常，不存在则做插入操作。

**问题关键：**

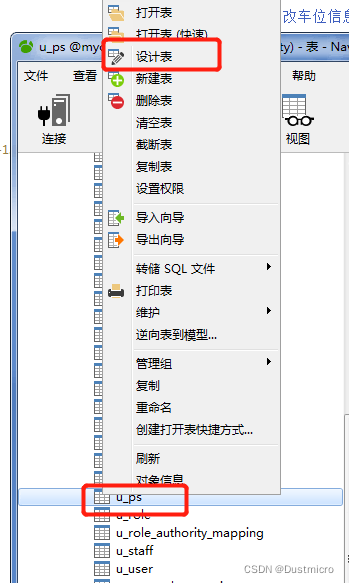
不同的小区==community\_id==中车位==num==可以重复，如果分开设计表设置==唯一约束==是可行的，但是我们的需求是将所有的小区车

位信息放在同一个表中，这就涉及到在插入一条数据的时候需要判断是否是同一小区id，在==community\_id==相同的情况下==num==不能重复。

**方法一：表分开时，对字段设置约束**

我们的主键为车位id ==ps\_id==，我们的思路是\*\*当插入一条数据，先查询车位信息表，遍历新的数据中的num是否存在，或者说新增的num在表中是否为空，是则抛出异常，否则执行插入操作。\*\* 这是代码逻辑。

在sql中最直接的方法就是设置唯一索引（这跟主键并不冲突）

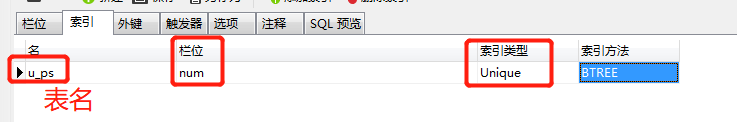


**选中num**

IMG_256

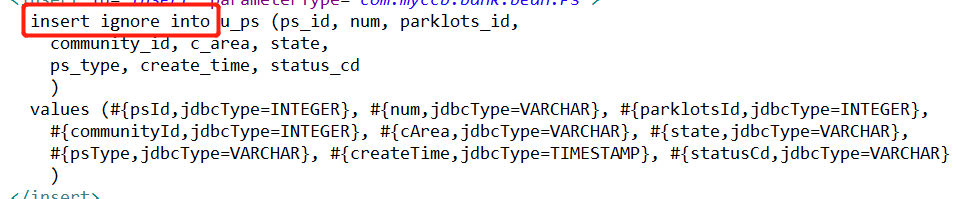
**点击索引**

IMG_256



这样就将num字段设置为唯一索引。

**在项目xml文件中设置插入语句**

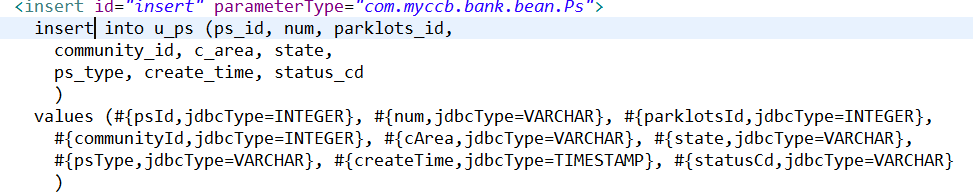


**\*\*insert ignore into\*\*** 语句是针对主键（唯一索引）字段的情况下做判断是否有重复的字段值，如果有则忽略本次操作抛出异常（**\*\*try{}catch(){}\*\***）,没有则执行插入操作。

**方法二：所有操作在同一张表中**

**我们在不修改sql的情况下利用代码逻辑来实现**

**sql语句是最基本的插入语句：**



**代码逻辑：**



**解析代码：**

new一个新的Ps（车位信息类）bean对象，将我们插入的数据中的==community\_id和num==放进listps对象中，对==listps== 执行sqlSession查询操作并放进list集合中，再对该查询结果集中的两个字段进行判空，如果为空说明该表中没有该字段的值，则进行插入操作，如果不为空则抛出异常并且在catch中再次抛出异常==ServiceException e==，这里设置两次catch，==ServiceException e==是为了返回给前端我们==车位编号重复的异常==，如果不设置的话前端只会收到==异常信息==。

StringUtilMycc为我们设置的工具类，该类下的==isEmpty==方法为判空操作。

**\*\*注意：判空时还需要保证传递的该字段的值不能为空，如果只判断表中数据是否为空的话最后的测试结果指挥报空指针！！！\*\***

**2.5批量插入**

**2.6单元测试案例写法**

**2.7解决常见问题**

**2.7.1 token刷新问题**

**（一）首先了解一下request和session的区别**

request 指在一次请求的全过程中有效，即从http请求到服务器处理结束，返回响应的整个过程，存放在HttpServletRequest对象中。在这个过程中可以使用forward方式跳转多个jsp。在这些页面里你都可以使用这个变量。request是用户请求访问的当前组件，以及和当前web组件共享同一用户请求的web组件。如：被请求的jsp页面和该页面用<include>指令包含的页面以及<forward>标记包含的其它jsp页面；

Session是用户全局变量，在整个会话期间都有效。只要页面不关闭就一直有效（或者直到用户一直未活动导致会话过期，默认session过期时间为30分钟，或调用HttpSession的invalidate()方法）。存放在HttpSession对象中 ，同一个http会话中的web组件共享它。

1. **token主要有两个作用**

①防止表单重复提交(防止表单重复提交一般还是使用前后端都限制的方式，

比如：在前端点击提交之后，将按钮置为灰色，不可再次点击，然后客户端和服务端的token各自独立存储，客户端存储在Cookie或者Form的隐藏域（放在Form隐藏域中的时候，需要每个表单）中，服务端存储在Session（单机系统中可以使用）或者其他缓存系统（分布式系统可以使用）中。)

②用来作身份验证

使用基于 Token 的身份验证方法，在服务端不需要存储用户的登录记录。大概的流程是这样的：

- 客户端使用用户名跟密码请求登录

- 服务端收到请求，去验证用户名与密码

- 验证成功后，服务端会签发一个 Token，再把这个 Token 发送给客户端

- 客户端收到 Token 以后可以把它存储起来，比如放在 Cookie 里或者 Local Storage 里

- 客户端每次向服务端请求资源的时候需要带着服务端签发的 Token

- 服务端收到请求，然后去验证客户端请求里面带着的 Token，如果验证成功，就向客户端返回请求的数据

简单介绍完之后再来回到我们的项目中，登录账号之后会发送收到一个token请求，我们要在拿到这个token请求之后才再去操作我们的功能。但是这个token不是一直固定不变的，缓存会定时刷新，这个刷新时间不会很长，换句话说过了时间之后token还没有使用就会自动作废，这时token将会自动刷新，想要继续操作就得再重新登录刷新一次，拿到一个新的token。

**2.7.2权限不足问题**

这个问题啊，这个问题其实也算不上问题。我们首先要明确一点，那就是不同的账号在登录的时候能够操作的东西是不一样的，换句话说就是会有权限差别的，对不同的账号赋不同的权限

**2.7.3报400、500错误**

**2.7.4报null错误**

跑代码抛null异常错误，最直观的就是代码逻辑哪一个环节有问题

**2.7.5断言不起作用问题**

所谓断言，就是Assert的一个用法，这主要针对在某种业务逻辑上的使用，比如说我们在实现某项功能的时候要保证传来的参数必须有某个字段，如果为空则直接抛出异常。

判空的操作有两种：

一种是利用我们的工具类StringUtilsMycc下的方法isEmpty，但是这个一般是去判断一个集合是否为空，比如上面提到的添加车位信息的时候查询一遍num在表里是否存在，就可以用这个方法，所谓判空的意思就是如果查询到的结果不为空，说明我们想要查询的东西集合里有、存在，如果为空，说明表里面没有我们想要的结果。根据这个就可以很好地去做一些逻辑上的判断。

另一种是Assert的用法，这个是只针对传来的其中一个字段，判断某一个字段是否为空。





上面这两个判空的方法一定要仔细注意这两者的区别，前者是判断传来的ps.getCommunityId不为空，为空则抛出“小区id不能为空”异常；后者是判断传来的ps.getCommunityId为空，不为空则抛出“小区id不能为空”异常。

所以，断言不起作用很大程度上是把这个判断的逻辑给弄反了。

**2.8如何配置连接数据库**



在项目的resources目录下创建application.yml，在该配置文件下配置如图信息

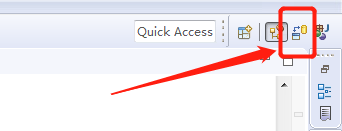
1. **Eclipse软件使用**

**3.1上传并更新代码**

我们的开发工具以eclipse为主要开发工具，项目小组采用各自本地开发代码上传至远程git合并。每当有成员上传代码时一方面为了避免代码冲突都要及时更新自己代码。

一个最基础的笨办法步骤：

①直接在项目视图右上方点击：



②依次是下拉更新代码、上传代码、查看代码冲突：

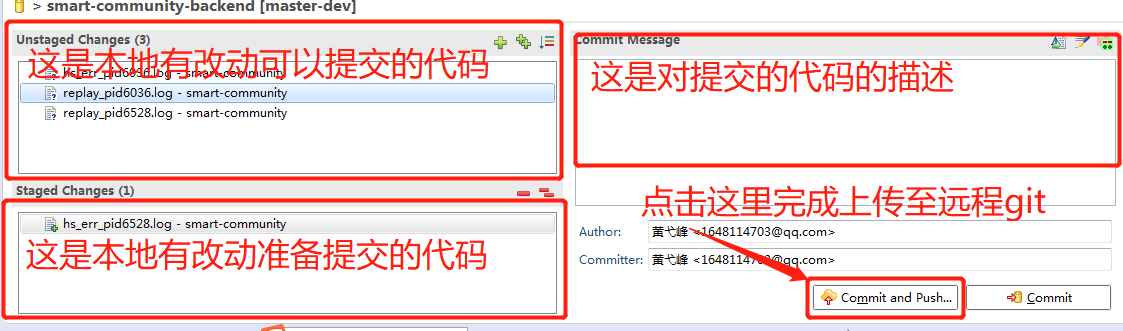


③切记在上传代码时一定要先更新代码：



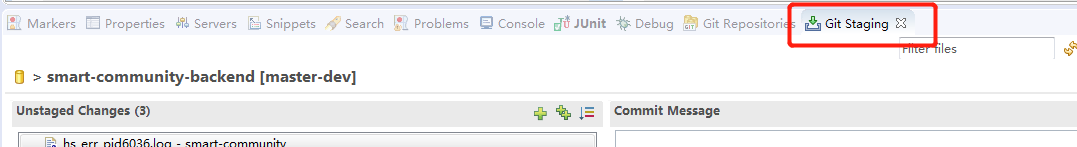
④更新之后选择这儿：



⑤ 

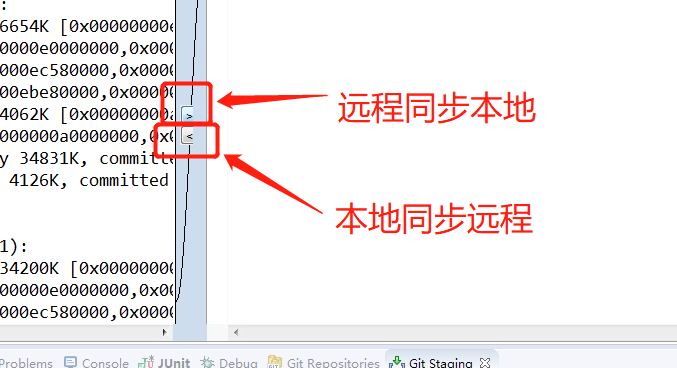
**3.2解决冲突最好的办法**

理论上来说解决冲突的最好办法就是不要出现代码冲突，每一次养成一个好习惯，及时更新自己的代码或者同时要改动某一个类的时候相互及时沟通这样绝对不可能出现代码冲突。但是偏偏就有冲突怎么办？第一想法是代码回退，回退至上一个版本，但是那样自己改了些啥不久都没了嘛。



还是我们的Git Staging，遇到代码冲突你得明白为什么回出现冲突？简单点说就是你本地的的代码有改动，在你没有上传更新之前别人也在该代码上做了改动并且上传了，你没上传，这下你的代码版本跟更新之前不一样了，远程git合并不了，因为不一样怎么合并嘛？

选择发生冲突的类，点击左箭头，表示本地先和远程一致



我们的目的是将本地代码同步上传到远程。这样冲突解决完了之后再更新一下代码就可以了。

**3.3合理使用断点配合Debug解决代码异常问题**

1. **Gitlab**
2. **新建分支到合并至主分支**
3. **查看上传的代码以及我的项目**
4. **Navicat**
5. **如何新建表以及注意事项**
6. **索引、主键、联合主键、唯一约束、自增长**
7. **建表的命名规则**
8. **如何通过错误日志里的sql语句去查找表问题所在**
9. **如何查看数据库连接信息以及配置**
10. **Postman**
11. **解析Body内的代码意义**

我们在编写好了一个接口服务之后使用工具Postman进行一个简单的本地或者是远程的接口测试，该工具的介绍不再描述。



通常我们是以json报文的形式进行数据交互，分为报文头、报文体。这是某一个接口测试的大文本。

“header”中：“serviceId”:”DeptTree”就是我们这次操作要测试的接口，“pageNo”、”pageSize”则是分页查询中的页码和页面数，针对查询接口统一约定在请求头中添加这两个字段，其余字段则是一些基本的码。

“body” 中：在这里面则是需要跟我们的接口具体业务相关，要知道我们的接口服务是针对数据库中表的操作，针对表中的每一个字段，在报文体中就是选传我们接口需求的字段值。例：”communityId”:1，就是说我们在实现”DeptTree”这个接口功能的时候需要传一个小区id为1的字段值过去以完成我们的测试。

**注意：body中的json格式很重要，为确保数据的准确性，格式很重要。**

1. **发送一次请求并成功的基本操作**

这里不讲postman的使用方法，只结合实际测试需求讲解：

我们在发送请求时首先要登录，发送登录请求，获取返回来的token



Token拿来是干嘛的？在我们的请求报文中我们有地址，但是怎么发送过去得到响应？当然是拿到我们的token。

（详细查阅有关token知识）

1. **Header? token?**