# 一、相关软件的安装

软件安装这部分都是实际操作，我这里就不再叙述详细步骤了。

# 二、mysql中数据库的设计

## C:\Users\gongc\Desktop\20190322225211.jpg1. 整型相关知识

## 2. 日期型相关知识

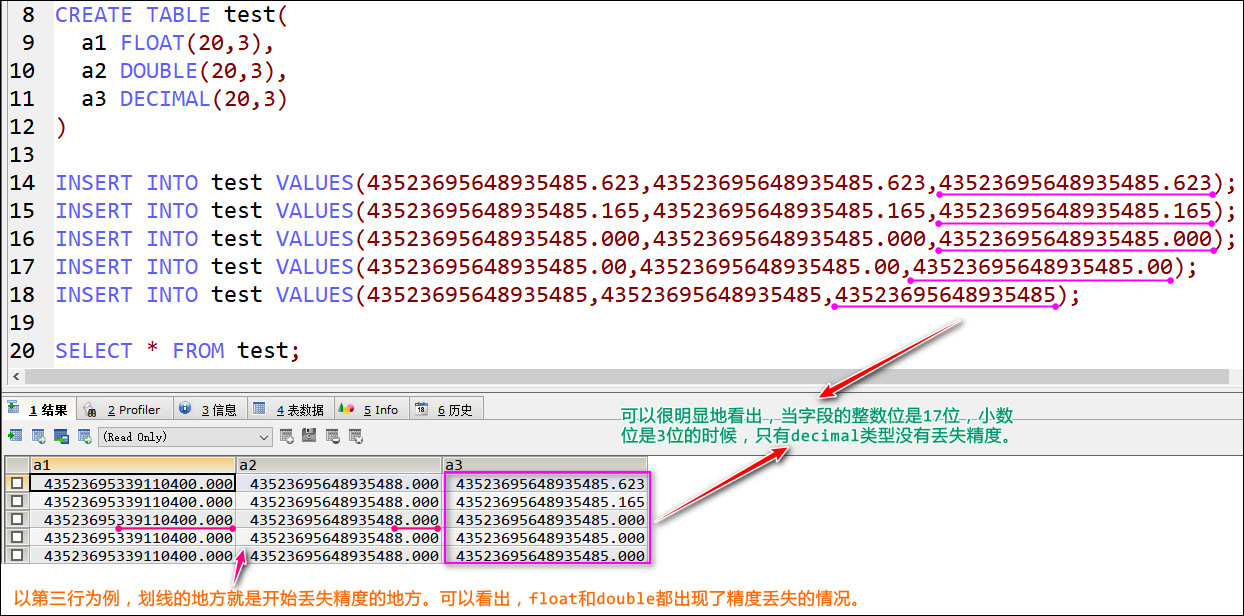
我们在建表的时候，经常使用到的日期类型是date。但是在该项目中，老师使用的日期类型不在是date，而是datetime。Date类型只能精确到天，而使用datetime类型的话则可以精确到秒，具体格式如：2019-3-23 10:25:38。

## 3. 字符型相关知识

老师在建表的时候，使用到了text这种字符类型。我们平常的时候使用最多的字符类型是varchar类型，比如varchar(10)，括号里的数字用于限定长度，而使用text类型的话，则表示长度不能超过**216**。除此之外，还有tinytext、mediumtext和longtext这三种类型。tinytext表示长度不能超过28，mediumtext表示长度不能超过224，longtext表示长度不能超过232。

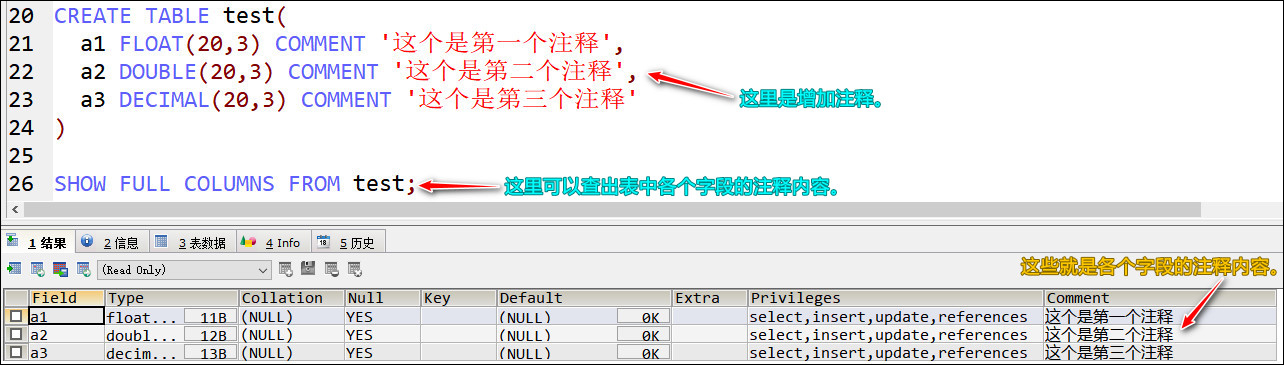
## 4. float、double以及decimal的区别

老师在建表的时候，凡是跟金钱有关的字段，使用的类型都是decimal类型。这种类型比较适合于对精确度要求比较高的情况下。如果在金钱字段上使用的不是decimal类型，而是float或者double类型的话，金钱字段的位数少的时候还没事，但是如果位数很多的话，就会出现丢失精度的情况。至于丢失的程度，可以从下面的截图中详细地观察到。



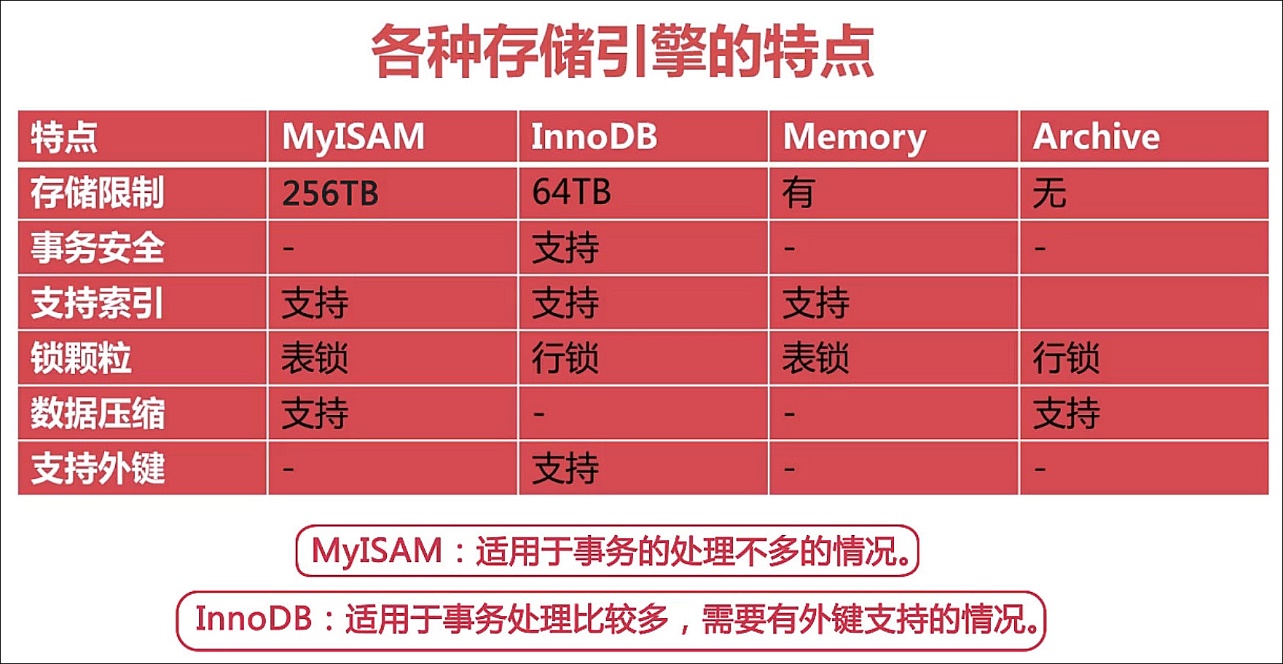
## 5. comment属性的用法

使用comment可以给字段加注释（或者说是描述），需要注意的是，这里的注释并不是别名。添加注释以及查询注释内容的方法都在下面的截图中了。



## 6. 存储引擎的相关知识

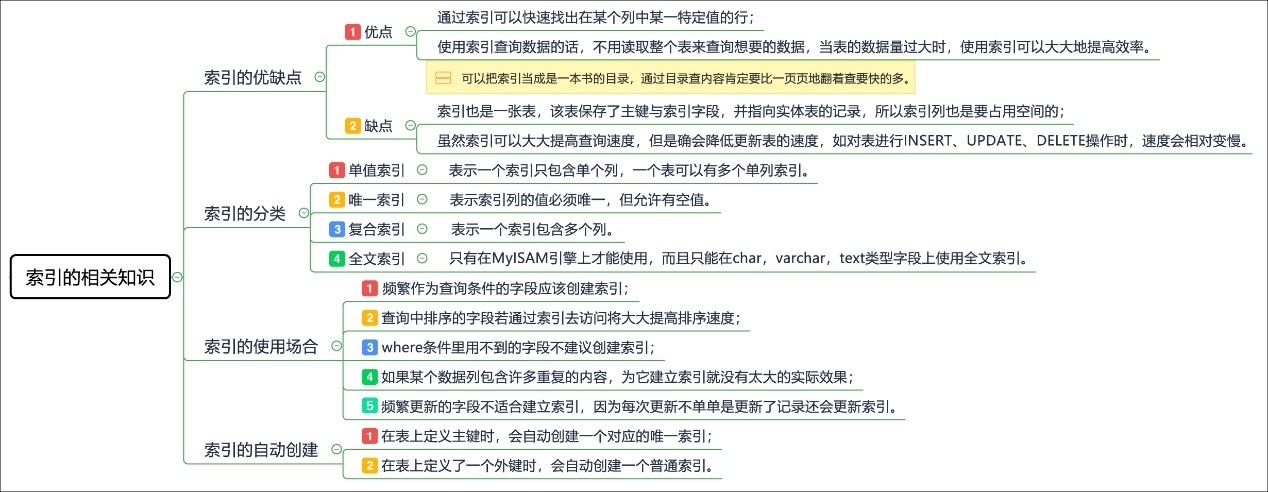
存储引擎的种类比较多，不过MyISAM和InnoDB是最常用的存储引擎。下图就是各种存储引擎特点的一个对照图。



老师在课程中使用的是InnoDB引擎。有一点需要注意的是，如果使用了InnoDB引擎的话，再使用索引，就只能使用Btree索引，而不能使用Hash索引了，因为Hash索引不支持InnoDB引擎。

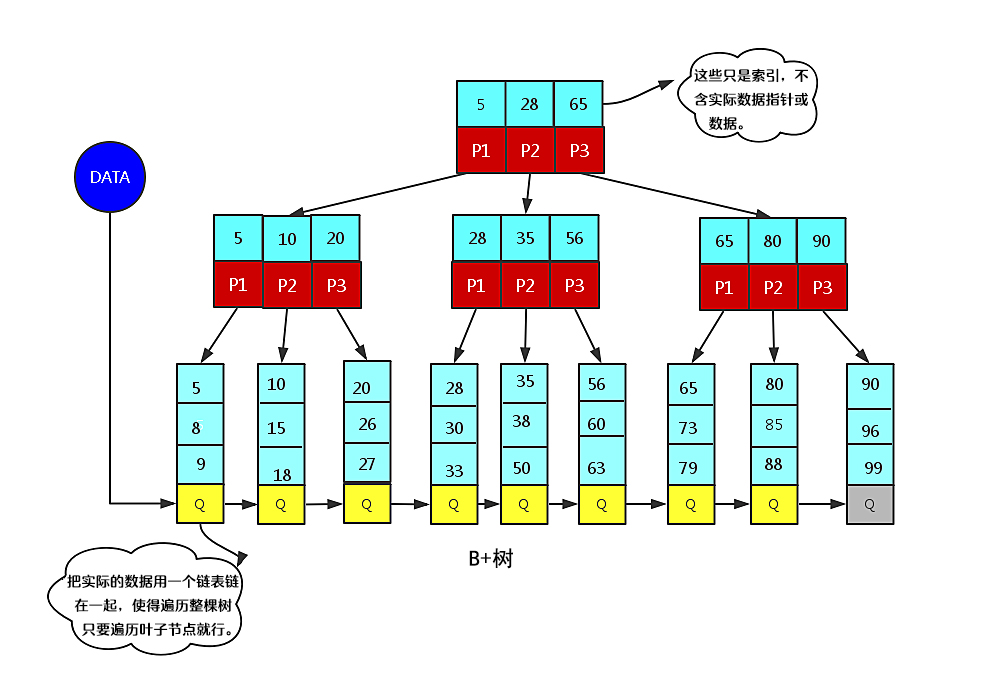
## 7. Btree索引和Hash索引

### 7.1 索引的相关知识



### 7.2 Btree索引

Btree是以B+树的结构存储数据的，如下图所示。



**Btree索引的特点：**

（1）Btree索引能够加快数据的查询速度；

（2）Btree索引更适合进行范围查找。

**Btree索引的使用场景：**

（1）全值匹配的查询适合使用Btree索引；

（2）匹配最左前缀的查询适合使用Btree索引；

（3）匹配列前缀的查询适合使用Btree索引；

（4）匹配范围值的查询适合使用Btree索引；

（5）只访问索引的查询适合使用Btree索引。

老师在建表的时候使用的就是Btree索引查询，不过Btree索引有一个使用限制，那就是如果不按照索引最左列开始查找，则无法使用索引。

### 7.3 Hash索引

**Hash索引的特点：**

（1）Hash索引是基于Hash表实现的，只有查询条件精确匹配Hash索引所有列的时候，才能够使用Hash索引；

（2）对于Hash索引中的所有列，存储引擎都会为每一行计算一个Hash码，Hash索引中存储的就是Hash码。

**Hash索引的使用限制：**

（1）Hash索引必须进行二次查找；

（2）Hash索引无法进行排序；

（3）Hash索引不支持部分索引查找，也不支持范围查找；

（4）Hash索引中Hash码的计算可能存在Hash冲突。

## 8. 数据库设计的小技巧

该项目的数据库中总共创建了八张表，而这八张表中都引入了时间戳，即都增加了数据创建时间和数据更新时间这两个字段，老师使用的是create\_time字段和update\_time字段。由于数据的更新次数可能会很多，所以这里的update\_time字段只保留最近一次更新的时间。使用这两个字段的好处是，我们可以知道数据的更新变化时间，这样的话，我们在查询一些业务或者对一些业务进行分析的时候，都会非常方便。

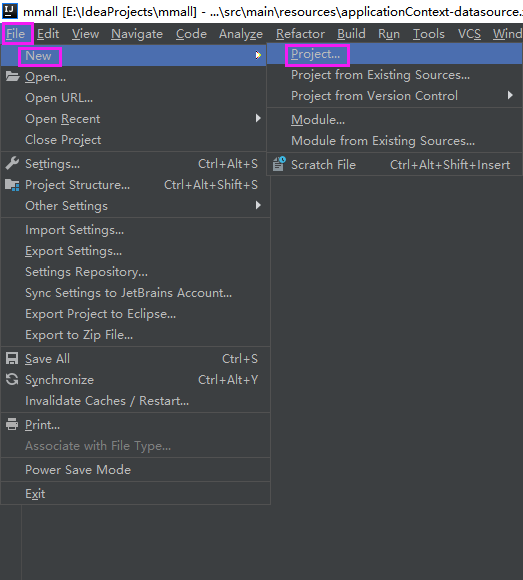
# 三、项目的初始化

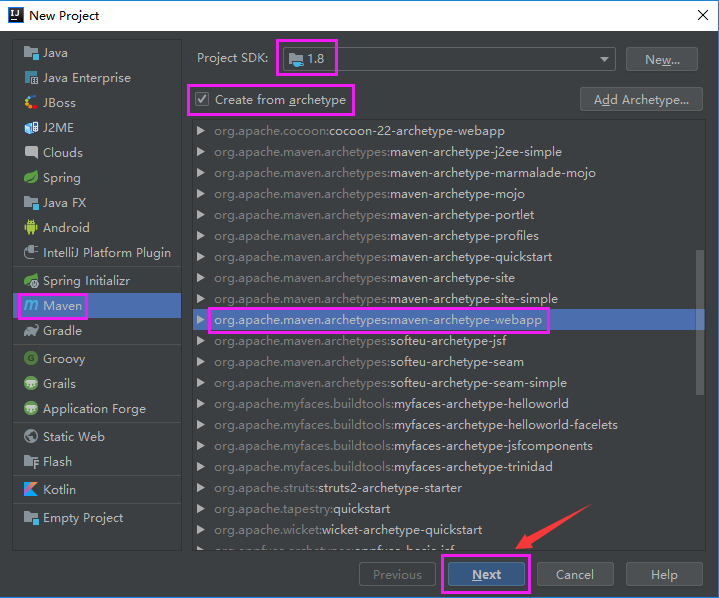
## 1. 数据库的初始化

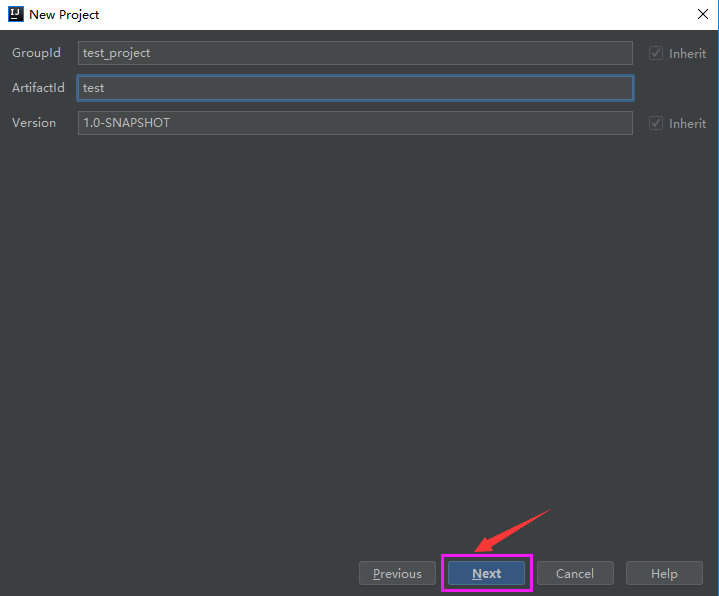
创建一个数据库（mmall），并设置数据库的字符集为utf8。然后在该数据库中依次创建用户表（mmall\_user），分类表（mmall\_category），产品表（mmall\_product），购物车表（mmall\_cart），收货地址表（mmall\_shipping），订单明细表（mmall\_order\_item），订单表（mmall\_order）和支付信息表（mmall\_pay\_info）。

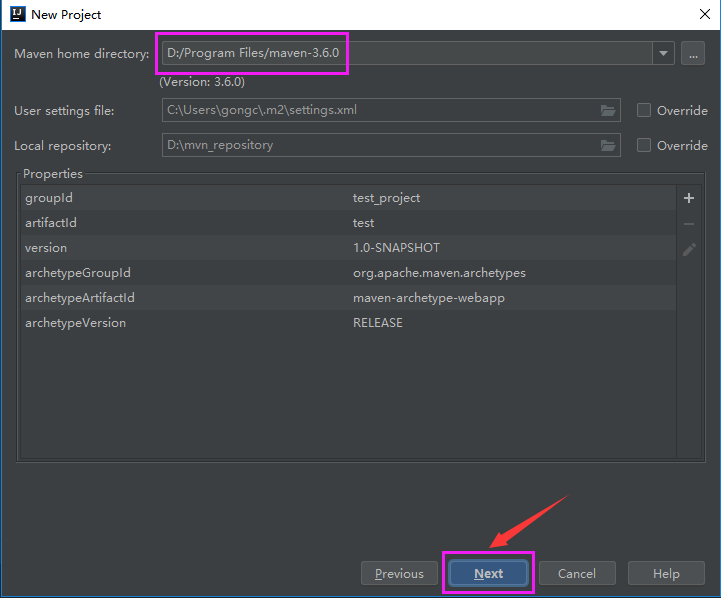
## 2. 使用IDEA新建maven项目

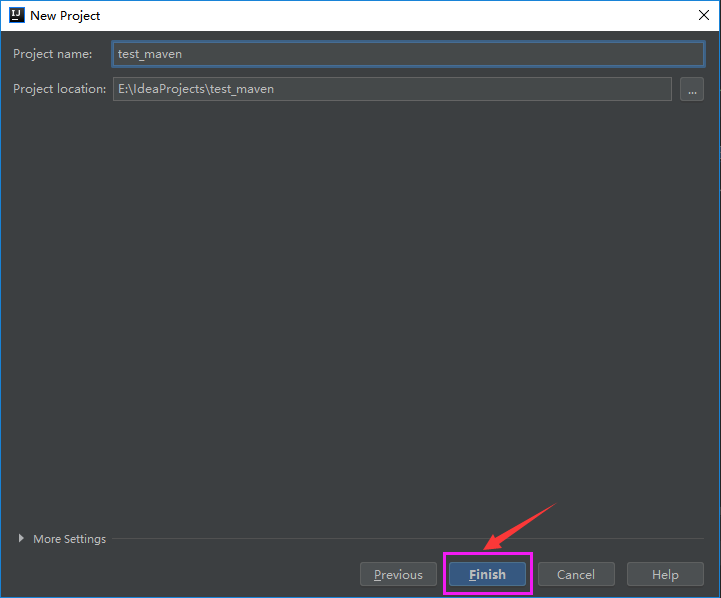
我已经把如何新建一个maven项目的步骤都截图下来了，根据下面截图的步骤就可以新建一个maven项目啦。











## 3. git的初始化

关于git初始化的步骤这里就不再写了，这里只说一点，那就是如果想使用SSH登录的话，不要忘记配置公钥。

## 4. mybatis的三剑客

这里所提到的mybatis三剑客就是三个可以提高代码开发效率的工具，它们分别是mybatis-generator，mybatis-plugin和mybatis-pagehelper。

mybatis-generator这个工具可以自动生成dao层的接口，实体类以及实体类对应的Mapper.xml文件，它可以大大提高我们的开发效率。

mybatis-plugin主要是方便实体类及其对应的Mapper.xml文件之间的切换，使用这个工具我们可以在一个实体类的某个方法上快速切换到该实体类对应的Mapper.xml文件中该方法对应的增删改查语句上，反之亦然。我们可以在idea软件的左上角的File中找到settings，然后选择plugins，在该项的右侧直接搜索mybatis-plugin安装即可。

如果想使用mybatis-pagehelper分页组件的话，需要在pom.xml以及spring的配置文件中引入相关的配置。

## 5. logback日志组件

logback是一个开源的日志组件，是log4j的作者开发用来替代log4j的。使用该组件主要就是引入一个logback.xml文件，然后详细的日志相关的配置都在这里面。还有就是，也需要在pom.xml中进行配置。

ftp服务器也引入了一个配置文件，是以properties结尾的，这里提一下，就不再专门用一个标题介绍它啦。

## 6. 两个提高效率的工具

这里说的两个工具分别是Restlet Client和fe助手，Restlet Client可以用于模拟http请求，而fe助手具有json格式化等功能。这两个工具都是谷歌浏览器的插件，可以在谷歌浏览器的应用商店里面下载到这两个插件。

注：视频里项目的初始化中所涉及的内容并不止以上提到的这些，这里提到的只是其中的一部分，像web.xml中的配置等这里就没有进行介绍。

# 四、用户模块开发

## 1. 常量类和通用的服务端响应对象

### 1.1 常量类的使用

用户模块的开发中，我觉得扩展性很强的就是这里定义的常量类以及另一个通用的服务端响应对象，我觉得以后在项目开发中，也可以像老师这么定义。

关于常量类，顾名思义，就是专门用于存放常量的一个类。老师把用户模块中常用到的一些常量都放到了这个类中，比如“username”以及“email”这些在校验的时候经常要以字符串的形式用到的常量，还有就是用于表示角色的常量，就像0表示普通用户，1表示管理员等，这些就可以都放到常量类中，然后用到的时候直接引用就可以啦。

### 1.2 通用的服务端响应对象

我觉得这个类设计的非常棒。视频中用户模块所有返回给前端的数据都是以json串的形式返回的，但是由于返回给前端的内容有可能是操作成功后返回的内容，也有可能是操作失败后返回的内容。那么如何通过返回的json串就判断出是操作成功了还是失败了呢？这个通用的响应类就可以做到。

首先这个类应该是一个泛型类，因为虽然返回给前端的都是json串，但是json串中的key对应的value值的类型无法确定，所以使用泛型最合适。

在这个类中，定义了三个属性。一个表示后端代码的操作状态，如果这个状态值是0，表示代码执行的结果就是我们预期的结果；如果这个状态值是1，就表示执行结果并不是我们想要的结果，如果是其他数字，我们也可以再定义这些数字各自表示的意思。然后还有一个属性是专门用于描述的属性，好比在登录时，返回给前端的状态值是1，这说明返回结果不是我们预期的结果，那么出现这种问题的可能性也有很多，有可能是登录密码错误，也有可能是用户名未填写等。如果我们定义了一个用于描述的属性的话，那么就可以很清楚地知道出错的原因了。第三个属性也是最重要的一个属性，后端返回给前端的数据都是要通过第三个属性展现出来的，所以这个属性的类型是泛型，因为它有可能是一个对象也有可能只是一个字符串。

由于返回给前端的json串中不确定会包含以上提到的三个属性中的哪些属性（可以确定的是，一定包含状态值这个属性），所以这个通用类中定义了很多的重载方法。

## 2. 用户模块详解

这里是使用注解的方式进行开发的，该模块下有用户登陆，用户登出，用户注册，校验用户名或邮箱是否存在，获取用户登陆信息，获取用于找回密码的问题，用于找回密码的问题的答案的校验，重置密码，登录状态下的重置密码，获取用户的个人信息以及更新用户的个人信息这些功能。

在开发中对于密码的处理问题上，使用了MD5以及增加盐值的方式进行加密。然后实体类、实体类对应的Mapper.xml文件以及dao层的代码都是使用mybatis-generator这个工具自动生成的。还有一点需要特别说一下的是，dao层并没有使用任何类，而是用的接口，然后通过接口中不同的方法来对数据库进行增删查改操作，传递的参数如果不止一个的话，需要用到@Param注解，这点是我以前不知道的新知识。

在进行校验的时候，比如校验用户名是否存在，sql语句中是使用username来进行查询的，然后使用count(1)来统计个数，如果统计结果大于0就表示当前用户名的用户已经存在于数据库中了，这里使用的count(1)和count(\*)的效果是一样的，但是效率却是要高一些。另外，如何在service中判断数据库中已经存在这个用户名的用户呢？这个其实很简单，在service中调用dao中的方法进行查询的时候，如果sql语句查到数据，返回的生效行数就会大于0，这就表示数据库中已经存在了该用户名的用户了，反之就表示使用该用户名在数据库中没有查到数据，就说明数据库中没有该用户名的用户。

# 五、分类管理模块开发

## 1. 新知识点汇总

### 1.1 intValue方法介绍

intValue()是java.lang.Number类的方法，Number是一个抽象类。Java中所有的数值类都继承它。也就是说，不单是Integer有intValue方法，Double，Long等都有此方法。不过在这里用的还是Integer中的intValue方法。这个方法的作用就是把一个值从Integer类型转成int类型。

### 1.2 @RequestParam注解的用法

使用@RequestParam注解的作用就是，如果没有给一个方法的某个参数传入值，那就给该参数设置一个默认值，案例如下。



上面这个案例中的parentId是一个方法的参数，注解中的value属性值的内容就是参数名，而defaultValue属性值的内容就是我们给parentId参数设置的默认值。

### 1.3 Sets类和Lists类

Sets类和Lists类都是com.google.common.collect包下的类。这个模块的开发中用到了Sets类的newHashSet()方法和Lists类的newArrayList()方法。这两个方法都很简单，像newHashSet()方法其实就是new一个ArrayList()，newArrayList()方法也是同样的道理。

## 2. 分类管理模块详解

这个模块下主要有添加分类，更新分类名称，查询子节点的分类信息以及递归查询当前节点及其子节点的id这些功能。

在这个模块中，没有什么特别有难度的东西。这个模块的最大亮点就是，使用递归算法的方式查找子节点。

# 六、商品管理模块开发

## 1. 新知识点介绍

### 1.1 pojo与vo的关系

在这一模块中，老师用到了vo包下的类，其实vo包下的类和pojo包下的类是差不多的，都是实体类。以展示商品信息为例，由于Product类已经在pojo包下与dao层相应的类相互对应，所以我们无法再对Product实体类中的属性进行修改。而展示商品信息的时候，我们不一定想要展示Product类中的所有属性，有可能只需要展示Product类中的部分属性，也可能还会需要展示一些Product类中没有的属性。这时候就可以使用一个过渡的实体类来完成这一操作，然后把这些类统一放在vo包下好方便管理。所以vo包下的实体类其实就相当于是一个过渡类，这些类是在pojo包下的对应的实体类的基础上，增加或删除一些属性后形成的满足我们业务需求的新的实体类。

### 1.2 DateTimeUtil工具类

还是以展示商品信息为例，在后台展示商品信息的时候我们一般会希望看到该商品的创建时间和最后一次的更新时间。关于这两个时间戳，我们在数据库建表的时候就已经给加上啦，但是返回的json文本中这两个时间显示的方式不是我们中国人常用的方式，所以不便于我们阅读，这时候就需要一个工具类来解决这个问题。在这个模块中，把这个工具类起名为DateTimeUtil。该工具类中设置时间以"yyyy-MM-dd HH:mm:ss"格式显示，这样就变得便于我们阅读了。

老师在课上提到了joda-time这个技术，而DateTimeUtil这个工具类就是使用这个技术完成的。

### 1.3 Mybatis的分页组件

这一模块中，在展示商品列表等地方使用到了mybatis-pagehelper分页组件。我这里就只是提一下这个组件，由于我对这个组件的使用逻辑没有看明白，所以这里就不再进行详解了，知道在这里用到了就行。

## 2. 商品管理模块详解

### 2.1 前台

商品管理模块中的前台主要有获取商品详情、商品搜索及动态排序这两个功能。

获取商品详情这个功能比较简单，这里是通过商品的id查询到对应的商品，然后再通过vo包下对应的用于过渡的实体类来展现我们需要展现的商品信息。由于是前台，所以需要注意的是，已经下架或者删除的商品是不能获取到商品详情的。

商品搜索及动态排序这个功能是通过关键字或者分类id来搜索商品的，然后再通过分页插件中的方法对搜索到的商品的价格进行升序或者降序排序。在搜索的时候，这里说的关键字指的是商品名称中的关键字，而且这里使用的是模糊搜索。搜索的时候还有一点需要注意，由于是前台搜素，所以下架或者已经删除的商品不能显示在搜索结果中。

### 2.2 后台

商品管理模块中的后台主要包括新增或更新商品、修改商品的上下架状态、获取商品详情、获取商品列表、商品搜索、图片上传和富文本上传这些功能。

新增或更新商品的方法中首先是对商品图片这一块进行了一些设置，商品的图片肯定不止一张，这里是设置第一张为商品的主图。另外，新增或者更新的区别就在于传入的商品信息中是否包含商品的id，如果不包含就新增商品，如果包含就更新商品。

修改商品的上下架状态这个方法比较简单，这里是直接通过参数传入商品的id和状态信息，然后根据商品的id把商品的状态更新成方法参数中传入的状态。

后台的获取商品详情的方法内容和前台的差不多，不过由于是后台，所以即使是已经下架的商品，这里也是可以获取到商品详情的。

获取商品列表这个方法需要用到分页插件和vo包下的类，然后在写SQL语句的时候也根据id进行了排序，这样一来，展示出的商品就是通过id排序后的商品了。

商品搜索方法比较简单，这里是根据商品的名称或者商品id进行搜索的，商品名称使用的是模糊查询。另外，由于是后台，所以即便是下架的商品，也可以搜索出来。

最后一个图片上传和富文本上传功能其实是两个方法，我是没看出它俩的区别，所以就放在一起说了。图片上传使用的是springmvc的上传功能，这里肯定也能上传文件，不过这个模块用不着上传文件，所以这里就说成是图片上传啦。另外，为了访问方便，这里也连接到了FTP服务器。也就是说，图片不仅会上传到tomcat服务器上，同时也会上传到FTP服务器上，这样做是为了以后访问方便。另外，为了减轻tomcat服务器的压力，在图片上传到FTP服务器上后，就自动删除了上传到tomcat服务器中的图片。当然，这一操作是通过代码完成的，并不是人为进行删除的。

# 七、购物车模块开发

## 1. 新知识点介绍

### 1.1 Splitter类的两个方法

在删除购物车中的商品的时候，使用的是根据商品id进行删除。由于我们删除商品的时候可能会出现同时删除多个商品，所以这里在方法中传入商品id的时候，可以允许传入多个id，然后这些id使用逗号进行分隔。可是如何把传入的像"1,2,3"这种类型的字符串分隔成一个个的id呢？com.google.common.base包下的Splitter类就可以做到。

|  |
| --- |
| List<String> productList = Splitter.on(",").splitToList(productIds); |

上面的代码中，productIds就是一个类似"1,2,3"的字符串，这里通过Splitter类中的两个方法把"1,2,3"通过逗号分隔成了三个字符串并都存入了List集合中。这样就把不同商品的id分开存放了，在sql语句中，只要在sql语句后使用in(1,2,3)就可以把对应的商品删除掉了。

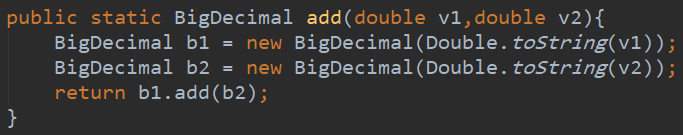
### 1.2 高复用性的一个方法

在购物车模块使用了一个高复用性的方法，这个方法返回的是一个CartVo实体类，这个实体类中把购物车中所有商品形成的集合作为了一个属性，还把所有商品总价格等作为了一个属性。这个高复用性的方法在返回CartVo类之前有很多的业务逻辑代码，这些代码中包含了怎么把购物车中所有商品的所有详细信息获取后把这些商品放入集合中，还包含了商品总价格的计算以及加入购物车的商品的数量大于商品库存数量的解决办法等。正是由于这个方法中的这些逻辑代码，才让方法返回的CartVo类中包含了那么多重要信息，而这些信息很多方法都可以用到，所以这个方法的复用性就变高了。

### 1.3 浮点型丢失精度的问题

在进行价格的加减乘除运算时，直接使用"System.out.println(0.05+0.01);"打印出来的结果是0.060000000000000005，并不是我们想象中的0.06。如果不解决这个问题的话，就会出现一种情况，比如有两个商品，一个0.05元，另一个0.01元，我们手上正好有0.06元，但是却买不了这两个商品，因为这两个商品的价格之和其实是大于0.06元的。如果想要解决这个问题，就需要用到BigDecimal这个类，因为使用这个类中关于加减乘除的方法，不会出现丢失精度的问题。

我们可以创建一个BigDecimalUtil类，然后在这个类中封装BigDecimal类中关于加减乘除的方法，当我们需要对价格进行加减乘除运算的时候，直接调用BigDecimalUtil类中对应的方法即可。以加法运算为例，如下图。



图中的add(double v1,double v2)方法就是BigDecimalUtil类中关于加法的一个方法，这个方法下封装了BigDecimal类的add方法。所以我们调用BigDecimalUtil类中的add方法，其实就是调用BigDecimal类中的add方法。有一点需要特别注意，这里在创建BigDecimal的对象时，只能使用String类型的构造器。

## 2. 购物车模块详解

该模块下主要包括添加商品到购物车、更新购物车中某个商品的数量、删除购物车中的商品、查询购物车所有商品(即购物车列表)、选中或取消选中购物车中的某个商品、全选或取消全选购物车中的全部商品、获取购物车中的商品数量这些功能。

添加到购物车这个方法的逻辑就是，先传入用户id、商品id和商品数量，然后通过用户id和商品id去数据库中查找有没有对应的Cart类，如果没有，说明这个产品不在这个购物车里,这时就需要新增一个这个产品的记录；如果已经存在购物车中，就把商品的数量进行相加，然后更新数据库中的数据即可。

更新购物车中商品的数量这个方法也是需要传入用户id、商品id和商品数量的，然后先根据用户id和商品id查出对应的Cart类，之后再把这个Cart中的商品数量的属性值设置成我们传入的商品数量的值，最后更新数据库中的数据即可。

删除购物车中的商品这个方法需要传入两个参数，一个是用户id，另一个是商品id组成的字符串。这里删除的时候是根据商品id进行删除的，但是可能删除的不止一种商品，所以商品id的类型不能是Integer，只能是String类型。因为如果是Integer类型的话，只能删除一种商品，就不能同时删除多种商品了。当传入的id是字符串的时候，我们可以在字符串中用逗号把各个商品的id隔开。然后通过一些方法使用逗号把字符串分隔成一个个纯数字的字符串，再把他们都放到List集合中。在sql语句中删除的时候，可以通过用户id以及这个纯数字的字符串集合进行多项删除。

查询购物车所有商品这个方法可以直接调用该模块中已经写好的那个高复用的方法，那个方法返回的是一个CartVo类，直接把这个类放到ServerResponse类的方法中，然后返回一个ServerResponse<CartVo>即可。

选中或取消选中购物车中的某个商品、全选或取消全选购物车中的全部商品这四个方法是有很多相通的地方的，所以在service层中只需要写一个方法即可，然后在Controller中都调用service层中这一个方法。Service层中的这个方法需要传入三个参数，分别是用户id、商品id和checked参数，checked参数用于表示是否选中。全选的方法是不需要商品id的，所以可以设置商品id为null，然后把checked参数设置为选中即可。假设1表示选中，0表示未选中的话，这里就设置checked参数的值为1即可。取消全选的方法也是要先把商品id设置成null，然后把checked参数的值设置成0即可。单选和取消单选的方法和上面提到的全选和取消全选的逻辑是差不多的，主要区别就在于它们需要传入一个商品id。因为是单选或者取消单选，所以我们需要通过商品id定位到具体的商品上，然后才能进行相应的操作。

获取购物车中的商品数量这一方法的逻辑是比较简单的，在sql语句中直接使用用户的id查询出商品数量这一字段的和即可，这里需要用到sum函数。这里还有一点需要注意的是，为了防止查询结果出现null的情况，这里还使用了一个IFNULL函数，使用这个函数的作用是，如果sum函数的结果是null的话，就把null变成0，这样就避免了出现null的情况了，例如"IFNULL(sum(quantity),0)"。

# 八、收货地址管理模块开发

## 1. 注意事项

这一模块没有什么难度特别大的地方，只有一个地方需要特别注意一下，那就是横向越权问题。在删除地址、更新地址以及查询地址详情的时候都有可能出现横向越权问题，所以在这三个功能开发的时候，就算是处于登录状态，在查询的时候，也要在sql语句中加上用户id这个字段。

## 2. 收货地址模块详解

该模块下主要包括新建地址、删除地址、更新地址、查询地址详情以及地址列表这些功能，这些功能都比较简单。

新建地址的时候，需要用户id和Shipping这个地址对象，将用户id设置到该对象的userId属性中后，直接插入整个对象进行新增即可。不过为了前端的方便，在新增成功之后，这里还给前端返回了该Shipping对象的id属性值。

删除地址的话，就直接根据用户userId和shippingId进行删除即可。更新地址也类似，虽然参数中传入的是userId和Shipping对象，不过在更新的sql语句的where后面是要有userId和shippingId才行的。查询地址详情也是一样，直接根据userId和shippingId进行查询。关于地址列表这个方法，使用了分页的技术，这里是先通过userId查询出该用户下的所有地址，然后把这些地址都放到一个List集合中，最后再结合分页技术把结果返回给前端。

# 九、支付模块和订单模块开发

## 1. 重难点介绍

我这里把支付模块和订单模块这两个模块放在了一起介绍，主要是因为这两个模块是相互影响的。在这两个模块中，我觉得重难点就是创建订单以及支付订单功能的开发。创建订单时，写了很多私有的方法供其调用，比如清除购物车中已经创建订单的商品的方法、减少产品库存的方法、生成一个订单的方法、生成订单号的方法以及获取订单中所有商品的总价等等一系列的方法。在支付订单时，使用了支付宝的官方Demo，然后进行了一些改动，由于我对这个官方Demo并不是很熟悉，所以支付订单的方法我就没有进行详细讲解。

这两个模块中还讲到了沙箱环境的使用，包括沙箱环境的配置等内容，还讲了很多支付宝官方的文档，这里就不再进行介绍啦。

## 2. 模块功能详解

### 2.1 前台

这两个模块中的前台部分有获取购物车中的商品信息、创建订单、支付订单、支付宝支付回调的方法、查询订单的支付状态、订单详情、未付款状态下取消订单、在个人中心查询所有订单这些功能。

#### 2.1.1 获取购物车中的商品信息

该方法需要传入一个用户id，即要处于登录状态，毕竟只有在登录时才知道获取的是哪个用户购物车中的商品信息。这个方法返回给前端的是一个orderProductVo实体类，这个类中主要有imageHost属性、productTotalPrice属性以及orderItemVoList属性，orderItemVoList的类型是List<OrderItemVo>。

该方法的逻辑如下。首先要从数据库中查询出该用户购物车中所有被选中的商品并存入集合中，即cartList集合。然后写一个通用且私有的方法getCartOrderItem，该方法的作用就是通过用户id和cartList获取到一个orderItemList，这个list集合中存的都是一个个的OrderItem。但是我们需要的是orderItemVoList，所以我们需要把这个集合中的OrderItem都遍历出来，然后都包装成OrderItemVo后再放到另一个集合中，那么这个集合就是我们想要的orderItemVoList集合了。在遍历orderItemList集合的时候还要把该集合下的每个OrderItem的订单金额进行相加，这样就得到了总金额。然后把总金额的值和得到的orderItemVoList赋给orderProductVo类对应的属性，再给这个类添加个主图，然后就可以把它返回给前端了。

#### 2.1.2 创建订单

该方法需要传入一个用户id和收货地址id，当创建订单成功后，需要返回一个具有更多订单信息的orderVo实体类给前端。

该方法的逻辑如下。首先要从数据库中查询出所有被选中的商品并存入集合中，即cartList集合。然后通过getCartOrderItem方法获取到orderItemList集合，这个集合里存的都是一个个的OrderItem，之后遍历orderItemList并求出所有OrderItem之和，得到订单总价。然后结合收货地址id、订单总价以及用户id生成订单，并将该订单的数据插入到数据库的订单表中。在生成订单的时候，订单号是以当前时间的毫秒数加随机数的形式随机生成的。订单生成后，再把订单号赋给该订单中的每一个OrderItem，然后再把该订单中所有的OrderItem批量插入到数据库的订单明细表中。之后就是减少相应商品的库存以及清理购物车中相应的商品。在生成订单时我们已经得到了Order对象，最后一步就是把该对象包装成OrderVo，然后返回给前端即可。

#### 2.1.3 支付订单

该方法需要传入一个订单号、用户id和支付二维码的路径。返回的是一个key和value都是String的Map集合。

该方法的逻辑我不好详解，因为该方法的主要内容都是引用的支付宝官方Demo示例中的内容，所以我就只大致说一下。既然是支付订单，首先就是根据用户id和订单号从数据库中查到要支付的订单。然后下面就是很多支付宝官方Demo中的一些设置，这些设置很多都是使用默认即可。然后下面还有创建扫码支付请求builder，设置请求参数，设置二维码的名字、尺寸和位置等，这里还要把二维码上传到ftp服务器上。最后返回给前端的是一个Map集合，这个集合包含了订单号、二维码的url等信息。

#### 2.1.4 支付宝支付回调的方法

该方法需要传入的是一个Map<String,String>集合，这个集合应该就是支付订单方法返回的那个Map集合。如果回调成功，该方法返回的必须是一个"success"字符串。

这个回调的方法没法直接在浏览器中进行测试，在支付订单的时候，会设置扫码支付请求参数，这些参数中就有一个url的参数，这个参数设置成回调方法的请求地址之后，在支付订单的时候就会调用这个回调的方法来完成支付宝的回调操作啦。在回调方法中我们一定要验证回调的正确性，验证它是不是支付宝发的。另外，还要避免支付宝的重复通知。

#### 2.1.5 查询订单的支付状态

该方法需要传入用户id和订单号，返回给前端的是一个布尔值，即true或false。true表示支付成功，false表示支付失败。

该方法的逻辑如下。首先通过用户id和订单号查询出订单，然后判断一下该订单目前的状态，如果该订单处于已付款或者已发货等状态的话，就给前端返回true。

#### 2.1.6 订单详情

该方法需要传入一个用户id和订单号，返回给前端的是一个OrderVo对象。

该方法的逻辑如下。首先通过用户id和订单号查询出对应的订单，然后再把查询出来的这个订单对象包装成一个OrderVo对象返回给前端即可。

#### 2.1.7 未付款状态下取消订单

该方法需要传入一个用户id和订单号，返回给前端一个能表明取消成功或者失败的状态就行。如果0表示取消成功的话，那么在取消成功后返回的状态为0就行。

该方法的逻辑如下。首先通过用户id和订单号查询出对应的订单，然后把这个订单的状态改成已取消的状态，之后再更新数据库即可。

#### 2.1.8 在个人中心查询所有订单

由于该方法涉及到分页，所以传入的参数中除了用户id外，还有使用分页插件时需要的两个参数，即pageNum和pageSize。由于使用到了分页，所以返回给前端的是分页插件中的PageInfo对象。

该方法的逻辑如下。首先通过用户id从数据库中查询出所有的订单，然后把这些订单都放到list集合中，得到orderList集合，之后写一个私有的方法，把这个orderList集合和用户id以参数的形式传入，通过这个方法得到一个orderVoList集合，这个集合中都是一个个的OrderVo。然后把这个集合以参数的方式传入到PageInfo对象中，最后把PageInfo对象返回给前端即可。

### 2.2 后台

支付模块是没有后台的方法的，订单模块的后台部分有通过订单号搜索订单、查询所有订单、订单详情、订单发货这些功能。

通过订单号搜索订单这个方法就是直接通过订单号向数据库中查询出相应的订单，然后把查询出的订单都包装成OrderVo，然后返回给前端即可。

查询所有订单这个方法就是直接从数据库中查询出所有订单即可，因为是后台操作，所以不需要传入用户id。把查询到的这些订单都包装成OrderVo后放入list集合中返回给前端即可。

订单详情这个方法就是通过订单号从数据库中查询出相应的订单，然后把这个订单包装成OrderVo对象后返回给前端即可。

订单发货这个方法首先是要通过订单号从数据库中查询出相应的订单，然后判断一下这个订单的状态，如果是已付款状态，那么就把它改成已发货状态，之后更新一下数据库，然后给前端返回一个发货成功的描述即可。