Javaweb-项目

# 项目介绍

后台管理 p2p\_management

实现登录，退出 产品管理（产品添加，产品查询，产品修改）

前台管理 p2p\_home

客户注册，登录，邮箱认证

帐户中心 产品查询 帐户信息查询 产品购买

技术架构：jsp+servlet+javabean+mysql+redis+filter

新的知识点:

1. 数据库操作的事务管理
2. Javamail
3. Beanutils工具它可以将请求参数直接封装成实体对象

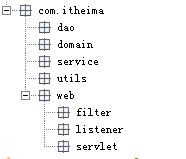
本次综合练习，主要是以ajax操作为主。

# 项目环境搭建

1. eclipse中项目环境搭建



搭建package结构



1. 关于数据库

五张表

* + - 1. user表，它是后台用户表
      2. product表，产品表
      3. account表 客户帐户表
      4. product\_account客户购买产品表
      5. customer 客户表

# 功能实现

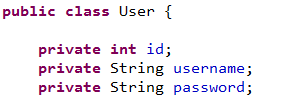
## p2p\_management功能实现

### 简单登录与退出

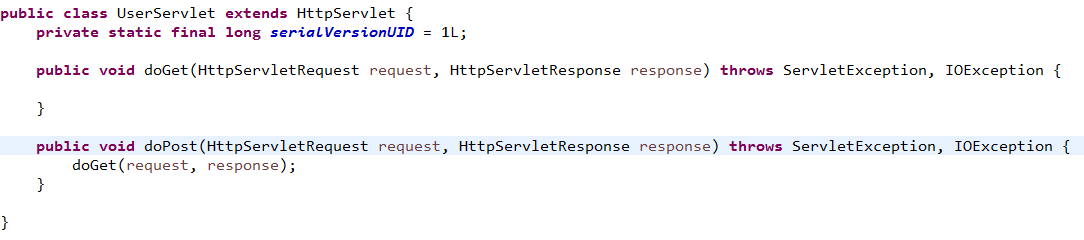
#### 登录操作

登录页面login.html

步骤1：创建一个User类它与表user对应



步骤2：创建UserServlet



步骤3：将login.html页面转换为login.jsp



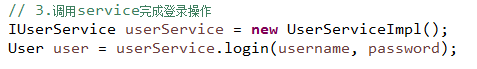
步骤4:在servlet中完成用户登录操作

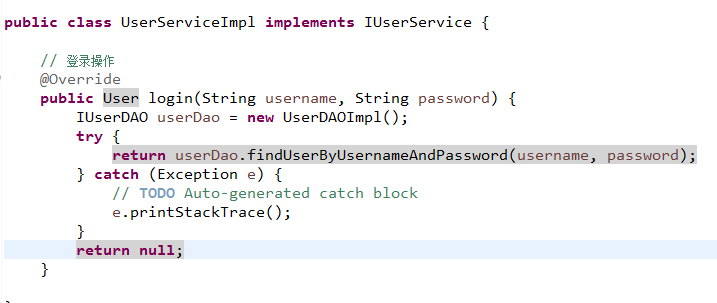
1.先获取username,password



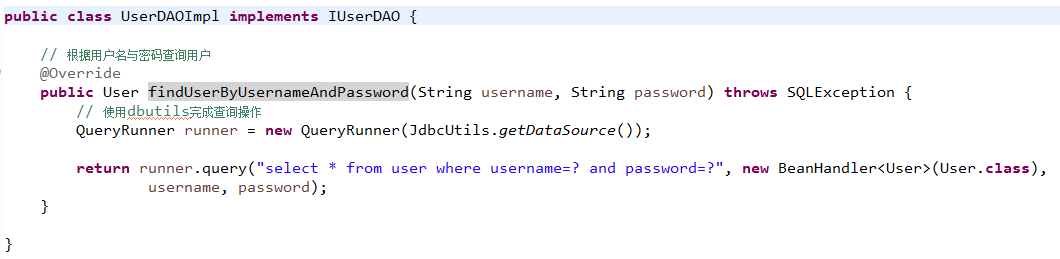
2.完成验证操作

3.调用service完成操作

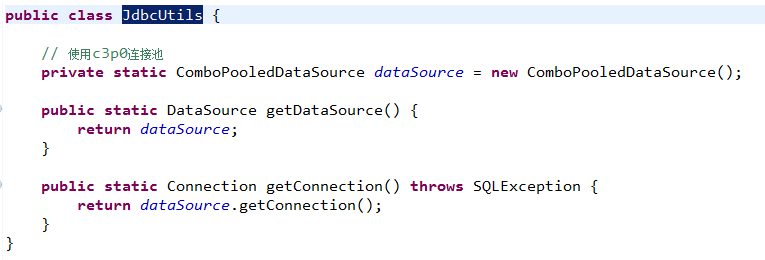




4.在service中调用dao完成操作



操作数据库使用的是c3p0连接池与dbutils工具，提供了一个JdbcUtils



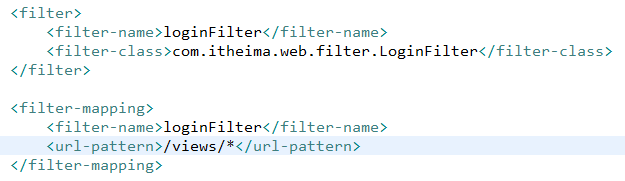
步骤5：从数据库中查询出user,判断是否登录成功，进行页面跳转



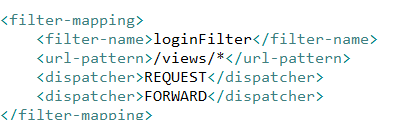
#### 用户未登录不能访问/views/hom.jsp

可以使用filter来完成操作





注意：上面的filter配置它只能对REQUEST请求拦截，我们也需要对FORWARD拦截



#### 退出操作

1.在home.jsp页面显示登录用户的名称



2.退出操作

我们退出时，一定要将session销毁，在将页面关闭。

我使用ajax操作来完成。

步骤1：创建home.js文件，在home.jsp页面导入



步骤2：修改home.jsp页面



步骤3:在home.js文件中创建一个函数shutdown()来完成关闭操作



### 产品管理-产品查询

关于产品信息的查询，我们使用ajax来完成操作，并且没有分页。

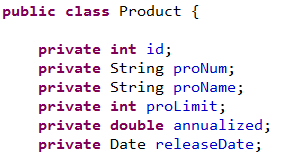
思路:

1.当home.jsp页面加载时就会向服务器发送请求来获取产品信息

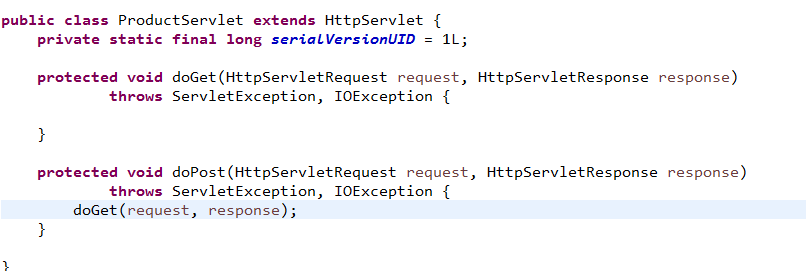
2.服务器端在servlet中处理请求，调用service,dao完成产品信息查询，并将信息转换成json响应到浏览器端

3.在浏览器中得到服务器响应的数据(json)，对数据进行处理，展示在home.jsp页面上。

#### 步骤1:创建Product模型



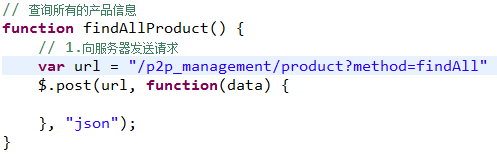
#### 步骤2创建服务器端的servlet ,service,dao



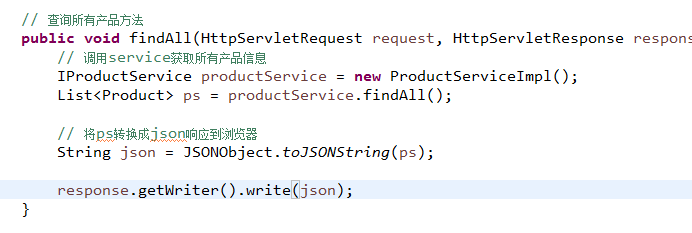
在service包中创建IProductService 与其实现ProductServiceImpl

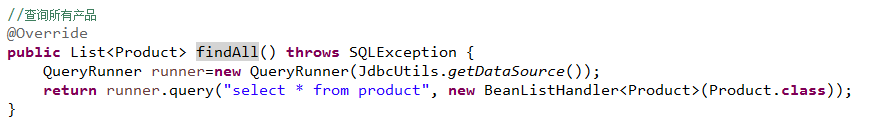
在dao包中创建IProductDAO 与其实现ProductDAOImpl

#### 步骤3 在home.js中完成页面加载时向服务器发送请求，获取产品信息



#### 步骤4.在服务器端查询所有产品信息，并转换成json响应到浏览器



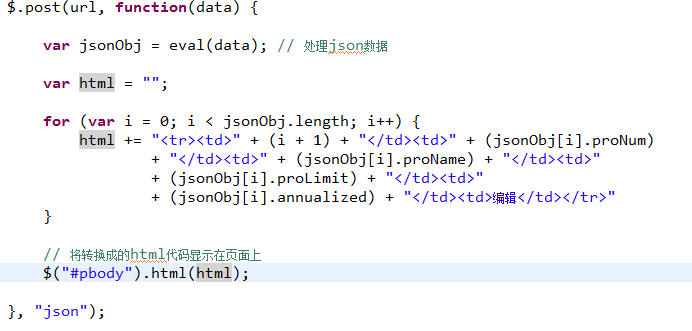


#### 步骤5:在浏览器端处理json数据，展示在页面上

在home.jsp页面上添加一个



将json数据处理后转换成的html代码显示在pbody中就可以。



### 产品管理-产品添加

产品添加思路:

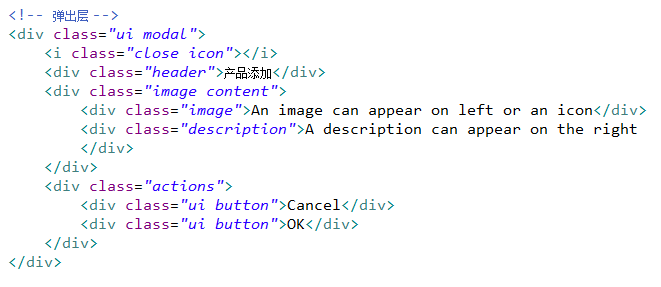
在home.jsp页面上点击添加按钮，弹出一个窗口，在窗口中录入相关产品信息，将信息发送到服务器端，在servlet中调用service,dao,完成将数据插入到product表中，最后我们要在调用查询所有产品操作，将添加的产品显示出来。

#### 步骤1：在home.jsp页面上点击添加按钮弹出窗口

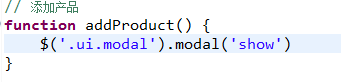
1.修改home.jsp页面上的按钮



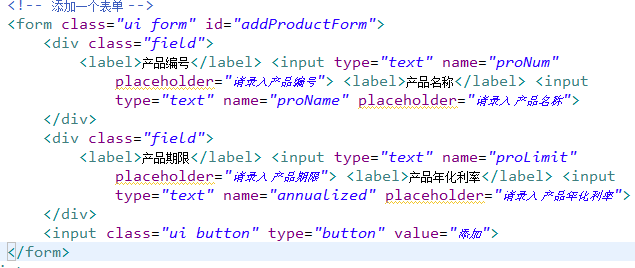
2.在home.jsp中添加弹出层



3.在home.js中添加js代码



4.在弹出的窗口中创建一个表单

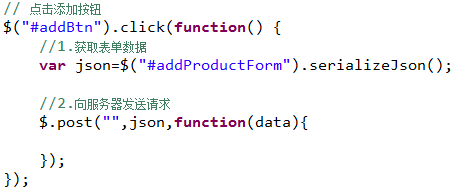


#### 步骤2：添加产品信息，将信息发送到服务器端完成添加操作

1.在home.js中完成点击添加按钮，获取添加的表单的数据

我们可以使用前面讲过的关于jquery插件。





#### 步骤3 在服务器端完成添加操作

使用beanutils来完成请求参数封装

1.导入jar包



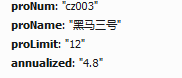
2.使用beanutils将请求参数封装到bean对象中



参数1 是bean对象

参数2 是我们的请求参数的Map集合，一般我们使用时都是使用request.getParameterMap()方法获取。

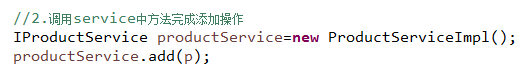
注意:要想将请求参数封装到bean对象中，必须让bean属性与请求参数的名称对应。



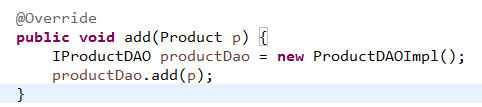
这些参数与Product类中的bean属性名称相同

完成服务器端添加操作

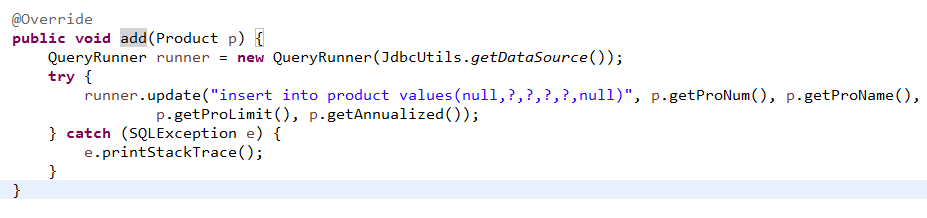
Servlet



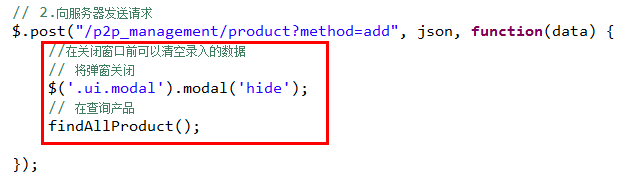
Service



Dao



#### 步骤4 添加完成后，在查询产品在页面上显示



### 产品管理-产品修改

对于修改操作就是进行查询，回显示数据后在修改。

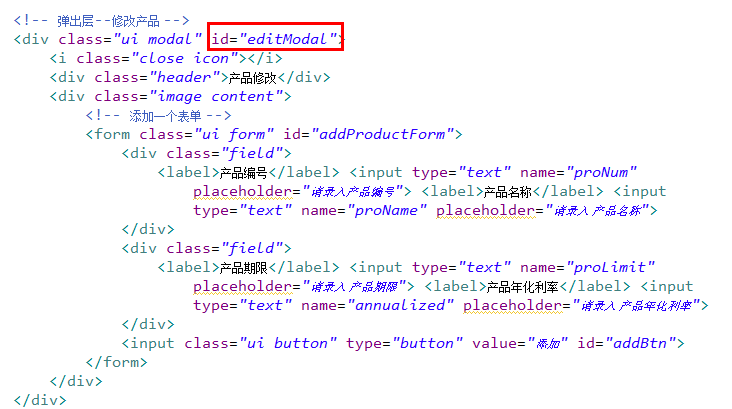
#### 步骤1:给home.jsp页面上的编辑添加连接，点击进行产品的查询

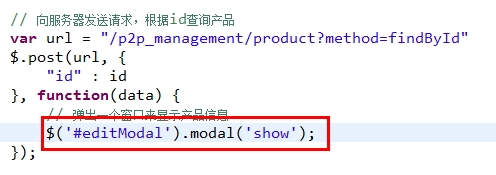
给编辑添加连接



创建了一个findById函数来完成根据产品id查询产品操作

注意:点击编辑后要弹出一个窗口

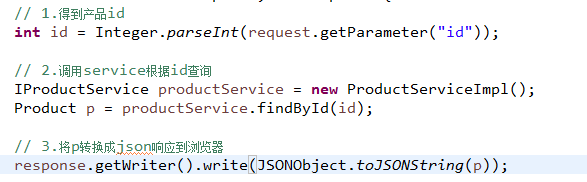




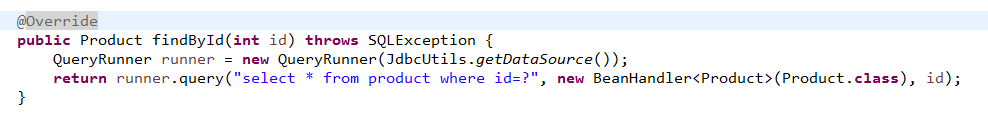
注意:要对添加操作中的弹窗进行修改，添加一个id进行操作

#### 步骤2 在服务器端完成根据id查询产品操作

Servlet

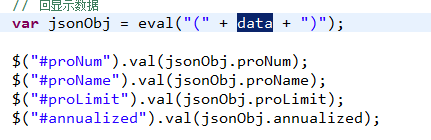


Dao



#### 步骤3 在home.jsp页面上显示查询到的产品信息

对每一个文本框添加id



#### 步骤4 修改操作

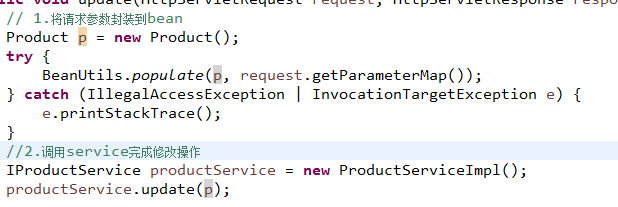
浏览器端操作



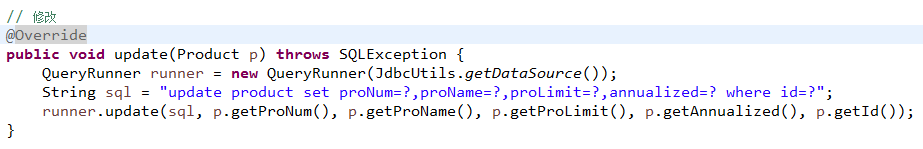
注意:参数中一定要携带id

服务器端操作

Servlet



dao



## p2p\_home功能实现

### 用户管理-注册操作

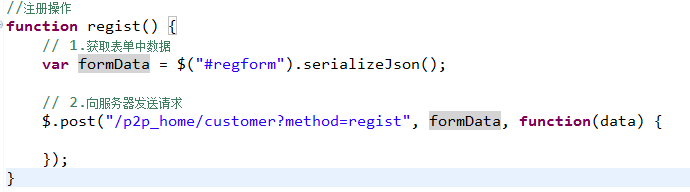
注册操作它不仅是添加客户信息，还要在注册时，为客户生成一个帐户。简单说，我们在进行客户注册操作时，要insert两条记录,一条是customer表中添加数据，代表的是客户信息，还需要向account表中插入数据，当创建一个客户时，也会为其生成一条帐户信息。

（本项中，没有使用支付宝，微信操作）去进行转账操作，只是简单的自己生成了帐户信息。

#### 步骤1:修改register.html页面，当点击注册按钮时向服务器发送请求

1.修改手机号为邮箱地址

2.给注册按钮添加事件，点击它完成向服务器发送请求操作



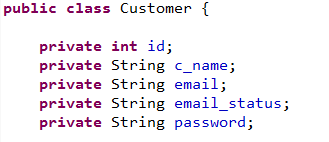
#### 步骤2:服务器端完成注册操作

注意：注册时要向customer与account表中插入数据，我们需要进行事务控制。

Servlet创建

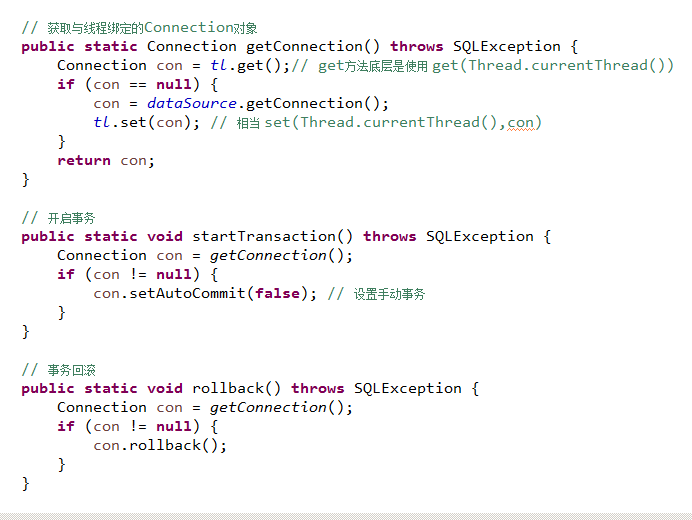


Domain



#### 步骤3:完成添加客户操作

关于JdbcUtils工具



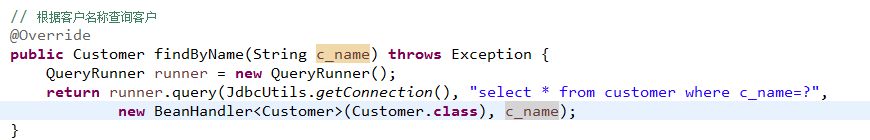
Service操作



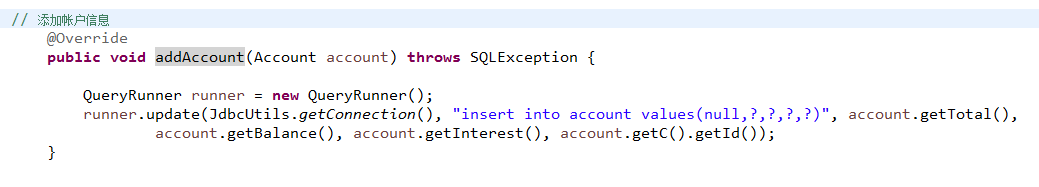
Dao操作

#### 步骤4:完成添加帐户操作

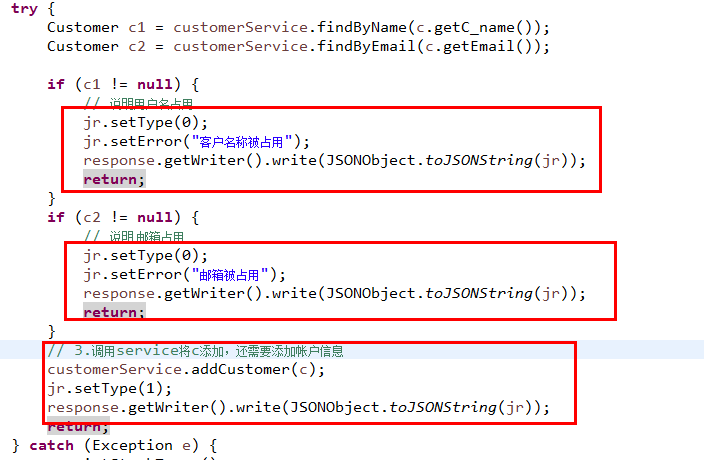
注意：添加客户时要保证它的email与c\_name是唯一的。



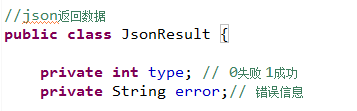
添加帐户信息

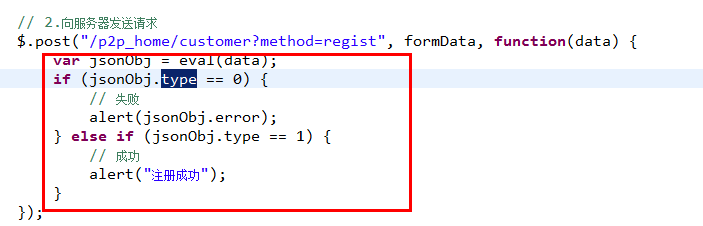


#### 步骤5:判断客户名称与邮箱唯一 ，并响应数据到浏览器



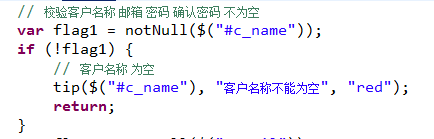
为了方便在浏览器端判断是否成功，我们可以设置一个json类型的返回结果。

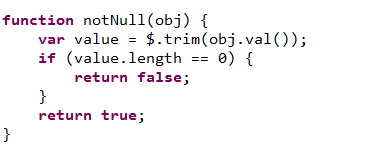




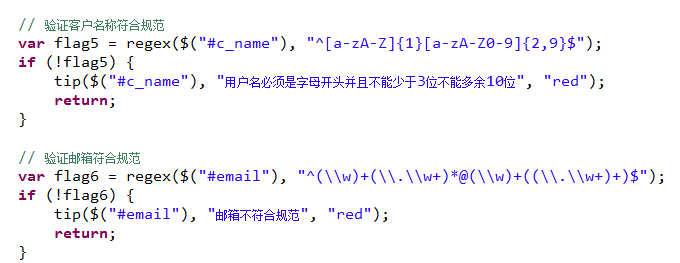
#### 步骤6:解决前端js校验

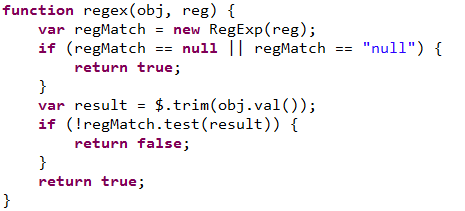
非空校验





规则校验



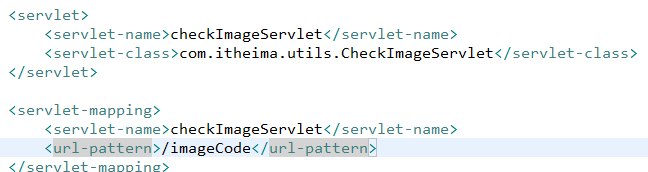


### 用户管理-登录操作

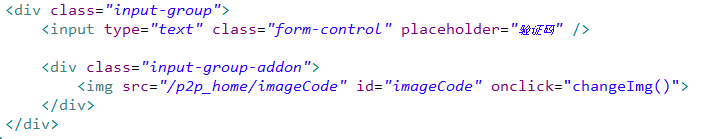
注意：在登录时，我们录制客户名称或邮箱地址都可以完成登录。

#### 步骤1：在登录页面上显示图片验证码

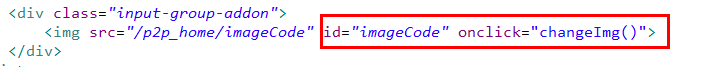
将资料中的CheckImageServlet复制到utils包下在web.xml文件中配置

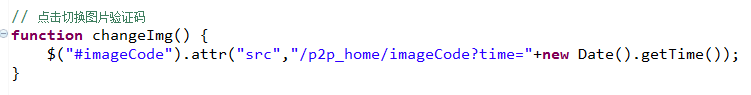


在login.html页面上添加



点击图片验证码切换图片

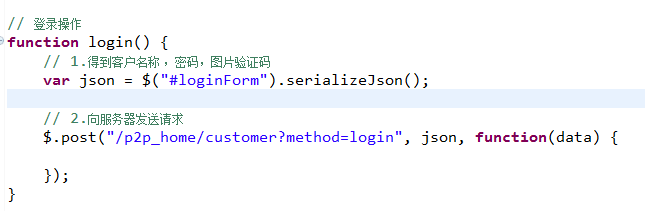




#### 步骤2：完成点击立即登录操作，将信息发送到服务器端

注意:在页面上需要给用户名密码验证码添加name属性



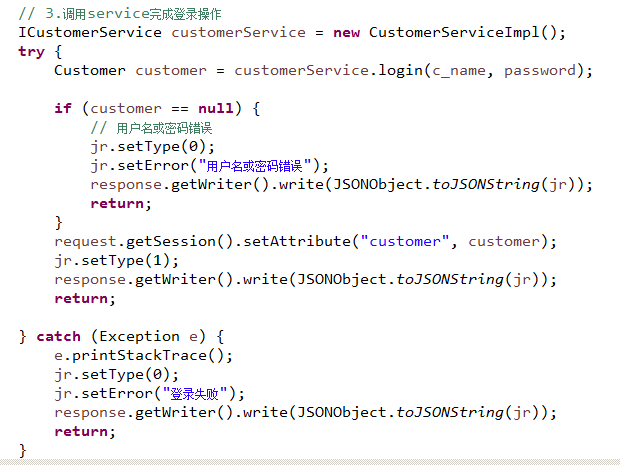


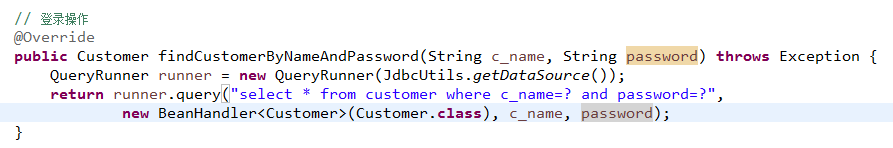
#### 步骤3：在服务器端完成登录操作，并响应数据到浏览器端

获取请求参数，判断验证码是否正确

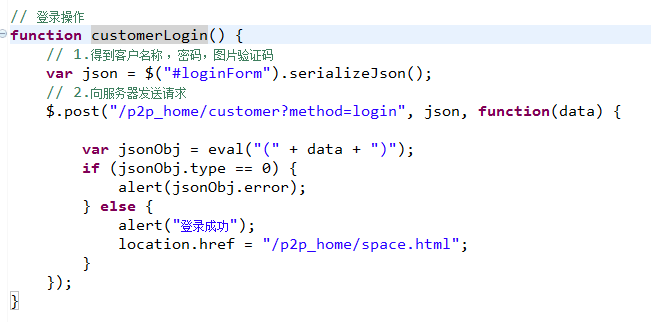


调用service完成登录操作





在浏览器端处理服务器的响应数据



注意:在login.html页面上我们进行了修改



#### 步骤4：完成邮箱登录

在登录程序中添加一段来判断是否是邮箱



#### 步骤5:在服务器端进行校验



#### 步骤6:关于密码md5加密问题

Java加密

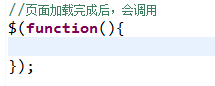


在mysql中对密码进行加密

UPDATE customer SET PASSWORD=MD5('123');

#### 步骤7 登录后，跳转到space.html页面加载客户帐户信息

给space.html页面中导入一个space.js文件，在space.js文件中

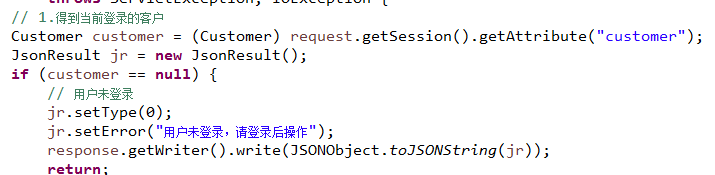


这个函数是页面加载完成后执行。我们就可以在函数中向服务器发送请求，来获取客户的帐户信息在页面上显示

1.向服务器发送请求



2.在服务器端判断客户是否登录



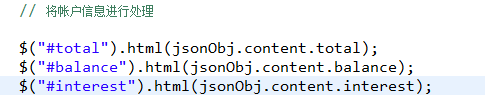
3.客户登录，查询客户的帐户信息



注意：在JsonResult中双封装了一个content信息，它主要是用于封装查询的对象，例如Account对象。

4.在页面上处理帐户信息显示

首先在space.html页面上添加了id,在js代码中进行了处理



### 用户管理-邮箱认证

邮箱认证主要就是修改customer表中的email\_status字段值修改为1

第一部分:点击”认证”弹出窗口，在窗口中显示用户注册时的邮箱，可以在点击按钮向服务器发送请求，服务器端要向邮箱发送一封邮件。

第二部分:完成邮箱认证操作

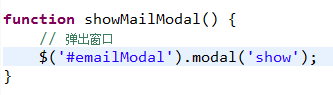
#### 步骤1:点击”认证”弹出窗口

在space.html页面上添加了模态框代码

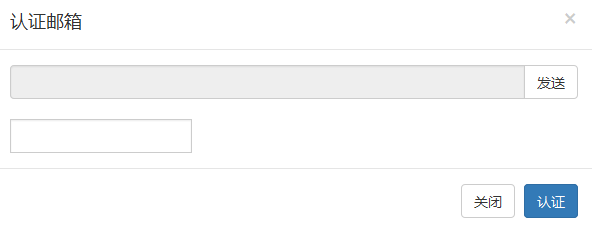
在”认证”上添加

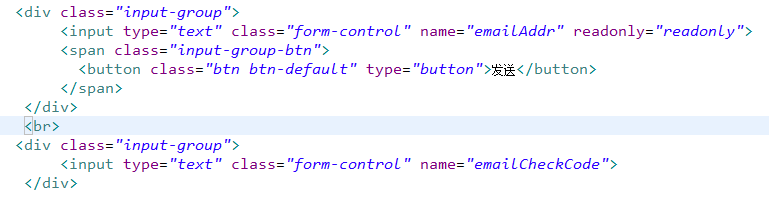


在space.js文件中



接下来我们修改模态框

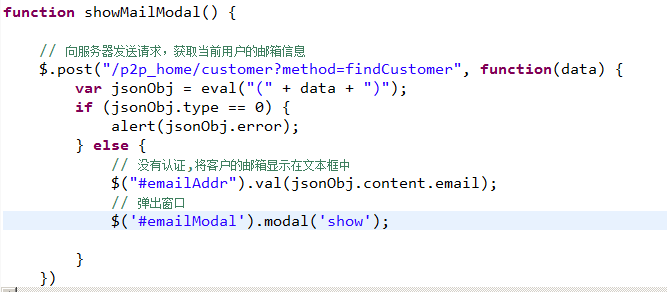




#### 步骤2:完成服务器端客户信息查询，在文本框中显示客户的邮箱地址

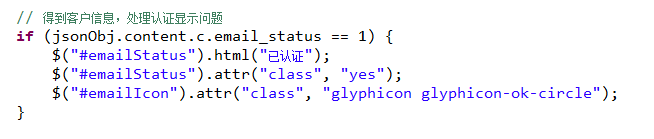


浏览器端处理

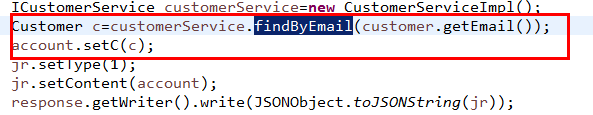


关于客户邮箱认证时显示问题处理

在整个space.html页面加载后，处理邮箱认证显示

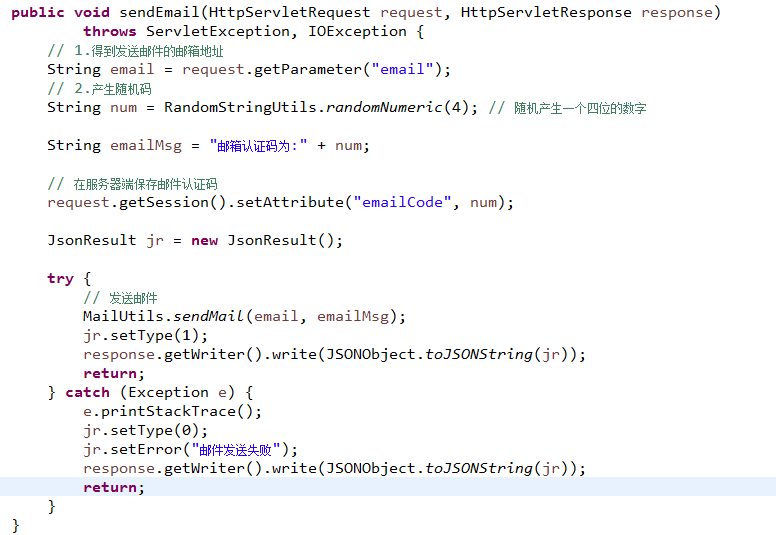


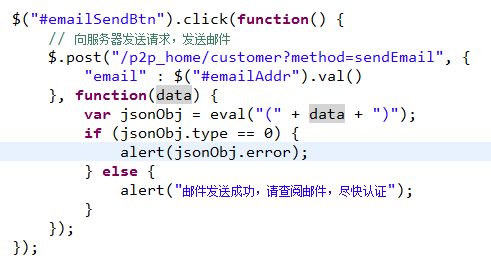
注意：服务器端在查询帐户信息时，将客户信息也封装到Account对象中。



#### 步骤3:完成邮件的发送操作

服务器端发送邮件

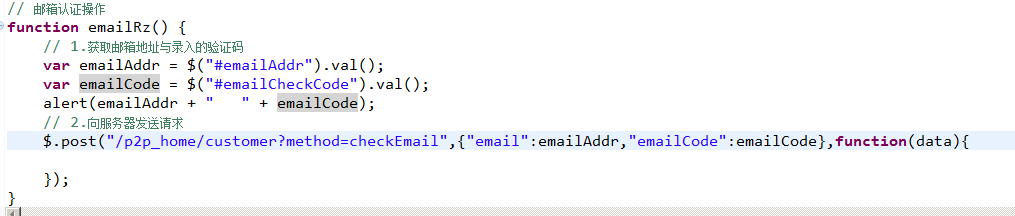


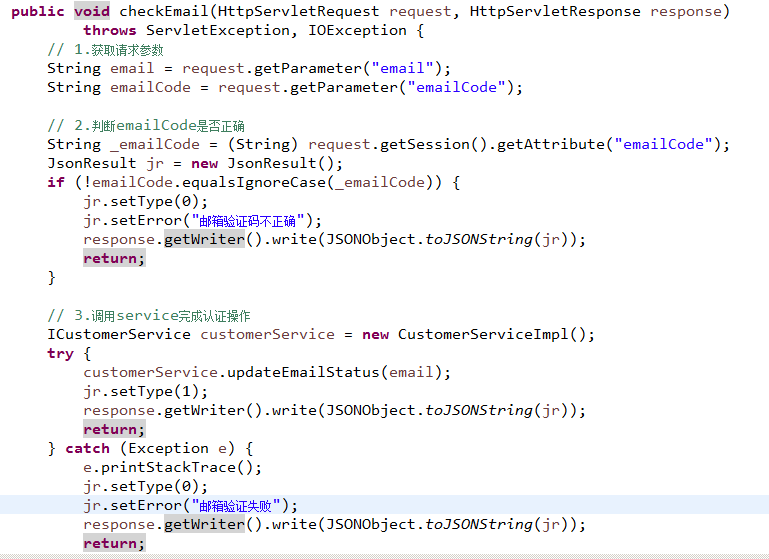


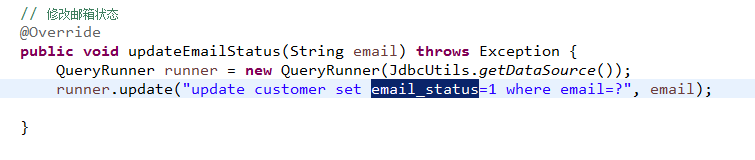
#### 步骤4：邮箱认证

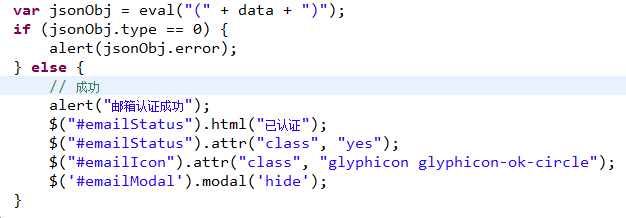
主要就是修改customer表中的email\_status字段值为1.

前台给“认证”按钮添加事件，点击它向服务器发送请求，携带着邮箱与录入验证码，在服务器端完成认证操作。









### 帐户中心-产品购买

产品购买，我们首先要对产品进行查询。我们在p2p\_home中要查询产品信息，在p2p\_management工程中已经提供了产品的crud操作，会使用跨域来完成数据查询。可以使用jquery提供的getJson操作

对于产品的购买，我们怎样操作?

1.用户的帐户表中的信息一定要改变

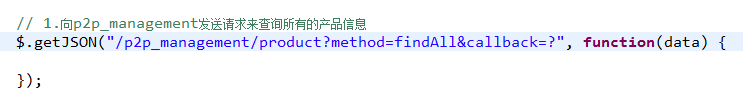
2.product\_account表，这张表中记录的是客户的投资信息。

注意：在product\_account 表中我们还需要添加两个字段，一个是money代表投资金额 ，还需要记录本次交易收益interest

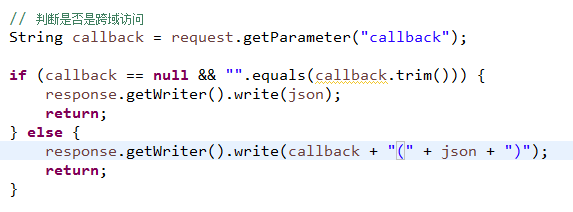


#### 步骤1：跨域查询产品信息在moto.html页面上展示产品

在moto.js中完成向p2p\_management工程发送请求，查询所有产品信息



在p2p\_management的ProductServlet的findAll方法中修改



#### 步骤2：在moto.html页面上展示产品信息

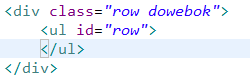
我们使用一个jquery的插件来实现产品信息的滚动展示。

1.导入自动滚动jquery插件相关的css与js文件

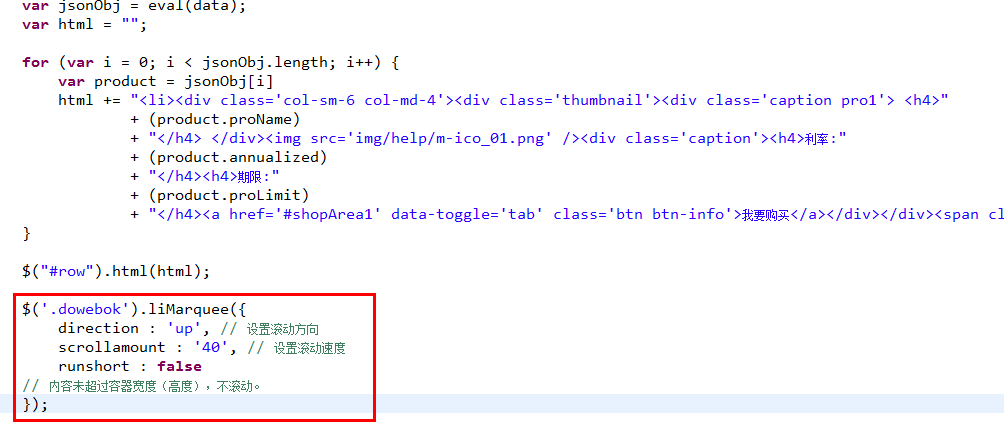




2.在页面上进行修改<li> <ul>

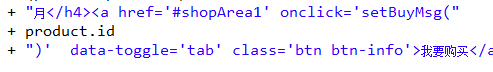


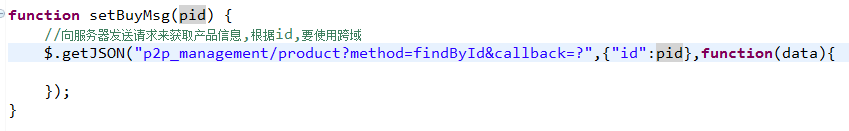
3.在moto.js文件中拼装html代码时



#### 步骤3：moto.html页面上产品购买信息生成

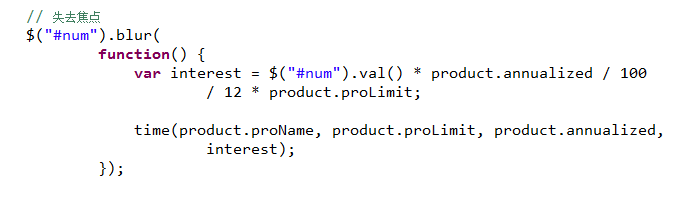
1.点击我要购买时在页页面上展示产品信息





2.在moto.html页面上展示当前要购买的产品的相关数据及计算收益信息

预期收益=本金\*预期收益率/100/12\*limit



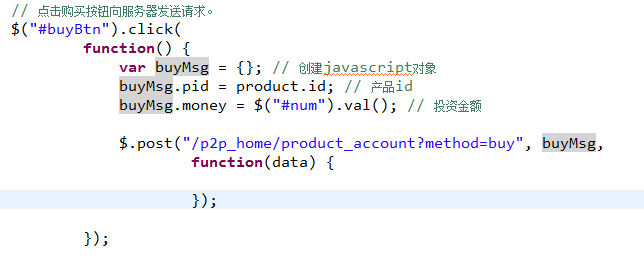


#### 步骤4:完成购买操作

将投资的金额与产品信息传递到服务器，在服务器端进行操作，来完成购买。

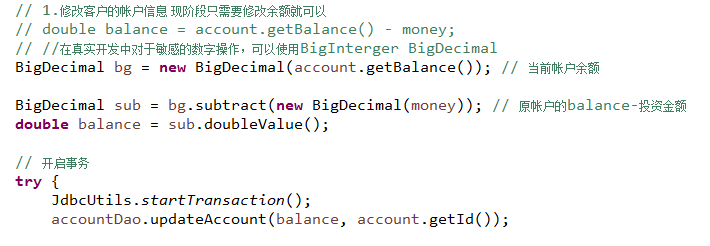
修改帐户信息，向product\_account表中插入数据。  
注意：在操作前，要判断当前用户是否登录，要判断帐户的余额是否充足。

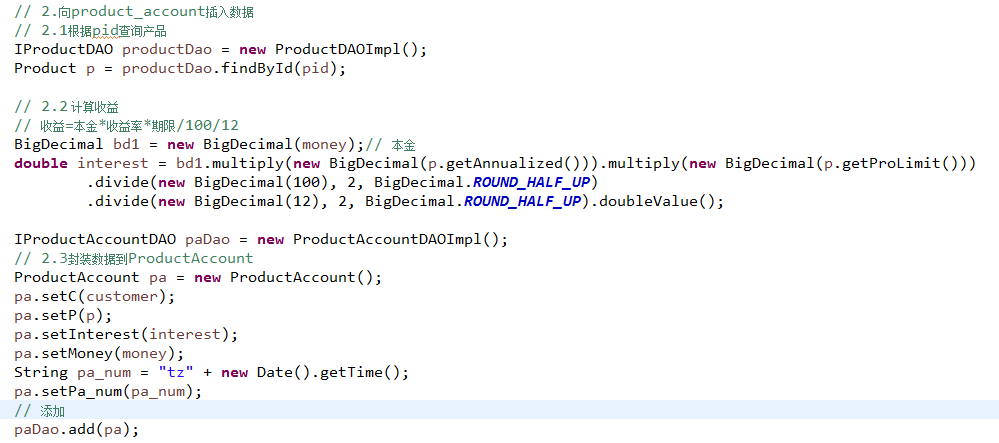
1.向服务器发送我们的投资请求



2.在服务器端完成操作

因为我们需要修改account表中数据，还需要向product\_account表中插入数据，它们是需要进行事务控制的。

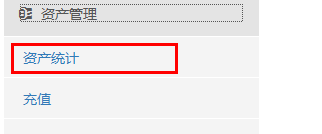




### 帐户中心-产品投资查询

#### 步骤1：产品投资查询分析

对于产品投资查询，我们主要是完成用户资产的信息查询。

、

我们要查询的信息，包括当前客户的帐户信息，购买的理财产品信息。

对于帐户信息，我们要查询account表。

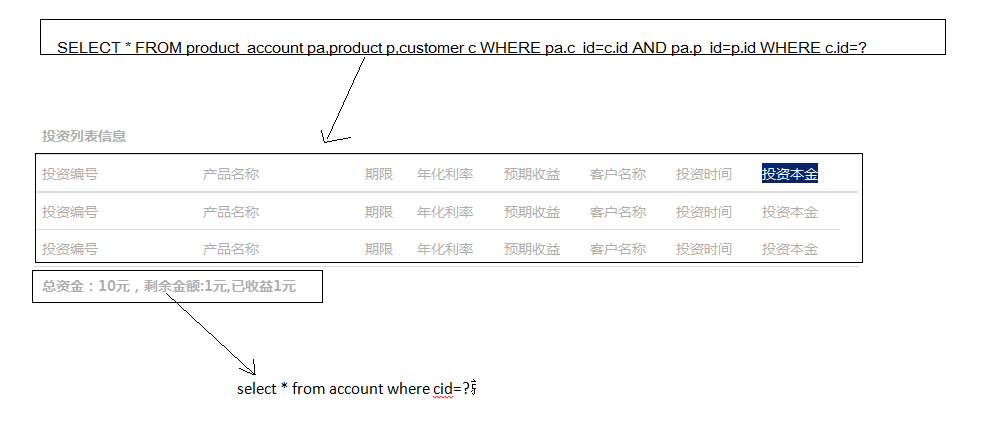
对于购买的产品信息，包括product表 product\_account表 customer表



问题:怎样去完成查询以上信息操作，查询到信息后，怎样响应到浏览器端？

关于帐户信息可以从select \* from account where cid=?就可以查询到。

表格中的数据要从product\_account, customer,product表中查询出来。

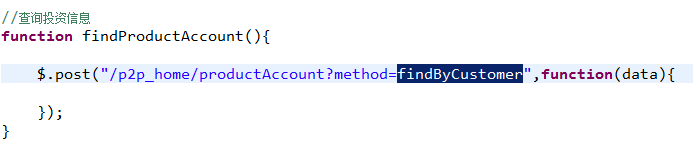


服务器给我们返回的数据应该是json类型。

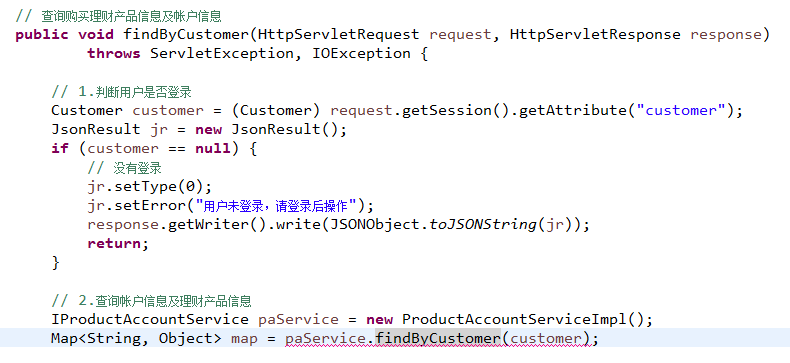
JsonResult来封装返回的数据就可以。JsonResult中定义了一个content对象，它是Object类型。



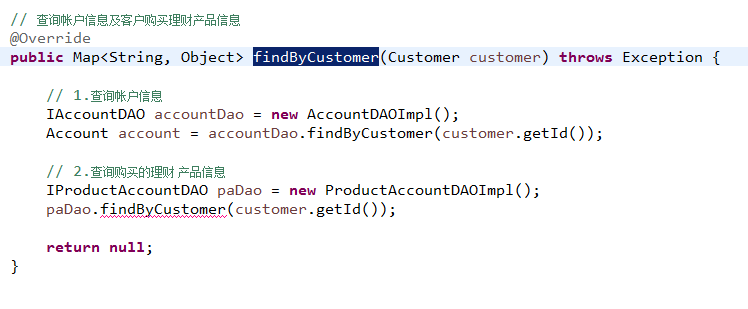
#### 步骤2：前端向服务器发送请求来查询投资信息



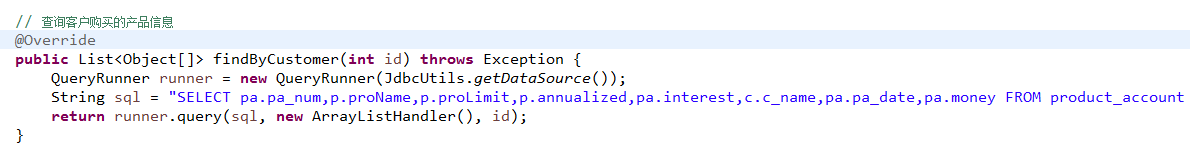
Servlet中操作



Service中操作

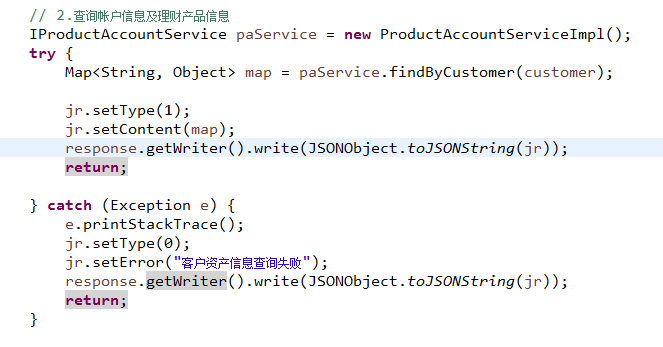


#### 步骤3 在dao中查询购买的产品相关信息

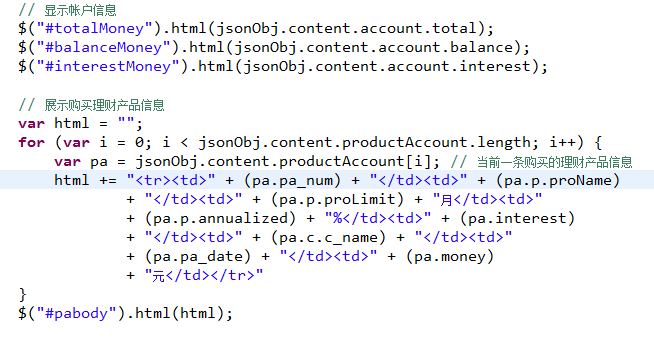


在service中将查询的结果进行了处理





#### 步骤4：在页面上展示投资产品及帐户相关信息



#### 步骤5：展示购买理财产品是否到期，处理页面上时间显示问题

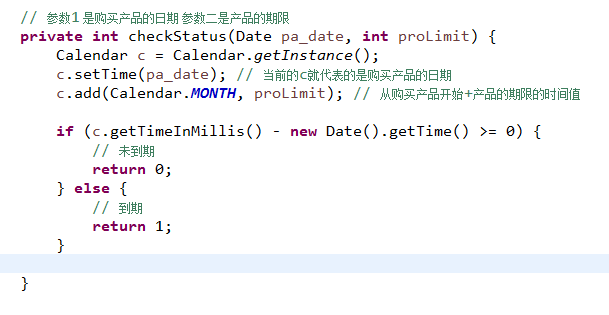
关于时间显示问题,在页面上修改



在页面上展示当前产品是否到期

可以在服务器端完成，在ProductAccount类中添加一个属性。

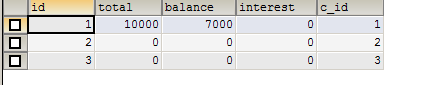






## 项目优化与增强

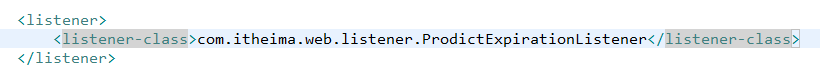
### 监听投资到期



帐户表中的interest它记录的当前用户的总收益情况。

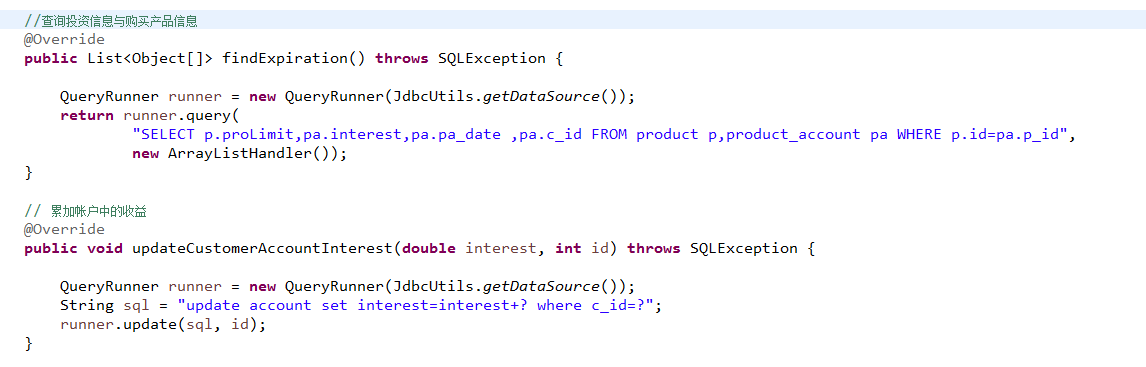
做一个定时处理程序，在每天晚上0点来扫描投资信息，如果当前投资到期了，将这次投资的收益添加到帐户的interest上。会使用到Listener.





具体功能实现





### 架构优化

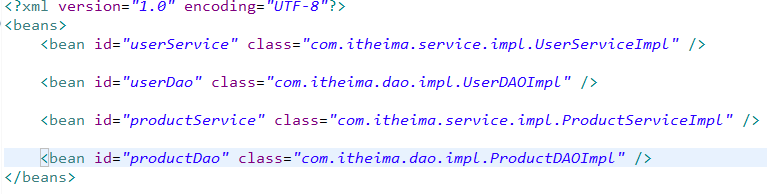
#### Servlet优化



#### Service与dao获取优化

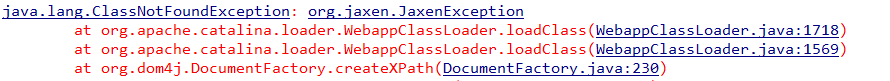
使用工厂模式来完成优化。结合着配置文件来减化操作。

我们的思路:将关于service与dao的相关类都配置在xml文件中，结合java中反射与工厂模式来完成service与dao的获取。





在使用dom4j是报这样一错，



原因是dom4j与xpath语法结合时，需要在单独导入一个jar包



在servlet中要使用service对象，而我们获取bean.xml文件的路径是通过ServletContext对象来获取classes目录



在service中要获取dao对象，而在service中获取bean.xml文件的路径

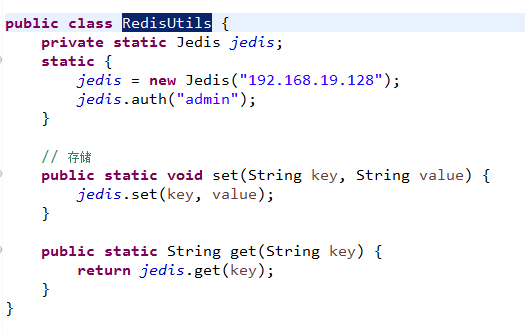


### Redis缓存优化

我们可以将产品信息保存到redis中。以后每一次查询都从redis中获取产品信息。

不用每一次都与数据库交互。

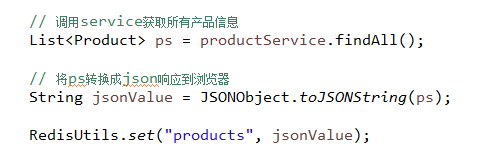
我们可以将关于产品信息的json串存储到redis中，以后从redis中获取就可以。



在查询理财产品时，先从redis中获取，如果没有在从数据库中查询出来，在保存到redis中



当我们的产品修改了或添加了，就需要修改redis中的数据。

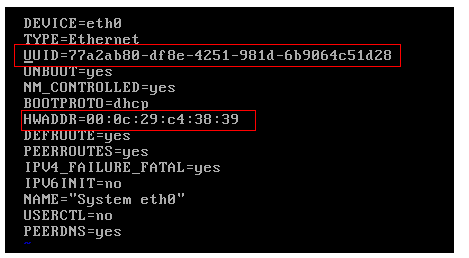


# 项目部署

## 克隆一份linux

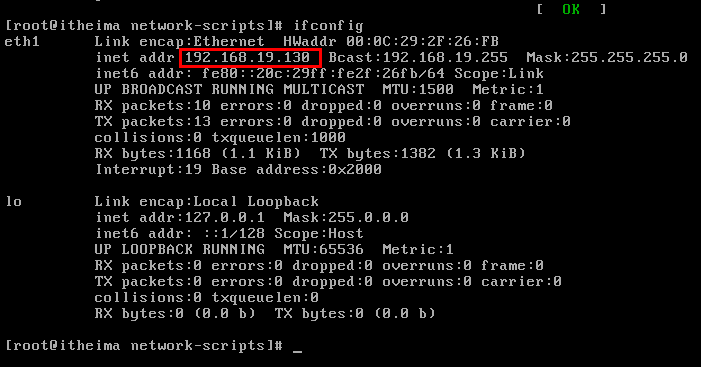
注意：关于克隆这份linux的网络问题。



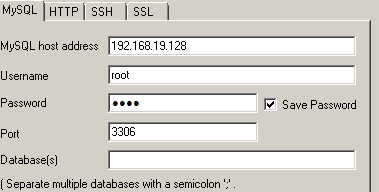


如果修改后还有问题，将DEVICE=eth0修改为DEVICE=eth1

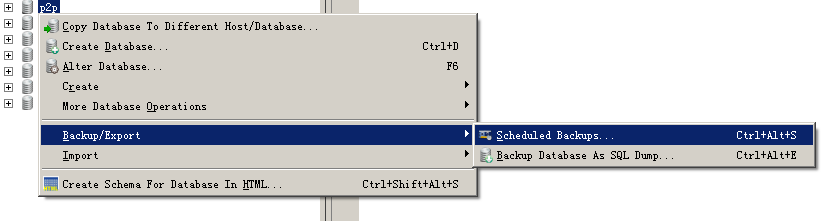
重启网络服务 service network restart



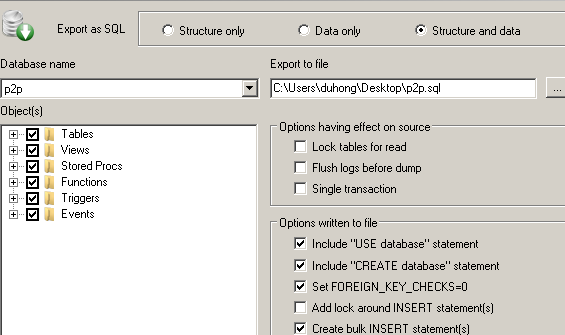
## 将mysql数据库导入，并生成war文件导入到tomcat中



在开发环境 下导出我们的数据库信息



将p2p.sql文件内容在linux服务器的mysql中执行一下。



### 关于项目导出成war文件

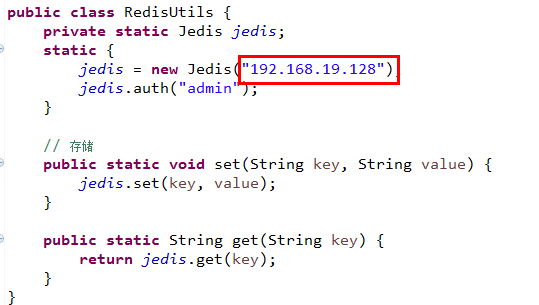
1.c3p0连接池的配置信息需要修改



2.所有的domain下的类都要实现java.io.Serializable



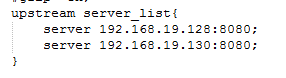
3.需要修改RedisUtils中的redis的服务器ip

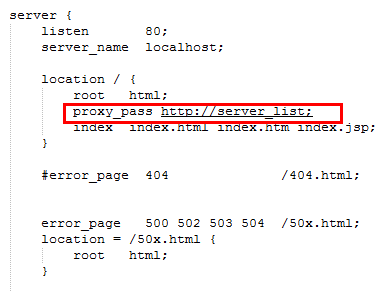


## 使用nginx搭建tomcat集群，并实现session共享

### 配置nginx实现反向代理并实现负载均衡

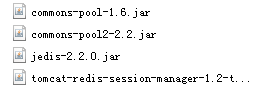
在nginx下的conf目录 中有一个nginx.conf文件，修改这个文件



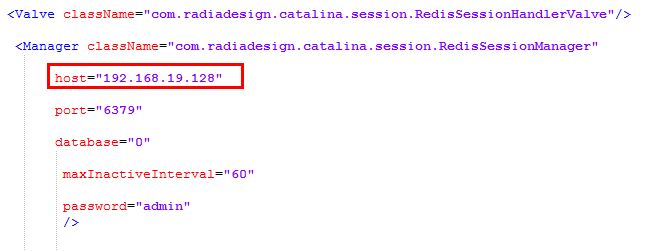


### 实现session共享

1.导入四个jar包到tomcat的lib下



2.修改tomcat/conf/context.xml文件



# 项目实战

## 产品信息分页展示

分页必备条件:

1.当前页码 pageNum

2.每页显示条数 pageSize

3.总条数 totalCount

4.总页数 totalPage

5.当前页要显示的数据List<Product>

问题:浏览器向服务器发送请求，查询当前页数据时，需要携带什么参数，服务器如何完成分页查询数据?

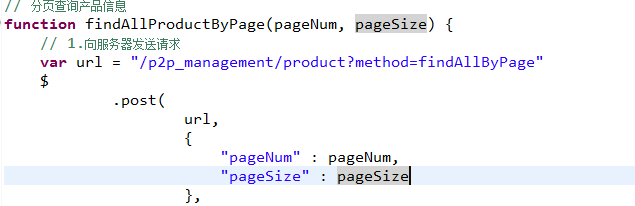
浏览器需要向服务器传递两个参数 pageNum pageSize

服务器查询分页数据可以使用mysql提供limit完成。

Limit使用时第一参数它的值 (pageNum-1)\*pageSize

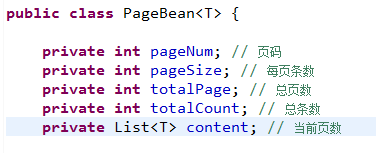
Limit使用时第二参数它的值 pageSize

### 浏览器端操作



### 服务器端操作

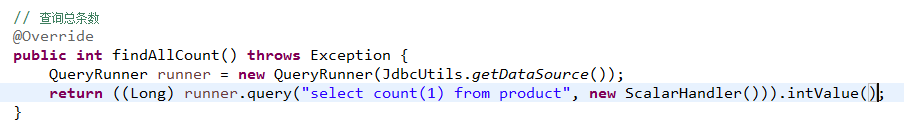
定义一个PageBean用于封装分页相关数据.



Servlet中应该定义一个findAllByPage方法，获取请求参数pageNum与pageSize

Service中获取一个PageBean对象，在pageBean中封装pageNum,pageSize,totalCount,totalPage,content;

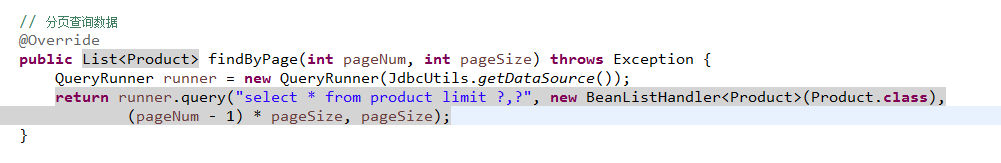
如何获取totalCount? 调用一个dao完成查询总条数操作



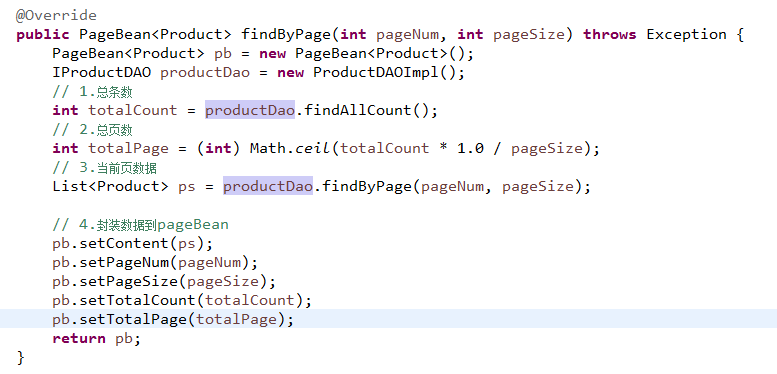
如何获取totalPage? 可以根据总条数与每页条数获取到



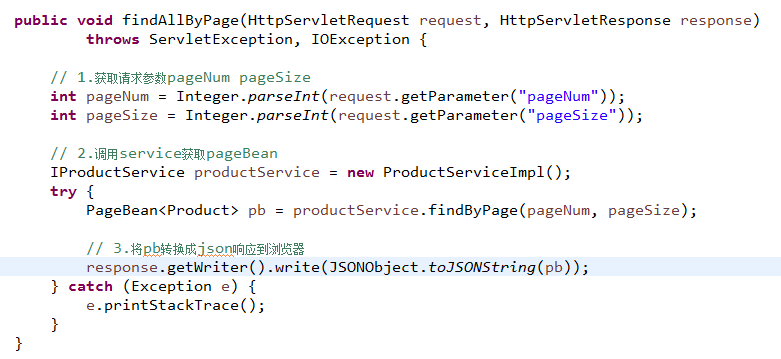
如何获取content? 调用一个dao完成查询当前页数据操作 limit



Service完整操作

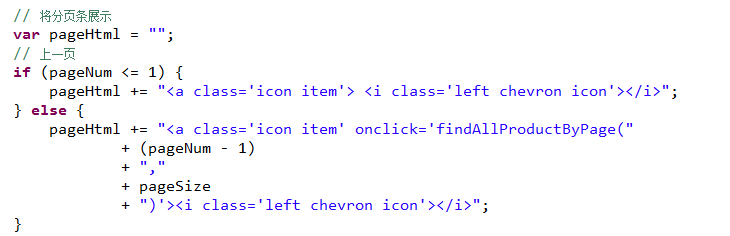


Servlet完整操作

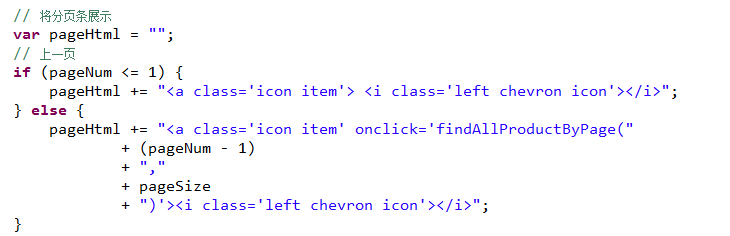


### 浏览器端分页展示

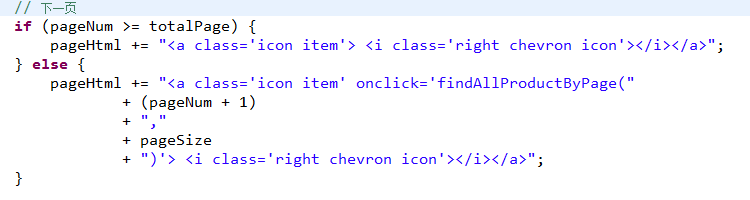
上一页



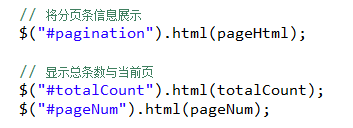
分页条码



下一页



其它信息

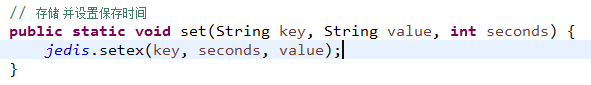


## 邮箱验证码保存到redis

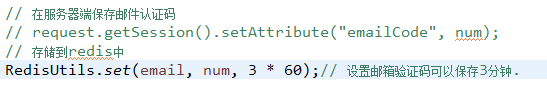
原来是存储在session中，当用户下线，session中的邮箱验证码就不存在了。

我们可以将邮箱验证码保存到redis中，并且设置保存时间。

在redis工具中添加一个set方法，并可以设置保存时间



在CustomerServlet中sendEmail方法内修改



在CustomerServlet中checkEmail方法内修改

