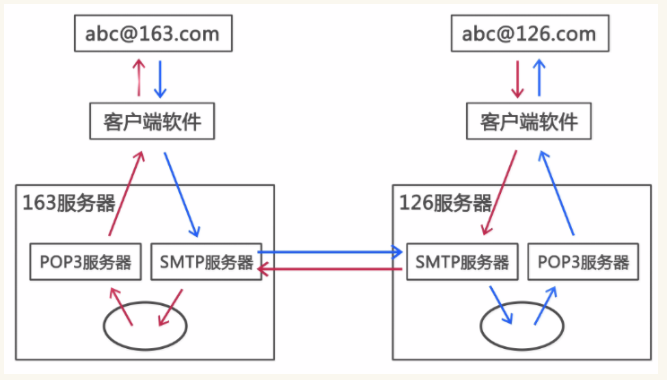
一.注册邮箱激活原理

1.邮箱发送接收的原理：



smtp：smtp协议属于tcp/ip协议簇，它帮助每台计算机在发送或中转信件时找一个目的地。通过smtp协议所指定的服务器，就可以把e-mail寄到收信人的服务器上，默认端口号为25.

pop3：pop3协议也属于tcp/ip协议族，用于客户端从服务器端得到电子邮件。

2.注册激活的原理：

用户表中首先必须有：

激活码字段、是否激活成功字段。

第一步：在注册的时候，首先获取一个激活码，然后把是否成功设置为否，并在发送成功后，通过另一个线程去发送激活邮件。

第二步：点击激活邮件的链接，跳到激活的action，先根据激活码获取到相应的User，然后把激活字段设置为是。

3.java实现邮件发送的原理和代码

|  |
| --- |
| **import java.security.GeneralSecurityException**; **import java.util.Properties**;  **import javax.mail.**Authenticator; **import javax.mail.**Message.**RecipientType**; **import javax.mail.MessagingException**; **import javax.mail.PasswordAuthentication**; **import javax.mail.Session**; **import javax.mail.**Transport; **import javax.mail.internet.AddressException**; **import javax.mail.internet.InternetAddress**; **import javax.mail.internet.MimeMessage**;  **import com.sun.mail.util.MailSSLSocketFactory**; **import shop.user.User**;   **public class MySendMailThread extends Thread** {   **private User** user = **null**;   **public** MySendMailThread(**User** user) {  **this**.user = user;  }   @Override  **public void** run() {   // 跟smtp服务器建立一个连接 新建一个properties文件  **Properties** p = **new** Properties();  // 设置邮件服务器主机名  p.setProperty("mail.host", "smtp.qq.com");// 指定邮件服务器，默认端口 25  // 发送服务器需要身份验证  p.setProperty("mail.smtp.auth", "true");// 要采用指定用户名密码的方式去认证  // 发送邮件协议名称  p.setProperty("mail.transport.protocol", "smtp");   // 开启SSL加密，否则会失败  **MailSSLSocketFactory** sf = **null**;  **try** {  sf = **new** MailSSLSocketFactory();  } **catch** (**GeneralSecurityException** e1) {  e1.printStackTrace();  }  sf.setTrustAllHosts(**true**);  p.put("mail.smtp.ssl.enable", "true");  p.put("mail.smtp.ssl.socketFactory", sf);   // 开启debug调试，以便在控制台查看  // session.setDebug(true);    // 创建session  **Session** session = **Session**.getDefaultInstance(p, **new** Authenticator() {  @Override  **protected PasswordAuthentication** getPasswordAuthentication() {  // 用户名可以用QQ账号也可以用邮箱的别名  **PasswordAuthentication** pa = **new** PasswordAuthentication(  "604460121@qq.com", "rvavanvgftehbfgh");  // 后面的字符是授权码，用qq密码不行！！  **return** pa;  }  });   session.setDebug(**true**);// 设置打开调试状态   //构建一个邮件信息。  **try** {  // 声明一个Message对象(代表一封邮件),从session中创建  **MimeMessage** msg = **new** MimeMessage(session);  // 邮件信息封装  // 1发件人  msg.setFrom(**new** InternetAddress("604460121@qq.com")); //这里面是一个地址对象，因此要用InternetAddress对象来创建。  // 2收件人  msg.setRecipient(**RecipientType**.TO, **new** InternetAddress(user.getEmail()));  // 3邮件内容:主题、内容  msg.setSubject(user.getName() + ",欢迎注册zk账号,请点击链接激活账号");   // StringBuilder是线程不安全的,但是速度快，这里因为只会有这个线程来访问，所以可以用这个  **StringBuilder** sbd = **new** StringBuilder();  sbd.append(user.getName() + "<br/>欢迎！请确认此邮件地址以激活您的账号。<br/>");  sbd.append("<font color='red'><a href='http://192.168.1.104:8080/ssh/user\_active.action?code="  + user.getCode() + "' target='\_blank'");  sbd.append(">立即激活</a></font><br/>");  sbd.append("或者点击下面链接:<br/>");  sbd.append("http://192.168.1.104:8080/shop/user\_active.action?code="  + user.getCode() + "<br/>");  sbd.append("这是一封自动发送的邮件；如果您并未要求但收到这封信件，您不需要进行任何操作。");   msg.setContent(sbd.toString(), "text/html;charset=utf-8");// 发html格式的文本 设置发送的格式。   // 发送动作  Transport.send(msg);   **System**.out.println("给" + user.getEmail() + "发送邮件成功。");  } **catch** (**AddressException** e) {  e.printStackTrace();  } **catch** (**MessagingException** e) {  e.printStackTrace();  }  } } |

发送邮件需要：

发件人，收件人，服务器验证，邮件本体。

第一步：

新建一个properties文件。并设置相应的参数。

|  |
| --- |
| **Properties** p = **new Properties**(); // 设置邮件服务器主机名 p.setProperty("mail.host", "smtp.qq.com");// 指定邮件服务器，默认端口 25 // 发送服务器需要身份验证 p.setProperty("mail.smtp.auth", "true");// 要采用指定用户名密码的方式去认证 // 发送邮件协议名称 p.setProperty("mail.transport.protocol", "smtp"); // 开启SSL加密，否则会失败 **MailSSLSocketFactory** sf = **null**; **try** {  sf = **new** MailSSLSocketFactory(); } **catch** (**GeneralSecurityException** e1) {  e1.printStackTrace(); } sf.setTrustAllHosts(**true**); */\* \* 1、参数类型。setproperty方法的参数类型都只能是string的字符串类型，而put方法的参数为(object key, object value)，类型是object！  2、空值。hashmap的value参数可以设置空值null，而当setproperty的value参数为null时，会抛出空指针异常nullpointerexception \* \*/* p.put("mail.smtp.ssl.enable", "true"); p.put("mail.smtp.ssl.socketFactory", sf); |

第二步：用相应的properties文件，创建邮箱的session，并添加服务器验证。

|  |
| --- |
| **Session** session = **Session**.getDefaultInstance(p, **new** Authenticator() {  @Override  **protected PasswordAuthentication** getPasswordAuthentication() {  // 用户名可以用QQ账号也可以用邮箱的别名  **PasswordAuthentication** pa = **new** PasswordAuthentication(  "604460121@qq.com", "rvavanvgftehbfgh");  // 后面的字符是授权码，用qq密码不行！！  **return** pa;  } });  session.setDebug(**true**);// 设置打开调试状态 |

第三步：根据session构建邮件。

|  |
| --- |
| //构建一个邮件信息。 **try** {  // 声明一个Message对象(代表一封邮件),从session中创建  **MimeMessage** msg = **new** MimeMessage(session);  // 邮件信息封装  // 1发件人  msg.setFrom(**new** InternetAddress("604460121@qq.com")); //这里面是一个地址对象，因此要用InternetAddress对象来创建。  // 2收件人  msg.setRecipient(**RecipientType**.TO, **new** InternetAddress(user.getEmail()));  // 3邮件内容:主题、内容  msg.setSubject(user.getName() + ",欢迎注册zk账号,请点击链接激活账号");   // StringBuilder是线程不安全的,但是速度快，这里因为只会有这个线程来访问，所以可以用这个  **StringBuilder** sbd = **new** StringBuilder();  sbd.append(user.getName() + "<br/>欢迎！请确认此邮件地址以激活您的账号。<br/>");  sbd.append("<font color='red'><a href='http://192.168.1.104:8080/ssh/user\_active.action?code="  + user.getCode() + "' target='\_blank'");  sbd.append(">立即激活</a></font><br/>");  sbd.append("或者点击下面链接:<br/>");  sbd.append("http://192.168.1.104:8080/shop/user\_active.action?code="  + user.getCode() + "<br/>");  sbd.append("这是一封自动发送的邮件；如果您并未要求但收到这封信件，您不需要进行任何操作。");   msg.setContent(sbd.toString(), "text/html;charset=utf-8");// 发html格式的文本 设置发送的格式。  // 发送动作  Transport.send(msg); //通过这个调用smtp协议，发送邮件。   **System**.out.println("给" + user.getEmail() + "发送邮件成功。"); } **catch** (**AddressException** e) {  e.printStackTrace(); } **catch** (**MessagingException** e) {  e.printStackTrace(); } |

效果：  
<http://192.168.1.104:8080/shop/user_active.action?code=a7b1d1a806f64da69128f78d2ddc0fea>这是一封自动发送的邮件；如果您并未要求但收到这封信件，您不需要进行任何操作。

二. struts2中接受参数的总结：

2.1.一般调用

2.1.1封装参数：

如果action没有继承ModelDriven<T>,那么提交到action的url上，输入&username=username，由于Struts2中有一个参数拦截器，只要在后台程序中写

private UserName username;再设置它的set方法，就可以得到username参数，并存入值栈中。

但是，要注意的是：如果你需要在前台取的这个数据，那么必须再设置get方法，然后前台取值方式： 直接用属性名就好了。

<**s:property** value="username"/>

2.1.2 封装到model中：

后台程序：

**public class ProductAction extends ActionSupport implements** ModelDriven<**Product**> {  
 //必须new出一个对象  
 **private Product** product=**new** Product();  
 @Override  
 **public Product** getModel() {  
 **return** product;  
 }

**public String** findByPid() {  
 product = productService.findByPid(product.getPid());  
 **return** "findByPidSuccess";  
}

Struts2中有一个model拦截器，只要你在提交到action的url上，输入&username=username;

那么只要action类继承了ModelDriven<T>, //必须new出一个对象

private Product product=new Product();且username这个字段存在于Product这个javaBean中。那么就会把这个参数自动封装到javabean中。或者通过form表单提交也会自动把字段封装到javaBean中。且把javaBean以model为名压入到值栈中了。

如果要取得当前product中的pid。

前台程序这样取值：

<**s:property** value="model.pid"/>

2.2.ajax调用

但是如果用ajax.post 调用action的话，参数就只能封装在js的对象中了，这样拦截器是不会自动封装数据.

**var** ajaxParm={};  
ajaxParm["userName"]=username;

**$**.post("**${**pageContext.request.contextPath**}**/user\_checkUserName.action?"+**new Date**().getTime(),ajaxParm,**function**(data){  
 **var** msg=data;  
 **$**("#span1").html(msg); //jquery给赋值html字符串是通过这种方式  
})

这样在后台取值，必须用

**String** userName=request.getParameter("userName");

但是，如果不采用ajaxParm的方式，而是直接用

{'username':username}

来代替的话，那么就可以直接被参数拦截器拦截下来。

原因：未知！！

三.验证码的实现：



3.1 前台页面：

|  |
| --- |
| <**input** type="text"  id="checkcode" name="checkcode" class="text captcha"  maxlength="4" autocomplete="off"/> <**img** id="checkImg" class="captchaImage"  src="**${** pageContext.request.contextPath **}**/checkImg.action"  title="点击更换验证码" onclick="*change*()"/>  src中调用checkImg.action  点击更换验证码：  **function** *change*(){  $("#checkImg").attr("src","**${**pageContext.request.contextPath**}**/checkImg.action?"+**new Date**().getTime()); **}** |

3.2 后台action程序：

**import java.awt.Color**;  
**import java.awt.Font**;  
**import java.awt.**Graphics;  
**import java.awt.**Graphics2D;  
**import java.awt.image.BufferedImage**;  
**import java.util.Random**;  
  
**import javax.imageio.ImageIO**;  
  
**import org.apache.struts2.ServletActionContext**;  
  
**import com.opensymphony.xwork2.ActionSupport**;  
  
**public class CheckImgAction extends ActionSupport** {  
  
 @Override  
 **public String** execute() **throws Exception** {  
 **int** width = 120;  
 **int** height = 30;  
  
 // 步骤一 绘制一张内存中图片  
 **BufferedImage** bufferedImage = **new** BufferedImage(width, height,  
 **BufferedImage**.TYPE\_INT\_RGB);  
  
 // 步骤二 图片绘制背景颜色 ---通过绘图对象  
 Graphics graphics = bufferedImage.getGraphics();// 得到画图对象 --- 画笔  
 // 绘制任何图形之前 都必须指定一个颜色  
 graphics.setColor(getRandColor(200, 250));  
 graphics.fillRect(0, 0, width, height);  
  
 // 步骤三 绘制边框  
 graphics.setColor(**Color**.WHITE);  
 graphics.drawRect(0, 0, width - 1, height - 1);  
  
 // 步骤四 四个随机数字  
 Graphics2D graphics2d = (Graphics2D) graphics;  
 // 设置输出字体  
 graphics2d.setFont(**new** Font("宋体", **Font**.BOLD, 18));  
  
 **String** words = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890";  
 **Random** random = **new** Random();// 生成随机数  
 // 定义StringBuffer  
 **StringBuffer** sb = **new** StringBuffer();  
 // 定义x坐标  
 **int** x = 10;  
 **for** (**int** i = 0; i < 4; i++) {  
 // 随机颜色  
 graphics2d.setColor(**new** Color(20 + random.nextInt(110), 20 + random  
 .nextInt(110), 20 + random.nextInt(110)));  
 // 旋转 -30 --- 30度  
 **int** jiaodu = random.nextInt(60) - 30;  
 // 换算弧度  
 **double** theta = jiaodu \* **Math**.PI / 180;  
  
 // 生成一个随机数字  
 **int** index = random.nextInt(words.length()); // 生成随机数 0 到 length - 1  
 // 获得字母数字  
 **char** c = words.charAt(index);  
 sb.append(c);  
 // 将c 输出到图片  
 graphics2d.rotate(theta, x, 20);  
 graphics2d.drawString(**String**.valueOf(c), x, 20);  
 graphics2d.rotate(-theta, x, 20);  
 x += 30;  
 }  
  
 // 将生成的字母存入到session中  
 **ServletActionContext**.getRequest().getSession()  
 .setAttribute("checkcode", sb.toString());  
  
 // 步骤五 绘制干扰线  
 graphics.setColor(getRandColor(160, 200));  
 **int** x1;  
 **int** x2;  
 **int** y1;  
 **int** y2;  
 **for** (**int** i = 0; i < 30; i++) {  
 x1 = random.nextInt(width);  
 x2 = random.nextInt(12);  
 y1 = random.nextInt(height);  
 y2 = random.nextInt(12);  
 graphics.drawLine(x1, y1, x1 + x2, x2 + y2);  
 }  
  
 // 将上面图片输出到浏览器 ImageIO  
 graphics.dispose();// 释放资源  
 **ImageIO**.write(bufferedImage, "jpg", **ServletActionContext**.getResponse()  
 .getOutputStream()); //这里是最重要的部分，把图片输出到浏览器。  
 **return** NONE;  
 }  
  
 /\*\*  
 \* 取其某一范围的color  
 \*   
 \* @param fc  
 \* int 范围参数1  
 \* @param bc  
 \* int 范围参数2  
 \* @return Color  
 \*/  
 **private Color** getRandColor(**int** fc, **int** bc) {  
 // 取其随机颜色  
 **Random** random = **new** Random();  
 **if** (fc > 255) {  
 fc = 255;  
 }  
 **if** (bc > 255) {  
 bc = 255;  
 }  
 **int** r = fc + random.nextInt(bc - fc);  
 **int** g = fc + random.nextInt(bc - fc);  
 **int** b = fc + random.nextInt(bc - fc);  
 **return new** Color(r, g, b);  
 }  
}

四.用session和cookie保持登录信息不变

需求：



一个网站登录后：

 在关闭浏览器后，再访问这个页面，依旧可以显示登录状态了。

解决方式：

通过aop切面的方式，在访问主页的action前给key=jesession的cookie，加一个时间，使得它不会因浏览器关闭而消失。

代码：

4.1 生成通知类：

@Aspect  
**public class MyAspect** {  
 **private boolean** flag=**true**;  
 @Pointcut("execution(\* shop.index.IndexAction.execute(..))")  
 **private void** myPointcut(){}  
 @Before("myPointcut()")  
 **public void** before(){  
 /\*\*  
 \* 把JSESSIONID 的cookie 添加一个持续时间。  
 \*/  
 **if**(flag){  
 HttpServletResponse response= **ServletActionContext**.getResponse();  
 HttpServletRequest request=**ServletActionContext**.getRequest();  
 **Cookie** c = **new** Cookie("JSESSIONID",request.getSession().getId());  
 c.setMaxAge(3600);//表示3600秒  
 c.setPath("/shop");//路径,/表示根目录,默认是"/项目名"  
 response.addCookie(c);//添加cookie到response中  
 flag=**false**;  
 }  
 }  
}

4.2 在applicationContext.xml中添加aspectJ的配置

<!--Spring的aop使用-->  
<**aop:aspectj-autoproxy** proxy-target-class="true" />  
<**bean** name="myAspect" class="shop.index.MyAspect"/>

这样，就实现了。在访问首页前实现aop切面，添加cookie的持续时间。

五.hibernate模版的分页查询：

5.1 后台调用代码：

/\*\*  
 \* @param beginIndex 分页的起始值  
 \* @param pageSize 一页展示的个数  
 \* @param cid  
 \* @return  
 \*/  
**public** List<**Product**> getProductsByPage(**Integer** beginIndex, **int** pageSize, **Integer** cid) {  
 **String hql**="select p from Product p, CategorySecond cs where p.categorySecond.csid=cs.csid " +  
 "and cs.category.cid = ? ";  
 **Object**[] params={cid};  
 **PagingUtil**<**Product**> productPagingUtil=  
 **new** PagingUtil<**Product**>(**hql**,params,beginIndex,pageSize);  
 List<**Product**> productList=**this**.getHibernateTemplate().execute(productPagingUtil);  
 **return** productList;  
}

5.2分页工具类：

**public class PagingUtil**<T> **implements** HibernateCallback<List<T>>{  
  
 **private String** hql;  
 **private Object**[] params;  
 **private int** startIndex;  
 **private int** pageSize;  
 /\*\*  
 \*  
 \* @param hql hql语句  
 \* @param params hql语句中的参数  
 \* @param startIndex 从哪一条开始  
 \* @param pageSize 一页显示多少条  
 \*/  
 **public** PagingUtil(**String** hql, **Object**[] params, **int** startIndex, **int** pageSize) {  
 **this**.hql = hql;  
 **this**.params = params;  
 **this**.startIndex = startIndex;  
 **this**.pageSize = pageSize;  
 }  
 @Override  
 **public** List<T> doInHibernate(Session session) **throws HibernateException**, **SQLException** {  
 //1.执行hql语句  
 Query query=session.createQuery(hql);  
 //2.设置参数  
 **if**(params!=**null**){  
 **for**(**int** i=0;i<params.length;i++){  
 query.setParameter(i,params[i]);  
 }  
 }  
 //3.分页  
 query.setFirstResult(startIndex);  
 query.setMaxResults(pageSize);  
 **return** query.list();  
 }  
}

5.3 分析：

通过hibernate模版类调用HibernateCallback<List<T>> 这个类的实现类。

**this**.getHibernateTemplate().execute(productPagingUtil);

而HibernateCallback<List<T>>本质上就是获取session，然后通过hibernate中的方法进行操作。

六.购物车模块模式。

要把

6.1