# 10.EasyMall商城项目

## 10.1.EasyMall商城项目简介

EasyMall(易买网)商城项目是以java语言为主，Jsp、HTML、css、JavaScript为辅开发的一个电子商城项目。该项目用最传统的技术（Servlet、JSP）实现了现代流行的电商项目中的主要的功能，比如浏览商品、将商品加入购物车、下订单及在线支付等功能。该项目还提供了后台管理功能来管理网站中的数据，比如用户管理、商品管理、订单管理、权限控制、销售榜单下载等功能。该项目采用了MVC软件分层的思想来组织整个软件的结构，还采用了Apache DBUtils来简化持久层的操作。最终实现了一个电商网站应该具有的功能。

**项目涉及技术**

该项目采用的技术包含HTTP协议、Tomcat服务器、静态Web资源开发技术(HTML、CSS、JS)、Java后台开发技术（Servlet、JSP）、JDBC、MySQL数据库技术、过滤器、监听器、软件编程思想实践、在线支付、权限控制等等。

## 10.2.EasyMall注册功能实现

### 10.2.1.项目环境搭建

（1）创建EasyMall工程

（2）配置[www.easymall.com](http://www.easymall.com)虚拟主机, 并将主机配置为默认虚拟主机：

在[tomcat]根目录下创建easymall文件夹，然后修改文件：[tomcat]\conf\server.xml，在Engine标签中配置Host标签，例如：

|  |
| --- |
| <Host name="www.easymall.com" appBase="E:\software\tomcat7.0\easymall">  </Host> |

然后修改Engine标签（104行）

|  |
| --- |
| <Engine name="Catalina" defaultHost="www.easymall.com"> |

最后修改hosts文件（C:\Windows\System32\drivers\etc\hosts），加入如下配置：

|  |
| --- |
| 127.0.0.1 [www.easymall.com](http://www.easymall.com) （注意，中间是制表符Tab键） |

（3）将EasyMall项目部署到[www.easymall.com](http://www.easymall.com)主机中, 并将项目配置缺省 的WEB应用, 并配置主页!

部署WEB应用到虚拟主机中的方式:

方式一: 在[tomcat]/conf/server.xml中easymall主机对应的Host标签内部, 添加一个Context标签：

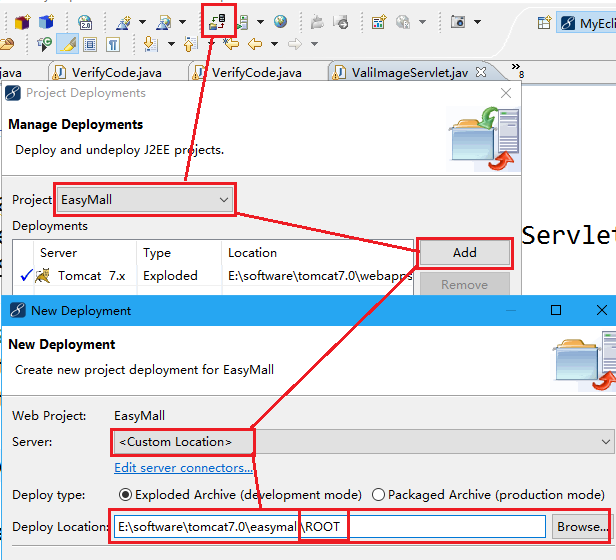
<Context path=”” docBase=”D:\softspace\Workspaces\BIG\_1707\EasyMall\WebRoot” />

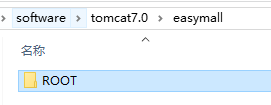
第一种方式不推荐, 每次配置完后都需要重启服务器.

方式二: 在[tomcat]/conf/Catalina/www.easymall.com目录下添加一个ROOT.xml文件, 文件中的配置如下：

<Context docBase=”D:\softspace\Workspaces\BIG\_1707\EasyMall\WebRoot” />

方式三: 直接将WEB应用部署到[www.easymall.com](http://www.easymall.com)主机默认管理的目录下即可!





配置完成后，之间访问www.easymall.com，如果可以进入首页，证明配置正确。

### 10.2.2.导入静态页面

采用jsp展示页面, jsp暂时可以理解为可以写java代码html

1. 可以将index.jsp中其他代码去掉，只保留第一行，编码改为utf-8

例如：

|  |
| --- |
| <%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"utf-8"*%> |

（2）然后复制index.jsp，起名为\_head.jsp、\_foot.jsp、regist.jsp、login.jsp

（3）在课前资料中EasyMall静态页面下，分别找到head、foot、index、login、regist文件夹，其中找到对应的html文件，编辑器打开，全选复制，放入到对应的jsp中，注意不要把jsp中第一行去掉了。同时，将css和img文件夹复制到WebRoot目录下（提示是否覆盖，点击Yes To All）

（4）在index.jsp中包含头部和尾部

|  |
| --- |
| 在body最前面加上如下代码：  <body>  <!-- 将头部引入进来 -->  <%@ include file=*"/\_head.jsp"* %>  在body最后面加上如下代码：  <!-- 将尾部引入进来 -->  <%@ include file=*"/\_foot.jsp"* %>  </body> |

1. 修改\_head.jsp中登录和注册的跳转路径

|  |
| --- |
| <a href=*"*<%= request.getContextPath() %>*/login.jsp"*>登录</a>  <a href=*"*<%= request.getContextPath() %>*/regist.jsp"*>注册</a> |

1. 将所有jsp中引入css或者image的路径全部改为绝对路径，这样更稳定。

修改时只需要在相对路径前面加上“<%= request.getContextPath() %>/”代码，就可以动态获取虚拟路径，不要写死了。

\_head.jsp中需要改3处。

\_foot.jsp中需要改1处。

login.jsp中需要改1处。

regist.jsp中需要改2处。

index.jsp中需要改14处。

最后，访问www.easymall.com，如果可以访问，说明修改无误。

### 10.2.3.实现注册功能-后端

（1）修改regist.jsp中的表单提交的地址

|  |
| --- |
| <form action=*"*<%= request.getContextPath() %>*/servlet/RegistServlet"* method=*"POST"*> |

1. 创建包cn.tedu.web，创建servlet：RegistServlet。
2. **处理请求乱码**

|  |
| --- |
| //1.处理请求参数问题（post）  request.setCharacterEncoding("utf-8");  //处理响应正文乱码  response.setContentType("text/html;charset=utf-8"); |

1. **获取请求参数**

|  |
| --- |
| //2.获取请求参数（用户提交的注册数据）  String username = request.getParameter("username");  String password = request.getParameter("password");  String password2 = request.getParameter("password2");  String nickname = request.getParameter("nickname");  String email = request.getParameter("email");  String valistr = request.getParameter("valistr"); |

1. **校验数据**

**//非空校验**

由于每个参数都需要判断是否为空或者为null，可以创建工具类抽取方法：

|  |
| --- |
| 创建包：cn.tedu.utils，创建类WebUtils，添加如下代码：  //构造方法私有化，防止别人创建示例  **private** WebUtils(){}    /\*\*  \* 检查字符串是否为空字符串或者null  \* **@param** str  \* **@return** boolean true表示空字符串或者null  \*/  **public** **static** **boolean** isNull(String str){  **return** str == **null** || "".equals(str.trim());  } |
| RegistServlet中继续添加如下代码：  //3.校验数据  //>>非空校验  **if**(WebUtils.*isNull*(username)){  //将提示存入request域中，通过转发将消息带到regist.jsp中  request.setAttribute("msg", "用户名不能为空！");  request.getRequestDispatcher("/regist.jsp").forward(request, response);  **return**;  }  **if**(WebUtils.*isNull*(password)){  //将提示存入request域中，通过转发将消息带到regist.jsp中  request.setAttribute("msg", "密码不能为空！");  request.getRequestDispatcher("/regist.jsp").forward(request, response);  **return**;  }  **if**(WebUtils.*isNull*(password2)){  //将提示存入request域中，通过转发将消息带到regist.jsp中  request.setAttribute("msg", "确认密码不能为空！");  request.getRequestDispatcher("/regist.jsp").forward(request, response);  **return**;  }  **if**(WebUtils.*isNull*(nickname)){  //将提示存入request域中，通过转发将消息带到regist.jsp中  request.setAttribute("msg", "昵称不能为空！");  request.getRequestDispatcher("/regist.jsp").forward(request, response);  **return**;  }  **if**(WebUtils.*isNull*(email)){  //将提示存入request域中，通过转发将消息带到regist.jsp中  request.setAttribute("msg", "邮箱不能为空！");  request.getRequestDispatcher("/regist.jsp").forward(request, response);  **return**;  }  **if**(WebUtils.*isNull*(valistr)){  //将提示存入request域中，通过转发将消息带到regist.jsp中  request.setAttribute("msg", "验证码不能为空！");  request.getRequestDispatcher("/regist.jsp").forward(request, response);  **return**;  } |

**//密码一致校验**

|  |
| --- |
| //两次密码是否一致  **if**(!password.equals(password2)){  //将提示存入request域中，通过转发将消息带到regist.jsp中  request.setAttribute("msg", "两次密码不一致！");  request.getRequestDispatcher("/regist.jsp").forward(request, response);  **return**;  } |

**//邮箱格式校验**

|  |
| --- |
| //邮箱格式校验  //abc@sina.com.cn  **if**(!email.matches("^\\w+@\\w+(\\.\\w+)+$")){  //将提示存入request域中，通过转发将消息带到regist.jsp中  request.setAttribute("msg", "邮箱格式不正确！");  request.getRequestDispatcher("/regist.jsp").forward(request, response);  **return**;  } |

**//验证码是否正确（以后做，需要用到session）**

//TODO --此注释可以在Tasks栏中看到并点开。

1. **实现注册**

|  |
| --- |
| 首先，导包，将c3p0和mysql的驱动包导入到WEB-INF下的lib文件夹中（无需build path）。  然后，将c3p0配置文件放到src根目录下，注意数据库改为easymall。  在mysql中创建easymall数据库和user表，并插入数据（直接执行课前资料中的sql脚本）。 |
| 在cn.tedu.utils包中创建JDBCUtils类，并加入释放资源代码：  **private** JDBCUtils(){}    /\*\*  \* 释放资源  \* **@param** conn 连接对象  \* **@param** stat 传输器对象  \* **@param** rs 结果集对象  \*/  **public** **static** **void** close(Connection conn,Statement stat,ResultSet rs){  **if**(rs != **null**){  **try** {  rs.close();  } **catch** (SQLException e) {  e.printStackTrace();  }**finally**{  rs = **null**;  }  }  **if**(stat != **null**){  **try** {  stat.close();  } **catch** (SQLException e) {  e.printStackTrace();  }**finally**{  stat = **null**;  }  }  **if**(conn != **null**){  **try** {  conn.close();  } **catch** (SQLException e) {  e.printStackTrace();  }**finally**{  conn = **null**;  }  }  } |
| 写代码时发现，如果将连接池创建在RegistServlet中，则每次需要连接数据库都要创建一个连接池，所以可以将创建连接池代码放在JDBCUtils中，并提供获取连接池的方法和获取连接的方法，代码如下：  **private** **static** ComboPooledDataSource *pool* = **new** ComboPooledDataSource();  **private** JDBCUtils(){}    /\*\*  \* 获取连接池的方法  \* **@return** DataSource  \*/  **public** **static** DataSource getPool(){  **return** *pool*;  }    /\*\*  \* 从连接池中获取一个连接  \* **@return** Connection  \*/  **public** **static** Connection getConnection(){  **try** {  **return** *pool*.getConnection();  } **catch** (SQLException e) {  e.printStackTrace();  **throw** **new** RuntimeException();  }  } |
| 在RegistServlet中加入如下代码：  //4.实现注册（将用户注册数据保存到数据库中）  Connection conn = **null**;  PreparedStatement ps = **null**;  ResultSet rs = **null**;  **try** {  //从连接池中获取连接  conn = JDBCUtils.*getConnection*();  //获取传输器  String sql = "insert into user values(null,?,?,?,?)";  ps = conn.prepareStatement(sql);  ps.setString(1, username);  ps.setString(2, password);  ps.setString(3, nickname);  ps.setString(4, email);  //执行sql语句  ps.executeUpdate();    } **catch** (Exception e) {  e.printStackTrace();  **throw** **new** RuntimeException();  }**finally**{  //释放资源  JDBCUtils.*close*(conn, ps, rs);  } |

**//用户名是否存在验证**

为了减少代码，可以将用户名是否存在验证放到注册代码中，在将注册信息插入到数据库之前先进行用户名验证，如果用户名已经存在，则提示用户，并且不要将数据写入到数据库。

|  |
| --- |
| 在获取连接后插入如下代码：  //从连接池中获取连接  conn = JDBCUtils.*getConnection*();    //>>用户名是否存在（如果存在，跳转回注册页面并提示）  String sql = "select \* from user where username=?";  ps = conn.prepareStatement(sql);  ps.setString(1, username);  rs = ps.executeQuery();  **if**(rs.next()){ //用户名已存在  request.setAttribute("msg", "用户名已存在！");  request.getRequestDispatcher("/regist.jsp").forward(request, response);  **return**;  } |

1. **提示用户注册成功, 3秒之后将会跳转到首页!**

|  |
| --- |
| //5.提示用户注册成功，3秒后跳转到首页。  response.getWriter().write("<h1 style='color:red;text-align:center'>恭喜您注册成功，3秒之后跳转回首页...</h1>");  response.setHeader("refresh", "3;url="+request.getContextPath()+"/index.jsp"); |

**问题一：无提示信息**

此时，虽然提示信息放入到request域中，但是页面中并未获取，需要在页面中获取后才能显示。

|  |
| --- |
| 在regist.jsp中的table中添加如下代码：  <table>  <tr>  <td colspan=*"2"* style="color:*red*;text-align: *center*">  <%= request.getAttribute("msg") == **null** ? "" : request.getAttribute("msg") %>  </td>  </tr> |

**问题二：无回显**

当注册提交后，如果提交信息错误，转发回注册页面，发现之前所写的所有信息都为空了，需要重新填写，所以应该让之前的数据回显。

|  |
| --- |
| 在regist.jsp中每个input中加入回显代码，例如：  <input type=*"text"* name=*"username"* value=*"*<%=request.getParameter("username") == **null** ? "" : request.getParameter("username")%>*"*/> |

**问题三：验证码并非动态生成**

验证码应该是动态生成，写验证码的java程序有很多，在课前资料中已经提供一个，可以直接拿过来用。

|  |
| --- |
| 将课前资料resource中的VerifyCode.java程序拷贝到utils包中，这是一个生成验证码图片的类 |
| 在regist.jsp中，将验证码图片的访问路径改为：  <img src=*"*<%= request.getContextPath() %>*/servlet/ValiImageServlet"* alt=*""* /> |
| 在web包中创建servlet：ValiImageServlet，在doGet()中添加如下代码：  //控制浏览器不要缓存图片  response.setDateHeader("Expires", -1);  response.setHeader("Cache-Control", "no-cache");    VerifyCode vc = **new** VerifyCode();  //将图片保存到response缓冲区中，再响应给浏览器  vc.drawImage(response.getOutputStream()); |

### 10.2.4.实现注册功能-前端

**验证码点击更换**

验证码应该可以点击更换，要不然想更换验证码还需要刷新浏览器，就太麻烦了。

|  |
| --- |
| 首先，需要在regist.jsp中引入jQuery库：  在WebRoot下新建文件夹js，然后将课前资料中的jquery-1.4.2.js文件拷贝到此文件夹中。 |
| 在regist.jsp中的验证码<img>标签中加上id属性：  <img id=*"img"* src=*"*<%= request.getContextPath() %>*/servlet/ValiImageServlet"* alt=*""* />  在regist.jsp中的<head>标签内部加入如下代码：  <!-- 引入jQuery库 -->  <script type=*"text/javascript"* src=*"*<%= request.getContextPath() %>*/js/jquery-1.4.2.js"*></script>    <script type=*"text/javascript"*>  /\* 点击图片换一张验证码 \*/  $(**function**(){  $("#img").click(**function**(){  /\*  在页面中，任意一个元素发生改变，浏览器都会检测到，并且重新显示页面元素。  所以，只需要每次改变访问验证码后台的路径，浏览器即可改变元素的内容。  但是为了每次都是访问ValiImageServlet，资源路径不能改变，  但是可以拼接动态的参数，这样既不会影响访问的资源，又能改变元素的内容  \*/  $(**this**).attr("src","<%= request.getContextPath() %>/servlet/ValiImageServlet?time="+new Date().getTime());  });  })  </script> |

**1.2.注册表单的js校验**

每次点击注册的时候，都将数据提交到后台进行校验，会增加服务器压力，所以可以在jsp页面中利用js进行数据的校验，如果数据有误，则无需提交到后台。

1. form中加入onsubmit属性，然后<script>中自定义formObj对象和checkForm属性，可以return true/false测试
2. 由于checkForm中大量代码重复，分别创建其他三个属性（checkNull、checkPassword、checkEmail）
3. 完成checkNull中代码，发现设置消息应该放在输入框后，td后加<span>
4. 设置消息代码以后需要经常用到，所以对象中再添加一个属性setMsg
5. checkForm中进行非空验证，并修改return。
6. 测试发现提示后即使输入内容，提示依然没有消失，在checkNull中先清空
7. 完成checkPassword中代码
8. 完成checkEmail中代码
9. 在文档就绪事件中加入失去焦点提示事件

|  |
| --- |
| 首先，在regist.jsp的<form>标签中加入onsubmit属性：  <!--  onsubmit事件在表单提交时触发，该事件会根据返回值决定是否提交表单，  如果onsubmit="return true"会继续提交表单  如果onsubmit="return false"则不会提交表单    onsubmit中如果报错，并非代码问题，  而是MyEclipse工具在检查语法时认为代码有误，其实没有错误  -->  <form onsubmit="return formObj.checkForm()" action=*"*<%= request.getContextPath() %>*/servlet/RegistServlet"* method=*"POST"*> |
| 在regist.jsp页面中的每个<input>后加上一个<span>标签，为了以后存放提示信息，例如：  <tr>  <td class=*"tds"*>用户名：</td>  <td>  <input type=*"text"* name=*"username"* value=*"*<%=request.getParameter("username") == **null**?"":request.getParameter("username")%>*"*/>  <span></span>  </td>  </tr> |
| 在regist.jsp的<head>标签内的<script>标签中加入如下代码：  /\* 注册表单的js校验 \*/  **var** formObj = {  "checkForm" : **function**(){  //1.非空检验  **var** flag1 = **this**.checkNull("username", "用户名不能为空！");  **var** flag2 = **this**.checkNull("password", "密码不能为空！");  **var** flag3 = **this**.checkNull("password2", "确认密码不能为空！");  **var** flag4 = **this**.checkNull("nickname", "昵称不能为空！");  **var** flag5 = **this**.checkNull("email", "邮箱不能为空！");  **var** flag6 = **this**.checkNull("valistr", "验证码不能为空！");  //2.两次密码是否一致  **var** flag7 = **this**.checkPassword("password", "两次密码不一致！");  //3.邮箱格式校验  **var** flag8 = **this**.checkEmail("email", "邮箱格式不正确！");    **return** flag1&&flag2&&flag3&&flag4&&flag5&&flag6&&flag7&&flag8;  },  /\* 非空校验 \*/  "checkNull" : **function**(name,msg){  **var** value = $("input[name='"+name+"']").val();  //清空之前的提示消息  **this**.setMsg(name, "");    **if**(value.trim() == ""){  //设置提示消息  **this**.setMsg(name,msg);  **return false**;  }  **return true**;  },  //设置错误提示消息  "setMsg" : **function**(name,msg){  $("input[name='"+name+"']").nextAll("span").html(msg).css("color","red");  },  /\* 检查两次密码是否一致 \*/  "checkPassword" : **function**(name,msg){  **var** pw1 = $("input[name='"+name+"']").val();  **var** pw2 = $("input[name='"+name+"2']").val();  **if**(pw1 != "" && pw2 != "" && pw1 != pw2){  //设置提示消息  **this**.setMsg(name+"2", msg);  **return false**;  }  **return true**;  },  /\* 检查邮箱格式是否正确 \*/  "checkEmail" : **function**(name,msg){  **var** email = $("input[name='"+name+"']").val();  **var** regExp = /^\w+@\w+(.\w+)+$/;  **if**(email != "" && !regExp.test(email)){  **this**.setMsg(name, msg);  **return false**;  }  **return true**;  }  } |
| 为了提高页面的友好性，当鼠标失去焦点时也应该进行提示。  在regist.jsp的<script>标签内的文档就绪事件中加入如下代码：  /\* 给所有输入框添加失去输入焦点事件，当失去输入焦点时进行非空校验或者密码一致校验或者邮箱格式校验 \*/  //用户名  $("input[name='username']").blur(**function**(){  formObj.checkNull("username", "用户名不能为空！");  });  //密码  $("input[name='password']").blur(**function**(){  formObj.checkNull("password", "密码不能为空！");  });  //确认密码  $("input[name='password2']").blur(**function**(){  formObj.checkNull("password2", "确认密码不能为空！");  formObj.checkPassword("password", "两次密码不一致！");  });  //昵称  $("input[name='nickname']").blur(**function**(){  formObj.checkNull("nickname", "昵称不能为空！");  });  //邮箱  $("input[name='email']").blur(**function**(){  formObj.checkNull("email", "邮箱不能为空！");  formObj.checkEmail("email", "邮箱格式不正确！")  });  //验证码  $("input[name='valistr']").blur(**function**(){  formObj.checkNull("valistr", "验证码不能为空！");  });  }) |

代码无误而MyEclipse报错问题：

选中报错的jsp，右键->MyEclipse->Exclude From Validation

如果还有报错，可以：右键->MyEclipse->Run Validation，然后互相切换。