

一、实现登录

1.1 建包、导包与建立jsp页面

1.1.1 建包

MVC三层架构三个包：web、service、dao包 建立

实体类包建立：domain包

数据库连接工具：util包

1.1.2 导包

mysql数据库驱动包

连接池包

DBUtils包

JSTL相关包两个

1.1.3 jsp页面建立

登录页面.jsp

登录页面跳转.jsp

错误时提醒

1.2 实现代码详解

1.2.1 web层代码

```

22     request.setCharacterEncoding("UTF-8");
23
24     HttpSession session = request.getSession();
25
26     // 获取数据
27     String username = request.getParameter("username");//中文 张三
28     String password = request.getParameter("password");
29
30     UserService service = new UserService();
31     User user = null;
32     try { //调用service层的方法，只是简单传递而已
33         user = service.login(username,password);
34     } catch (SQLException e) {
35         e.printStackTrace();
36     }
37

```

成功则 // 将登录的用户user对象存到session中

```

session.setAttribute("user", user);
// 重定向到首页
response.sendRedirect(request.getContextPath());

```

失败则 // 失败 转发到登录页面 提出提示信息

```

request.setAttribute("loginInfo", "用户名或密码错误");
request.getRequestDispatcher("/login.jsp").forward(request, response);

```

`session.setAttribute(名, 值)`为啥要执行? : 因为我们需要将user的信息转发给别的资源, 这里是转给首页显示已经登录。

首页index.jsp如何显示用户是否登录? 用JSTL的if标签即可, 如果user不为空, 则得到user的值并显示已经登录; 不然显示请登录。

```

13     <div class="col-md-3" style="padding-top:20px">
14         <ol class="list-inline">
15
16             <c:if test="${empty user }">
17                 <li><a href="login.jsp">登录</a></li>
18                 <li><a href="register.jsp">注册</a></li>
19             </c:if>
20             <c:if test="${!empty user }">
21                 <li>欢迎您 ${user.username}</li>
22                 <li><a href="#">退出</a></li>
23             </c:if>
24
25             <li><a href="cart.jsp">购物车</a></li>
26             <li><a href="order_list.jsp">我的订单</a></li>
27         </ol>
28     </div>

```

登录错误提醒:

当然是EL表达式及JSTL, 只是登录错误时不跳转了, 或者还是跳转到本页, 即login.jsp。


```


55         <font>会员登录</font>USER LOGIN
56     <div>
57         <span style="color: red">${loginInfo }</span>
58     </div>

```

如果用户的参数很多呢

还用getParameter来获得，然后再封装到User里面吗？显然效率太低

 commons-beanutils-1.8.3.jar

 commons-logging-1.1.1.jar

有一个神器，BeanUtils，包如上所示，简化了操作

```

// 使用BeanUtils进行自动映射封装
// BeanUtils工作原理：将map中的数据 根据key与实体的属性的对应关系封装
// 只要key的名字与实体的属性的名字一样 就自动封装到实体中
Map<String, String[]> properties = request.getParameterMap();
User user = new User();
try {
    BeanUtils.populate(user, properties);
} catch (IllegalAccessException | InvocationTargetException e) {
    e.printStackTrace();
}

// 现在这个位置 user对象已经封装好了
// 手动封装uid----uuid---随机不重复的字符串32位--java代码生成后是36位
user.setUid(UUID.randomUUID().toString());

```

只要某个域中的参数的key与我们的实体类的属性一致，就能将其属性封装到实体类中

从而实现表单数据到实体数据的映射

如果有特别需要设置的值，可以调用实体类的set方法重新设置

1.2.2 service 层的代码

```

10 public class UserService {
11
12     public User login(String username, String password) throws SQLException {
13         UserDao dao = new UserDao();
14         return dao.login(username, password);
15     }
16
17 }

```

1.2.3 dao层的代码

```

11 public class UserDao {
12
13     public User login(String username, String password) throws SQLException {
14
15         QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
16         String sql = "select * from user where username=? and password=?";
17         return runner.query(sql, new BeanHandler<User>(User.class), username, password);
18     }
19
20 }
21

```

根本不用讲解

二、实现自动登录

一个过滤器

并做以下三步：

判断cookie是否存在，注意需要强转一个Request对象和Response对象

```
23 public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
24     throws IOException, ServletException {
25     HttpServletRequest req = (HttpServletRequest) request;
26     HttpServletResponse resp = (HttpServletResponse) response;
27     HttpSession session = req.getSession();
28
29     // 获得cookie中用户名和密码 进行登录的操作
30     // 定义cookie_username
31     String cookie_username = null;
32     // 定义cookie_password
33     String cookie_password = null;
34     // 获得cookie
35     Cookie[] cookies = req.getCookies();
36     if(cookies!=null){
37         for(Cookie cookie : cookies){
38             // 获得名字是cookie_username和cookie_password
39             if("cookie_username".equals(cookie.getName())){
40                 cookie_username = cookie.getValue();
41                 // 恢复中文用户名
42                 cookie_username = URLDecoder.decode(cookie_username, "UTF-8");
43             }
44             if("cookie_password".equals(cookie.getName())){
45                 cookie_password = cookie.getValue();
46             }
47         }
48     }
```

判断user信息是否为空，不为空则用该信息调用登录代码

```
50     // 判断username和password是否是null
51     if(cookie_username!=null&&cookie_password!=null){
52         // 登录的代码
53         UserService service = new UserService();
54         User user = null;
55         try {
56             user = service.login(cookie_username, cookie_password);
57         } catch (SQLException e) {
58             e.printStackTrace();
59         }
60         // 将登录的用户的user对象存到session中
61         session.setAttribute("user", user);
62     }
63
64     // 放行
65     chain.doFilter(request, response);
66
67 }
```

放行该请求，依旧访问该资源(只是会得到是否已经自动登录的状态)

```
64     // 放行
65     chain.doFilter(request, response);
66
67 }
```

三、关于乱码问题

四、关于装饰者模式