3、项目第六阶段

3.1、登陆---显示用户名

UserServlet 程序中保存用户登录的信息

```
} else {
    // 登录 成功
    // 保存用户登录的信息到Session域中
    req.getSession().setAttribute( s: "user", loginUser);
    //跳到成功页面login_success.html
    req.getRequestDispatcher( s: "/pages/user/login_success.jsp").forward(req, resp);
}
```

修改 login_succuess_menu.jsp

```
<div>
    <span>欢迎<span class="um_span">${sessionScope.user.username}</span>光临尚硅谷书城</span>
    <a href="pages/order/order.jsp">我的订单</a>
    <a href="index.jsp">注销</a>
    <a href="index.jsp">返回</a>
</div>
```

还要修改首页 index.jsp 页面的菜单 ：

```
<div>
    <%--如果用户还没有登录，显示 【登录 和注册的菜单】 --%>
    <c:if test="${empty sessionScope.user}">
        <a href="pages/user/login.jsp">登录</a> |
        <a href="pages/user/regist.jsp">注册</a>
    </c:if>
    <%--如果已经登录，则显示 登录 成功之后的用户信息。--%>
    <c:if test="${not empty sessionScope.user}">
        <span>欢迎<span class="um_span">${sessionScope.user.username}</span>光临尚硅谷书城</span>
        <a href="pages/order/order.jsp">我的订单</a>
        <a href="index.jsp">注销</a>
    </c:if>
```

3.2、登出---注销用户

1、销毁 Session 中用户登录的信息（或者销毁 Session）
2、重定向到首页（或登录页面）。

UserServlet 程序中添加 logout 方法

```
    /**
     * 注销
     * @param req
     * @param resp
     * @throws ServletException
     * @throws IOException
     */
    protected void logout(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
//        1、销毁 Session 中用户登录的信息（或者销毁 Session）
        req.getSession().invalidate();
//        2、重定向到首页（或登录页面）。
        resp.sendRedirect(req.getContextPath());
    }
```

修改【注销】的菜单地址

```
<a href="userServlet?action=logout">注销</a>
```

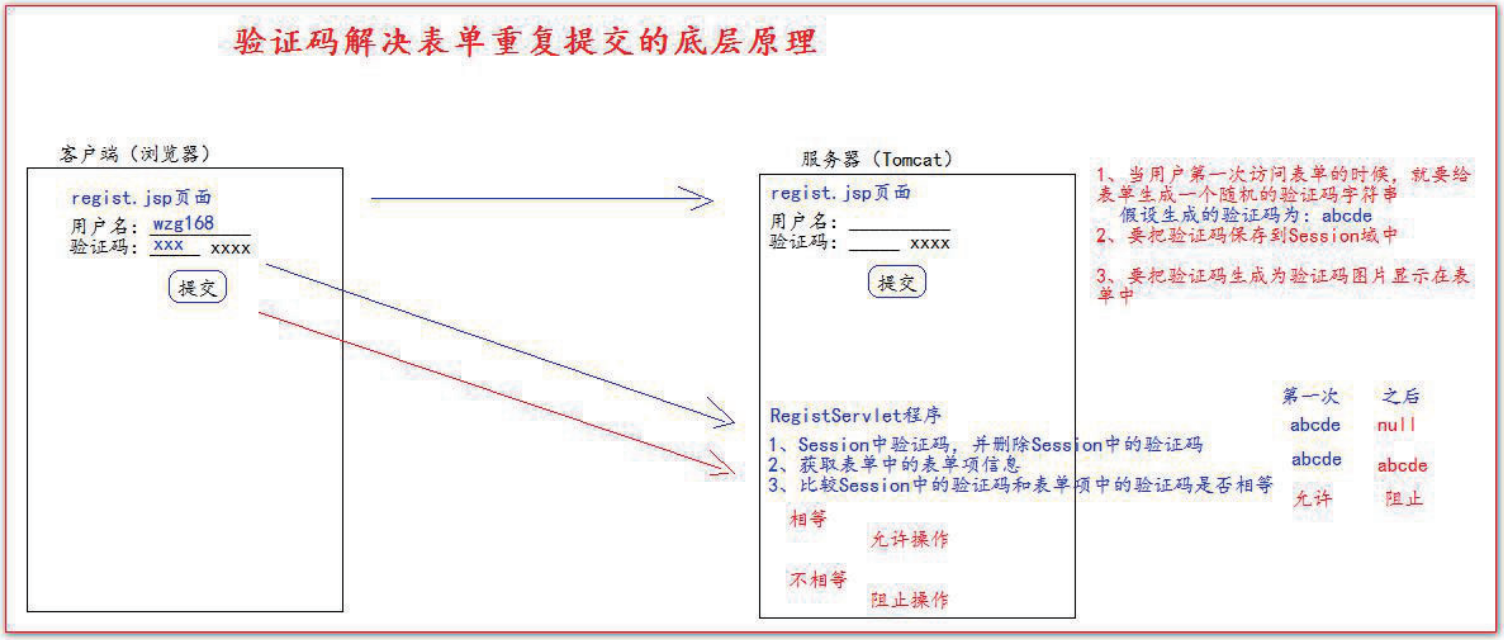
3.3、表单重复提交之-----验证码

表单重复提交有三种常见的情况：

一：提交完表单。服务器使用请求转来进行页面跳转。这个时候，用户按下功能键 F5，就会发起最后一次的请求。造成表单重复提交问题。解决方法：使用重定向来进行跳转

二：用户正常提交服务器，但是由于网络延迟等原因，迟迟未收到服务器的响应，这个时候，用户以为提交失败，就会着急，然后多点了几次提交操作，也会造成表单重复提交。

三：用户正常提交服务器。服务器也没有延迟，但是提交完成后，用户回退浏览器。重新提交。也会造成表单重复提交。



3.4、谷歌 kaptcha 图片验证码的使用

谷歌验证码 kaptcha 使用步骤如下：

1、导入谷歌验证码的 jar 包

kaptcha-2.3.2.jar

2、在 web.xml 中去配置用于生成验证码的 Servlet 程序

```
<servlet>
    <servlet-name>KaptchaServlet</servlet-name>
    <servlet-class>com.google.code.kaptcha.servlet.KaptchaServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>KaptchaServlet</servlet-name>
    <url-pattern>/kaptcha.jpg</url-pattern>
</servlet-mapping>
```

3、在表单中使用 img 标签去显示验证码图片并使用它

```
<form action="http://localhost:8080/tmp/registServlet" method="get">
  用户名：<input type="text" name="username" > <br>
  验证码：<input type="text" style="width: 80px;" name="code">
  <img src="http://localhost:8080/tmp/kaptcha.jpg" alt="" style="width: 100px; height: 28px;"> <br>
  <input type="submit" value="登录">
</form>
```

4、在服务器获取谷歌生成的验证码和客户端发送过来的验证码比较使用。

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
    // 获取 Session 中的验证码
    String token = (String) req.getSession().getAttribute(KAPTCHA_SESSION_KEY);
    // 删除 Session 中的验证码
    req.getSession().removeAttribute(KAPTCHA_SESSION_KEY);


    String code = req.getParameter("code");
    // 获取用户名
    String username = req.getParameter("username");

    if (token != null && token.equalsIgnoreCase(code)) {
        System.out.println("保存到数据库：" + username);
        resp.sendRedirect(req.getContextPath() + "/ok.jsp");
    } else {
        System.out.println("请不要重复提交表单");
    }
}
```

切换验证码：

```
// 给验证码的图片，绑定单击事件
$("#code_img").click(function () {
   // 在事件响应的 function 函数中有一个 this 对象。这个 this 对象，是当前正在响应事件的 dom 对象
   // src 属性表示验证码 img 标签的 图片路径。它可读，可写
   // alert(this.src);
   this.src = "${basePath}kaptcha.jpg?d=" + new Date();
});
```