<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>  
<h3>登录页面</h3>  
<form action="text/yourLogin.action" method="post">  
 <table>  
 <tr>  
 <td>用户名：</td>  
 <td><input type="text" name="userName"/></td>  
 </tr>  
 <tr>  
 <td>密 码：</td>  
 <td><input type="password" name="userpwd"/></td>  
 </tr>  
 <tr>  
 <td><input type="submit" value="提交"/></td>  
 <td><input type="reset" value="重置"/></td>  
 </tr>  
 </table>  
</form>  
</body>  
</html>



登录页面实验报告

核心功能解析

1.页面结构设计

使用<table>布局实现表单元素对齐

标题<h3>突出页面主题

表单元素包含：

文本输入框：<input type="text">（用户名）

密码输入框：<input type="password">（密码）

提交按钮：<input type="submit">

重置按钮：<input type="reset">

2表单提交机制

html

<form action="text/yourLogin.action" method="post">

action：指定后端处理登录请求的路径

method="post"：敏感信息通过POST请求传输（避免密码暴露在URL中）

3.参数传递方式

用户名参数名：userName

密码参数名：userpwd

后端可通过request.getParameter("userName")获取数据

实验心得：登录功能实现要点

1. 安全性增强建议

XSS防护：

jsp

<input type="text" name="userName" value="${fn:escapeXml(param.userName)}"/>

使用JSTL的fn:escapeXml防止跨站脚本攻击

密码加密传输：

html

<form action="yourLogin.action" method="post" autocomplete="off">

添加autocomplete="off"禁用浏览器自动填充

2. 后端验证逻辑（示例）

java

// LoginServlet.java

protected void doPost(HttpServletRequest request, HttpServletResponse response) {

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

String password = request.getParameter("userpwd");

// 1. 空值校验

if (username == null || password == null || username.isEmpty()) {

request.setAttribute("error", "用户名和密码不能为空");

request.getRequestDispatcher("/login.jsp").forward(request, response);

return;

}

// 2. 密码验证（实际应查数据库）

if ("admin".equals(username) && "123456".equals(password)) {

request.getSession().setAttribute("user", username);

response.sendRedirect("welcome.jsp");

} else {

request.setAttribute("error", "用户名或密码错误");

request.getRequestDispatcher("/login.jsp").forward(request, response);

}

}

3. 用户体验优化

错误信息显示：

jsp

<c:if test="${not empty error}">

<div style="color:red">${error}</div>

</c:if>

在表单上方添加错误提示区域

记住用户名功能：

jsp

<input type="checkbox" name="rememberMe" value="true"> 记住用户名

后端通过Cookie实现记住用户名功能

4. 安全性深度加固

防暴力破解：

java

// 记录登录失败次数

Integer attempts = (Integer) session.getAttribute("loginAttempts");

if (attempts == null) attempts = 0;

if (attempts > 3) {

request.setAttribute("error", "超过尝试次数，请10分钟后再试");

// 转发回登录页

}

密码哈希处理：

java

// 使用BCrypt加密验证

if (BCrypt.checkpw(password, storedHash)) {

// 登录成功

}

问题分析与解决方案

当前代码存在的不足

1.路径问题：

action="text/yourLogin.action" 可能导致404错误

解决方案：

html

<form action="${pageContext.request.contextPath}/login" method="post">

2.密码字段命名：

name="userpwd" 建议改为 name="password" 保持行业惯例

3.缺少标签关联：

改进方案：

html

<label for="username">用户名：</label>

<input id="username" type="text" name="userName">

安全风险防范表

风险类型 防护措施

SQL注入 PreparedStatement参数化查询

会话劫持 启用HTTPS + HttpOnly Cookie

CSRF攻击 添加CSRF Token验证

密码明文传输 前端RSA加密/启用HTTPS

暴力破解 验证码+登录失败锁定机制

总结

本实验实现了基础登录页面功能：

1.前端设计：通过表格布局实现简洁的表单界面，包含用户名/密码输入和操作按钮

2.数据传递：使用POST方法提交敏感信息，防止密码暴露

3.扩展方向：

添加验证码功能（防止机器人攻击）

实现OAuth2.0第三方登录

增加密码强度实时检测

4.安全核心：

后端必须进行输入验证和参数过滤

密码存储使用BCrypt/PBKDF2等哈希算法

关键操作记录审计日志

关键经验：登录功能是系统安全的第一道防线，必须采用深度防御策略。前端实现只是基础，需要结合后端验证、会话管理、加密传输等多层防护才能构建安全的认证系统。