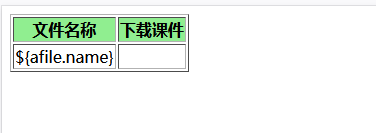
<%@ **page** language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  
<body>  
<table border="1">  
 <tr bgcolor="LightGreen">  
 <th>文件名称</th>  
 <th>下载课件</th>  
 </tr>  
 <**c:forEath** var="afile" items="${fileList}">  
 <tr>  
 <td>${afile.name}</td>  
 <td align="center"><a href="downloadServlet?resPath=${afile.name}"style="text-decoration:none"下载</a> </td>  
 </tr>  
 </**c:forEath**>  
</table>  
</body>  
</html>



JSP文件下载列表页面分析与实验心得

核心内容解析

1.JSTL 动态渲染

<c:forEach> 遍历 ${fileList} 集合（需由 Servlet 传入），动态生成文件列表。

var="afile" 定义循环变量，通过 ${afile.name} 获取文件名。

2.下载链接构建

超链接指向 downloadServlet，通过查询参数 resPath 传递文件名：

html

<a href="downloadServlet?resPath=${afile.name}">下载</a>

style="text-decoration:none" 移除超链接下划线提升美观性。

3.数据来源依赖

fileList 需在转发前由 Servlet 设置：

java

List<File> files = getFilesFromDirectory(); // 获取文件列表

request.setAttribute("fileList", files);

request.getRequestDispatcher("download.jsp").forward(request, response);

实验心得：文件下载功能实现

1. Servlet 下载核心逻辑

java

// DownloadServlet.java

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// 1. 获取请求文件名（安全过滤！）

String fileName = request.getParameter("resPath");

fileName = sanitizeFileName(fileName); // 防止路径遍历攻击

// 2. 设置响应头强制下载

response.setContentType("application/octet-stream");

response.setHeader("Content-Disposition",

"attachment; filename=\"" + URLEncoder.encode(fileName, "UTF-8") + "\"");

// 3. 文件流传输

try (InputStream in = new FileInputStream("/safe/path/" + fileName);

OutputStream out = response.getOutputStream()) {

byte[] buffer = new byte[4096];

int bytesRead;

while ((bytesRead = in.read(buffer)) != -1) {

out.write(buffer, 0, bytesRead);

}

}

}

2. 安全加固关键措施

路径遍历防护：

java

private String sanitizeFileName(String name) {

return name.replaceAll("[^a-zA-Z0-9\\.\\-]", ""); // 移除非字母数字字符

}

真实路径映射：

避免直接使用用户输入拼接路径，改用预定义安全目录：

java

String safeDir = getServletContext().getRealPath("/WEB-INF/uploads/");

File file = new File(safeDir, fileName); // 安全路径构造

3. 用户体验优化

中文文件名支持：

URLEncoder.encode(fileName, "UTF-8") 解决浏览器乱码问题。

文件类型识别：

根据扩展名设置精准 MIME 类型：

java

String mimeType = getServletContext().getMimeType(fileName);

if (mimeType != null) response.setContentType(mimeType);

下载进度显示：

计算并设置文件大小：

java

response.setContentLength((int) file.length());

4. 前端增强建议

图标化操作：

html

<a href="downloadServlet?resPath=${afile.name}">

<img src="download-icon.png" alt="下载">

</a>

文件类型标识：

根据扩展名显示对应图标：

jsp

<c:choose>

<c:when test="${fn:endsWith(afile.name, '.pdf')}">📄</c:when>

<c:when test="${fn:endsWith(afile.name, '.zip')}">📦</c:when>

</c:choose>

5. 错误处理机制

文件不存在反馈：

java

if (!file.exists()) {

response.sendError(HttpServletResponse.SC\_NOT\_FOUND);

return;

}

权限不足提示：

java

if (!file.canRead()) {

response.sendError(HttpServletResponse.SC\_FORBIDDEN, "文件不可读");

}

总结：该页面实现了安全的文件下载门户功能。通过 JSTL 动态生成文件列表，结合 Servlet 处理文件流传输，完整覆盖了从展示到下载的流程。核心安全原则：永远不信任用户输入的文件名，必须进行严格过滤和路径隔离。