**Filter作业**

**账号密码默认：admin password**

**1. 创建 LoginFilter 类**

实现 javax.servlet.Filter 接口，定义过滤器的基本结构和功能。

**2. 配置过滤器**

使用 @WebFilter 注解将过滤器应用于所有 URL 路径（"/\*"）。这确保了所有请求都会经过该过滤器进行验证。

import jakarta.servlet.annotation.WebFilter;

@WebFilter("/\*")

public class LoginFilter implements Filter {

// Filter implementation goes here

}

**3. 实现 doFilter 方法**

在 doFilter 方法中实现以下逻辑：

• **检查请求类型**：

• 检查当前请求是否为登录页面、注册页面或其他公共资源。如果是，则允许请求通过。

• **验证用户登录状态**：

• 检查用户的 session 中是否存在表示已登录的属性（例如 "user" 属性）。

• 如果用户已登录，允许请求继续。

• 如果用户未登录，将请求重定向到登录页面。

@Override

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

throws IOException, ServletException {

HttpServletRequest httpRequest = (HttpServletRequest) request;

HttpServletResponse httpResponse = (HttpServletResponse) response;

HttpSession session = httpRequest.getSession(false);

String loginURI = httpRequest.getContextPath() + "/login.html";

String registerURI = httpRequest.getContextPath() + "/register.html";

// 检查是否为登录或注册请求

if (isPublicResource(httpRequest)) {

chain.doFilter(request, response); // 允许请求通过

} else {

// 验证用户登录状态

if (session != null && session.getAttribute("user") != null) {

chain.doFilter(request, response); // 已登录用户，允许请求继续

} else {

httpResponse.sendRedirect(loginURI); // 未登录用户，重定向到登录页面

}

}

}

**4. 创建排除列表**

定义一个排除列表，包含不需要登录即可访问的路径（如 /login、/register、/public）。

**5. 实现检查方法**

创建一个方法来检查当前请求路径是否在排除列表中。此方法会返回布尔值，指示请求是否可以被允许通过。

private boolean isPublicResource(HttpServletRequest request) {

String uri = request.getRequestURI();

return uri.equals(request.getContextPath() + "/login.html") ||

uri.equals(request.getContextPath() + "/register.html") ||

uri.startsWith(request.getContextPath() + "/public/");

}