Spring Security JSP Tag Library - How to Secure JSP Pages with Examples

Last Updated : 23 Jul, 2025

Spring Security provides a powerful **JSP Tag Library** that enables developers to manage **authentication and authorization** directly within **JSP files**. In this article, we will learn **how to use Spring Security JSP tags to secure JSP pages**, use of **role-based access**, and enhance web application security.

Prerequisites:

- Java 11 or later
- Maven or Gradle
- Spring Boot with Spring Security (Latest version: 6.x)
- JSP and Servlet API

Step-by-Step Guide to Using Spring Security JSP Tags

Step 1: Add Spring Security Dependencies

To use the Spring Security JSP Tag Library, you will first need to add the necessary dependencies to your project. The following dependencies should be added to your project's pom.xml file:

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>6.2.0</version>
</dependency>
```

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-taglibs</artifactId>
    <version>6.2.0</version>
</dependency>
```

Step 2: Configure Spring Security

Java Config (Recommended for Spring Security 6+)

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
        http
            .authorizeHttpRequests(auth -> auth
                .requestMatchers("/admin/**").hasRole("ADMIN")
                .anyRequest().authenticated()
            )
            .formLogin(form -> form
                .loginPage("/login")
                .defaultSuccessUrl("/home")
            )
            .logout(logout -> logout
                .logoutSuccessUrl("/login?logout")
            );
        return http.build();
    }

    @Bean
    public UserDetailsService userDetailsService() {
        UserDetails admin = User.withUsername("admin")
            .password("{noop}admin123") // Temporary plaintext (use BCrypt
in production)
            .roles("ADMIN")
            .build();
        return new InMemoryUserDetailsManager(admin);
    }
}
```

Legacy XML Config (If Still Needed):

```
<http auto-config="true">
    <intercept-url pattern="/admin/**" access="hasRole('ADMIN')"/>
    <form-login login-page="/login"/>
    <logout logout-url="/logout"/>
</http>
<authentication-manager>
    <authentication-provider>
        <user-service>
            <user name="admin" password="{noop}admin123"
authorities="ROLE_ADMIN"/>
        </user-service>
    </authentication-provider>
</authentication-manager>
```

Step 3: Use Spring Security Tags in JSP

Add the taglib declaration:

> *<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %>*

1. Role-Based Access (authorize)

> *<sec:authorize access="hasRole('ADMIN')">*
>
> *<p>Admin-only content!</p>*
>
> *<a href="/admin/dashboard">Admin Dashboard</a>*
>
> *</sec:authorize>*

2. Display User Info (authentication)

> *Welcome, <sec:authentication property="name"/>!*
>
> *Your roles: <sec:authentication property="authorities"/>*

3. CSRF Protection (csrfInput)

> *<form action="/update" method="post">*

```
    <sec:csrfInput />
    <input type="text" name="data"/>
    <button type="submit">Submit</button>
</form>
```

4. Logout Button (logout)

```
<sec:authorize access="isAuthenticated()">
    <form action="/logout" method="post">
        <sec:csrfInput />
        <button type="submit">Logout</button>
    </form>
</sec:authorize>
```

Other tags that are provided by the Spring Security JSP tag library include:

- "sec:csrfInput" - generates a hidden input field containing the CSRF token.
- "sec:csrfMetaTags" - generates meta tags containing the CSRF token.
- "sec:http" - generates an HTTP method input field.
- "sec:logout" - generates a logout link.

The Spring Security JSP Tag Library provides a set of tags that can be used to secure pages in a web application without having to write java code. These tags include:

- <security:authorize>: This tag is used to control access to a certain part of the page based on the user's role or other security attributes. The tag can be used to check if a user has a specific role or if they are authenticated, and only show the content within the tag if the user passes the check.

- <security:authentication>: This tag is used to display information about the currently logged-in users, such as their username and role.
- <security:accesscontrollist>: This tag is used to display a list of all the access rules that have been configured for the application.
- <security:csrfInput>: This tag is used to generate a hidden input field that contains the CSRF token. It should be used in forms that are submitted to the server to prevent CSRF attacks.
- <security:accessdenied>: This tag is used to display a message or page when a user tries to access a page they are not authorized to access.

To use these tags, you will need to import the tag library at the top of your JSP file using the following code:

By using these tags, you can secure your pages without having to write any java code and also make it more convenient for developers to control access to certain parts of a page based on the user's role or other security attributes.

Complete Example: Secure Admin Dashboard

admin.jsp

```
<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags'
%>
<!DOCTYPE html>
<html>
<head>
    <title>Admin Panel</title>
</head>
<body>
    <sec:authorize access="hasRole('ADMIN')">
        <h1>Admin Dashboard</h1>
        <p>Logged in as: <sec:authentication property="name"/></p>
        <form action="/logout" method="post">
            <sec:csrfInput />
            <button type="submit">Logout</button>
        </form>
    </sec:authorize>
</body>
</html>
```

The Spring Security JSP tag library is a powerful tool for controlling access to resources in a web application. By using the tags provided by the library, you can easily control access to resources based on the user's role and display information.