Interview Questions: Spring Security

November 20, 2013

On this page we are providing spring security interview questions with answers. Spring security provides authentication and authorization both. Using it, we can save our spring applications from attacks such as session fixation, clickjacking, cross site request forgery, etc. To work with spring security, we use spring boot which helps to quick start our application easily. Authentication and authorization both can be handled using spring context XML as well as java configuration. Here on this page we are providing good spring security interview questions which will strengthen spring security knowledge to crack the interview. For the spring security examples visit the page.

1. Which filter class is needed for spring security?

Ans: In case we are using XML file we need to configure org.springframework.web.filter.DelegatingFilterProxy in web.xml and if we are using Servlet 3, we can use `AbstractSecurityWebApplicationInitializer`. It configures `DelegatingFilterProxy` and `ContextLoaderListener`. In web.xml we define as follows.

```
<filter>
        <filter-name>springSecurityFilterChain</filter-name>
        <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
        <filter-name>springSecurityFilterChain</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

For servlet 3, we can define `AbstractSecurityWebApplicationInitializer` as follows.

```
import org.springframework.security.web.context.AbstractSecurityWebApplicationInitializer;
public class SecurityInitializer extends AbstractSecurityWebApplicationInitializer {
}
```

2. How to create custom login page in spring security?

Ans: To create custom login page we need to follow two steps.
1. Create a form as follows.

```
<form name='form' action='j_spring_security_check' method='POST'>
 <table>
        <tr>
          <td>User Name:</td>
          <td><input type='text' name='j_username' value=''></td>
        </tr>
        <tr>
          <td>Password:</td>
          <td><input type='password' name='j_password'/></td>
        </tr>
        <tr>
          <td colspan='2'>
                <input name="submit" type="submit" value="Login"/></td>
        </tr>
  </table>
</form>
```

j_spring_security_check: This is the action of form.
j_username : Username input type name.

j_password : Password input type name.

2. In spring configuration file, use `<form-login>` tag.

```
<http auto-config="true">
    <intercept-url pattern="/login" access="ROLE_USER" />
    <form-login login-page='/customLogin?login_error=1' default-target-url="/loginSuccess"/>
</http>
```

3. How to logout the session in spring security?

To logout the session, use **j_spring_security_logout** as follows.

```
<a href="j_spring_security_logout">logout </a>
```

And in spring context XML , configure <logout > tag for the URL to redirect after logout as follows.

```
<http auto-config="true">
        <intercept-url pattern="/login" access="ROLE_USER" />
        <logout logout-success-url="/login" />
</http>
```

4. How to configure user name and password in spring security using XML?

Ans: In spring context XML using <user> tag, we define username and password within <user-service> tag.

```
<authentication-manager>
        <authentication-provider>
                <user-service>
                        <user name="concretepage" password="con1234" authorities="ROLE_USER" />
                </user-service>
        </authentication-provider>
</authentication-manager>
```

5. What is Authentication and Authorization in Spring Security

Ans: **Authentication**: An application needs to know who is accessing the application. So authentication is related to word *who*. Application will check it by a login form. User will enter user name and password and these inputs will be validated by the application. Once the validation is successful, user is declared as authenticated.

Authorization : Authorization is to check whether user can access the application or not or what user can access and what user cannot access.

6. How to encode password in spring security using XML?

Ans: To encode password, spring security provides `<password-encoder/>` tag. Find sample use.

```
<authentication-manager>
  <authentication-provider>
    <password-encoder hash="sha"/>
    <user-service>
```

```
      <user name="concretepage" password="0733824cc1549ce36139e8c790a9344d1e385cd2"
            authorities="ROLE_USER" />
    </user-service>
  </authentication-provider>
</authentication-manager>
```

7. How to login with database in spring security?

Ans: Spring security provides `<jdbc-user-service>` tag using which we access user information from database.

```
<authentication-manager>
     <authentication-provider>
       <password-encoder hash="sha"/>
       <jdbc-user-service data-source-ref="dataSource" authorities-by-username-query="SELECT usernam
            users-by-username-query="SELECT username, password, enabled FROM users WHERE username =
     </authentication-provider>
</authentication-manager>
```

8. How to achieve "Remember Me" in spring security?

Ans: "Remember Me" concept facilitates a user that once the user logged in, next time user need not to login again. This is the user choice whether user wants to get this option or not. There will be a checkbox that can be checked or unchecked to remember the password. Spring security uses a token to perform this task. Spring security provides `<remember-me>` tag which is used as follows.

```
<http auto-config="true">
        <intercept-url pattern="/login" access="ROLE_USER" />
        <remember-me key="remMeKey"/>
</http>
```

9. What is channel security in spring?

Ans: Spring security provides the feature to secure the URL patterns. For any URL pattern if we want to allow only HTTPS access, we have to do a small configuration in our spring security configuration.

```
<intercept-url pattern="/login" access="ROLE_USER" requires-channel="https" />
```

10. How to manage Session in spring security?

Ans: To manage session we need to use `<session-management>` tag within `<http>` tag in our spring context XML.

```
<session-management invalid-session-url="/login" session-fixation-protection="newSession" >
   <concurrency-control max-sessions="1" error-if-maximum-exceeded="true" />
</session-management>
```

11. How to access user role in JSP in spring security?

Ans: First we need to include tag library as follows.

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

And then use `<security:authorize>` tag.

```
<security:authorize access="hasRole('ROLE_SUPERWISER')">
 Your Message
</security:authorize>
```

12. How to access user role in Controller in spring security?

Ans: Using `GrantedAuthority` class, we can access user role.

```
private boolean hasRole(String role) {
  Collection<GrantedAuthority> authorities = (Collection<GrantedAuthority>)
  SecurityContextHolder.getContext().getAuthentication().getAuthorities();
```

```
    boolean hasRole = false;
    for (GrantedAuthority authority : authorities) {
       hasRole = authority.getAuthority().equals(role);
       if (hasRole) {
            break;
       }
    }
    return hasRole;
}
```

13. How to get user details in spring security?

Ans: Using `UserDetails` class we can access user details. To get the instance of `UserDetails`, we need to use `SecurityContextHolder` as follows.

```
UserDetails userDetails = (UserDetails)SecurityContextHolder.getContext().
getAuthentication().getPrincipal();
```

14. How to use security pointcut in spring?

Ans: Spring security provides `<protect-pointcut>` tag to define expression.

```
<global-method-security >
    <protect-pointcut expression="execution(* com.concretepage.service.*Service.*(..))"
         access="ROLE_USER"/>
</global-method-security>
```

15. How to secure a method in spring security using annotation?

Ans: Spring security provides `@Secured` annotation that is annotated at method level.

```
@Secured("ROLE_ADMIN")
public void deleteUser(String name);
```

The above method can only be accessed by ADMIN role.

16. How to pre-authorize and post-authorize a method in spring security?

Ans: To pre-authorize a method we need to annotate it by `@PreAuthorize` and to post-authorize a method, we need to annotate it by `@PostAuthorize` in the interface.

```
@PreAuthorize ("hasRole('ROLE_WRITE')")
public void addBook(Book book);
@PostAuthorize ("returnObject.owner == authentication.name")
public Book getBook();
```

17. What is the role of @PreFilter and @PostFilter in spring security?

Ans: `@PreFilter` and `@PostFilter` is a strong feature in spring security to filter collection or arrays on the basis of authorization. This can be achieved using expression-based access control in spring security. `@PreFilter` filters the collection or arrays before executing method. `@PostFilter` filters the returned collection or arrays after executing the method.

```
@PostFilter ("filterObject.owner == authentication.name")
public List<Book> getBooks();
@PreFilter("filterObject.owner == authentication.name")
public void addBook(List<Book> books);
```

18. How to enable method level security in spring?

Ans: 1. If we are using xml based spring security, we need to use

```
<global-method-security secured-annotations="enabled", pre-post-annotations="enabled"/>
```

where `secured-annotations` attribute is for `@Secured` and `pre-post-annotations` attribute is for `@PreAuthorize` and `@PostAuthorize`.

2. If we are using java configuration, we need to annotate java configuration class with

```
@EnableGlobalMethodSecurity(securedEnabled=true, prePostEnabled=true)
```

Where `securedEnabled` is for `@Secured` and `prePostEnabled` attribute is for `@PreAuthorize` and `@PostAuthorize`.

19. How to enable web security using java configuration in spring?

Ans: Spring security provides `@EnableWebSecurity` annotation which is used with `@Configuration`. `@EnableWebSecurity` provides spring security configuration defined in `WebSecurityConfigurer` or `WebSecurityConfigurerAdapter`. To override security configuration we can extend `WebSecurityConfigurerAdapter` class in our java configuration class.

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {}
```

20. What is the spring security boot?

All the spring security JAR dependencies can be resolved in one go by using spring security boot. At the time of writing, the latest version of spring boot is *1.3.2.RELEASE*.

1. For gradle

```
compile 'org.springframework.boot:spring-boot-starter-security:1.3.2.RELEASE'
```

2. and for Maven

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
        <version>1.3.2.RELEASE</version>
</dependency>
```

21. What are the new annotations introduced in spring 4 security for JUnit test?

Ans: `@WithMockUser` and `@WithUserDetails` annotations are introduced in spring 4 security for JUnit test . `@WithMockUser` annotation allows mock user at server side in spring security JUnit testing. We use it as follows.

```
@Test
@WithMockUser(username = "ram", roles={"ADMIN"})
public class SpringSecurityTest {}
```

@WithUserDetails annotation provides custom UserDetailsService in spring security JUnit testing and we can use it as follows.

```
@Test
@WithUserDetails("ram")
public void testFour() {
        userService.methodFour();
}
```

22. What is the meaning of the message "Access is denied"?

Ans: User is authenticated but is not authorized for a particular resource, in that case that user gets the message **Access is denied**. We can authorize user roles for a URL pattern or for specific methods to secure it. When the users whose roles are not authorized for these resources, they get the message "Access is denied".

23. How to use hasRole() and hasAnyRole() in spring security?

Ans: **hasRole()**: It checks for the role which is passed as arguments and returns true or false. It checks for role in current principal.

```
<http auto-config="true" use-expressions="true">
    <intercept-url pattern="/login" access="hasRole('ROLE_READ')" />
</http>
```

hasAnyRole(): We pass more than one role and it checks for any of them in current principal and returns true or false.

```
<http auto-config="true" use-expressions="true">
    <intercept-url pattern="/login" access="hasAnyRole('ROLE_READ','ROLE_WRITE')" />
</http>
```


FIND MORE TUTORILAS

SPRING BOOT | HIBERNATE | PRIMEFACES | RESTEASY | FREEMARKER