February 23, 2017 NO COMMENTS

Spring Security Interview Questions and Answers

« Previous Next »

In this Spring security interview questions and answers tutorial I have selected some important question and their answers. Spring Security is one of the powerful and highly customizable authentication and access-control framework. Spring security provide Authentication and Authorization mechanism. Read my Spring Security Tutorial in detail to getting good understating into Spring Security before going to interview questions.

Spring Security Interview Questions

1. *What is the delegating filter proxy?*
2. *What is the security filter chain?*
3. *Mandatory Filter Name Main Purpose?*
4. *Are you able to add and/or replace individual filters?*
5. *Is it enough to hide sections of my output (e.g. JSP-Page)?*
6. *Why do you need the intercept-url?*
7. *In which order do you have to write multiple intercept-url's?*
8. *Why do you need method security?*
9. *Is security a cross cutting concern? How is it implemented internally?*

10. *What do @Secured and @RolesAllowed do? What is the difference between them?*
11. *What is a security context?*
12. *How is a Principal defined?*
13. *What is authentication and authorization? Which must come first?*
14. *In which security annotation are you allowed to use SpEL?*
15. *Does Spring Security support password hashing? What is salting?*

What is the delegating filter proxy?

Spring's DelegatingFilterProxy provides the link between web.xml and the application context. In Spring Security, the filter classes are also Spring beans defined in the application context and thus able to take advantage of Spring's rich dependency-injection facilities and lifecycle interfaces.

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

What is the security filter chain?

In Spring Security you have a lot of filters for web application and these filters are Spring Beans. Each Spring security filter bean that require in your application you have to declare in your application context file and as we know that filters would be applied to application only when they would be declared on web.xml. Now DelegatingFilterProxy comes into picture for delegating the request to fillter which declared into application context file by adding a corresponding DelegatingFilterProxy entry to web.xml for each filter and we have to make sure about ordered, it should be define correctly, but this would be cumbersome and would clutter up the web.xml file quickly if you have a lot of filters. FilterChainProxy lets us add a single entry to web.xml and deal

entirely with the application context file for managing our web security beans.

```
<bean id="filterChainProxy" class="org.springframew
<constructor-arg>
 <list>
 <sec:filter-chain pattern="/restful/**" filters="
  securityContextPersistenceFilterWithASCFalse,
  basicAuthenticationFilter,
  exceptionTranslationFilter,
  filterSecurityInterceptor" />
 <sec:filter-chain pattern="/**" filters="
  securityContextPersistenceFilterWithASCTrue,
  formLoginFilter,
  exceptionTranslationFilter,
  filterSecurityInterceptor" />
 </list>
</constructor-arg>
</bean>
```

Mandatory Filter Name Main Purpose?

1. **SecurityContextIntegrationFilter** – Establishes SecurityContext and maintains between HTTP requests
2. **LogoutFilter** – Clears SecurityContextHolder when logout requested
3. **UsernamePasswordAuthenticationFilter** – Puts Authentication into the SecurityContext on login request
4. **ExceptionTranslationFilter** – Converts SpringSecurity exceptions into HTTP response or redirect
5. **FilterSecurityInterceptor** – Authorizes web requests based on on config attributes and authorities

- *Spring HATEOAS*
- *Microservices with Spring Boot*
- *REST Webservice*
- *Core Java*
- *Hibernate Tutorial*
- *Spring Batch*

Are you able to add and/or replace individual filters?

Spring Security maintains a filter chain internally where each of the filters has a particular responsibility and filters are added or removed from the configuration depending on which services are required.

Is it enough to hide sections of my output (e.g. JSP-Page)?

No, because we cannot readily reverse engineer what URL is mapped to what controller endpoint as controllers can rely on headers, current user, etc to determine what method to invoke.

JSP Tag Libraries- Spring Security has its own taglib which provides basic support for accessing security information and applying security constraints in JSPs.

Why do you need the intercept-url?

intercept-url element is used to define the set of URL patterns that the application is interested in and to configure how they should be handled.

```
<intercept-url pattern='/secure/**' access='ROLE_A,
```

In which order do you have to write multiple intercept-url's?

When matching the specified patterns defined by element intercept-url against an incoming request, the matching is done in the order in which the elements are declared. So the most specific patterns should come first and the most general should come last.

```
<intercept-url pattern='/secure/a/**' access='ROLE_
<intercept-url pattern='/secure/b/**' access='ROLE_
<intercept-url pattern='/secure/**' access='ROLE_US
```

Why do you need method security? What type of object is typically secured at the method level.

Spring Security uses AOP for security at the method level

– annotations based on Spring annotations or JSR-250 annotations

– Java configuration to activate detection of annotations

It typically secure your services

– Do not access repositories directly, bypasses security (and transactions)

Is security a cross cutting concern? How is it implemented internally?

Yes, Spring Security is a cross cutting concern. Spring security is also using Spring AOP internally.

What do @Secured and @RolesAllowed do? What is the difference between them?

@Secured and @RolesAllowed both annotation provide method level security in to Spring Beans. @Secured is Spring Security annotation from version 2.0 onwards Spring Security. But @RolesAllowed is JSR 250 annoatation. Spring Security provides the support for JSR 250 annotation as well for method level security. @RolesAllowed provides role based security only.

What is a security context?

Security context in Spring Security includes details of the principal currently using the application. Security context is always available to methods in the same thread of execution, even if the security context is not explicitly passed around as an argument to those methods.

How is a Principal defined?

Inside the SecurityContextHolder we store details of the principal currently interacting with the application. Spring Security uses an Authentication object to represent this information.

```
</>
Object principal = SecurityContextHolder.getContext

if (principal instanceof UserDetails) {
String username = ((UserDetails)principal).getUsern
```

```
} else {
String username = principal.toString();
}
```

What is authentication and authorization? Which must come first?

- Authentication – Establishing that a principal's credentials are valid
- Authorization – Deciding if a principal is allowed to perform an action

Authentication comes first before Authorization because authorization process needs princial object with authority votes to decide user allow to perform a action for secured resource.

In which security annotation are you allowed to use SpEL?
They are @PreAuthorize, @PreFilter, @PostAuthorize and @PostFilter. These annotations support expression attributes to allow pre and post-invocation authorization checks and also to support filtering of submitted collection arguments or return values
Method security is a bit more complicated than a simple allow or deny rule. Spring Security 3.0 introduced some new annotations in order to allow comprehensive support for the use of expressions.

```
<global-method-security pre-post-annotations="enabl
```

```
@PreAuthorize("hasRole('USER')")
public void create(Contact contact);
```

Does Spring Security support password hashing? What is salting?
Yes, Spring Security provides support for password hashing.

The salt is used to prevent dictionary attacks against the key in the event your encrypted data is compromised.

Spring Security Related Posts

- Spring Security Interview Questions and Answers
- Spring Security Java Based Configuration with Example
- Spring Security XML Namespace Configuration Example
- Spring Security XML Based Hello World Example
- Spring Security form-based login example
- Spring Security Login Form Based Example Using Database
- Spring Security Authentication Example Using HTTP Basic
- Spring Security Authorized Access Control Example
- Spring Security Customized Access Denied Page
- Spring Security Custom Error Message
- Spring Security Logout Example
- Spring Security Fetch Logged in Username
- Spring Security Password Hashing

« Previous Next »

INTERVIEW QUESTIONS

SPRING SECURITY

Related Posts