[#Spring](https://www.facebook.com/hashtag/spring?source=feed_text&epa=HASHTAG) [#Security](https://www.facebook.com/hashtag/security?source=feed_text&epa=HASHTAG) [#Interview](https://www.facebook.com/hashtag/interview?source=feed_text&epa=HASHTAG) [#Questions](https://www.facebook.com/hashtag/questions?source=feed_text&epa=HASHTAG) [#answers](https://www.facebook.com/hashtag/answers?source=feed_text&epa=HASHTAG) 2019  
Question 1. What Is The Delegating Filter Proxy?  
Answer :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.DelegatingFilterProxy</filter-class>

</filter>

<filter-mapping>

<filter-name>myFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

Question 2. What Is The Security Filter Chain?

Answer :

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.springframework.security.web.FilterChainProxy">

<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>

Question 3. What Is Mandatory Filter Name Main Purpose?

Answer :

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

Question 4. Are You Able To Add And/or Replace Individual Filters?

Answer :

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.

Question 5. Is It Enough To Hide Sections Of My Output (e.g. Jsp-page)?

Answer :

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.

Question 6. Why Do You Need The Intercept-url?

Answer :

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.

Question 7. In Which Order Do You Have To Write Multiple Intercept-url’s?

Answer :

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\_A'/>

<intercept-url pattern='/secure/b/\*\*' access='ROLE\_B'/>

<intercept-url pattern='/secure/\*\*' access='ROLE\_USER'/>

Question 8. Why Do You Need Method Security? What Type Of Object Is Typically Secured At The Method Level.

Answer :

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)  
Question 9. Is Security A Cross Cutting Concern? How Is It Implemented Internally?

Answer :

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

Question 10. What Do @secured And @rolesallowed Do? What Is The Difference Between Them?

Answer :

@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.

Question 11. What Is A Security Context?

Answer :

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.

Question 12. How Is A Principal Defined?

Answer :

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().getAuthentication().getPrincipal();

if (principal instanceof UserDetails) {

String username = ((UserDetails)principal).getUsername();

} else {

String username = principal.toString();

}

Question 13. What Is Authentication And Authorization? Which Must Come First?

Answer :

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.

Question 14. In Which Security Annotation Are You Allowed To Use Spel?

Answer :

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="enabled"/>

@PreAuthorize("hasRole('USER')")

public void create(Contact contact);

Question 15. Does Spring Security Support Password Hashing? What Is Salting?

Answer :

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.

Question 16. Which Filter Class Is Needed For Spring Security?

Answer :

org.springframework.web.filter.DelegatingFilterProxy.

Question 17. What Are Access Controls In Spring Security?

Answer :

To access the account list, you must be authenticated.  
The files in the directory "/secure" should only be visible to authenticated users.  
The files in the directory "/secure/extreme" should only be visible to Supervisors.  
Withdrawal and deposits can be made only by Tellers and Supervisors.  
Overdraft limit for an account can be exceeded only by Supervisors.  
Question 18. How To Restrict Static Resources Processed By Spring Security Filters?

Answer :

< http pattern="/static/\*\*" security="none" / >

Question 19. From The Applications Perspective, How Many User Roles Needed In Spring Security?

Answer :

Three user roles are there in spring.

Supervisors  
Tellers  
Plain Users  
Question 20. Will Spring Security Secures All The Applications?  
Answer :No, in web application, we need to do some more things to secure full application to save from attackers.  
Question 21. How To Add Security To Method Calls Made On Spring Beans In The Application Context?  
Answer :-< global-method-security pre-post-annotations="enabled" / >  
Question 22. Which Java And Spring Version Are Needed For Spring Security?

Answer :

Spring security 3.0 and jdk 1.5.

Question 23. What Are All Security Layers In Spring Security Framework?

Answer :

Authentication:  
Web request security  
Service layer and domain object security  
Question 24. When I Login In The Application Where Spring Security Is Applied And Got The Messages "bad Credentials". What Is Wrong?

Answer :

Authentication has failed for the given userid and password.

Question 25. When I Try To Login, Application Goes In Endless Loop. What Is Wrong?

Answer :

It happens when login page is secured resource. Login page should not be secured, it should be marked as ROLE\_ANONYMOUS.