1. **What is Spring Security?** Spring security is a customizable framework to provide authentication (establishing that user/device credentials are valid) and authorization (provide access-control by deciding if the user/device can perform an action within the application) for J2EE applications.
2. **How is Spring security implemented?** Security in Spring is enabled using URL filtering through a DelegatingFilterProxy which is provided by the Spring framework. In Spring, the filter classes are Spring beans that implements Filter interface, which is defined in the application context, thereby making use of its advanced features such as dependency-injection and auto-configuration. The following is added to your web.xml, to enable Spring security.

|  |
| --- |
| <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> |

1. **Name some features of Spring Security.** Spring Security provides support for both Authentication and Authorization. It supports protection against cyber-attacks. Spring security provides Servlet API integration with Spring Web MVC.
2. **What is OAuth?** OAuth is an open standard that provides Authorization for third party services to access their server resources without exposing the user’s credentials.
3. **Explain salting in Spring security? What are the two methods of salting?** Salting is a password hashing mechanism which secures your Java application by adding an extra string to the password, thereby making it difficult for a hacker to crack passwords and preventing dictionary attacks. The extra string added to the password is called “the salt”. Using a salt means that an attacker has to build a separate dictionary of hashes for each salt value, making the attack complicated. **There are two salting methods**, namely, **Global Salt –** one common word is appended to all the passwords and **per User Salt** – one user attribute (user’s id, email address, etc.) serves as the Salt string for that user’s password.
4. **What is security filter chain?** Spring Security maintains a security filter chain internally where filters can be added/removed from the configuration depending on the services required and each of the filters has a particular responsibility.
5. **Can you name few filters in security filter chain and state the main purpose of each?**

* **SecurityContextIntegrationFilter –** establishes SecurityContext and maintains it between HTTP requests.
* **LogoutFilter –** Clears the SecurityContextHolder when logout is requested.
* **UsernamePasswordAuthenticationFilter –** Adds Authentication to SecurityContext when login is requested.
* **ExceptionTranslationFilter –** converts Spring Security exceptions to HTTP response.
* **FilterSecurityInterceptor –** Performs security handling of HTTP resources via a filter implementation and authorizes requests based on authorities.

1. **How would you configure Spring security ?** Spring security can be configured by editing web.xml or by in extending the WebSecurityConfigurerAdapter implementation. In both methods, the providers for authentication and authorization can be defined as well as descriptions of application scopes that need authentication and/ or authorization. The user can specify a custom form for login or fallback to the standard login screen provided by Spring.
2. **What is the minimum java and spring version required for spring security?** Minimum version required to enable spring security is Spring security 3.0 and jdk 1.5.
3. **Can individual filters be added while configuring Spring security?** Yes. Based on the services required, filters can be added or replaced, as each filter has a specific responsibility in the filter chain.
4. **What is intercept-url?** The intercept-url is used to define a set of url patterns for the filter chain to intercept and handle. It can be configured in the web.xml file as part of the application. Every HTTP request sent to any URL that matches the intercept-url pattern is first passed to the filter chain. Using this method any application scope that needs to be secured can be intercepted and passed to the spring security filter chain.
5. **What is @PreFilter and @PostFilter in spring security? What is the difference between them? @PreFilter** and **@PostFilter** are method-level annotations used in Spring security to filter collection or arrays on the basis of authorization to realize security rules. This can be achieved using expression-based access control to the elements of the collections that are sent to the method as a parameter or are returned by the method.
6. **The difference between them is: @PreFilter** filters the collection or arrays before executing the method. **@PostFilter** filters the returned collection or arrays after executing the method.
7. **What is the Intercepting filter design pattern?** Intercepting filter pattern is a design pattern used widely in Spring security. For example, the filter and intercept-url configuration in the web.xml result in Spring security filter chain setup with a FilterManager that is responsible for maintaining the list of active filters and for routing any requests to the intercepted URLs through the filter chain relevant to it along with the appropriate session context. This pattern is also extended to the class/ method level via spring security annotations such as **@Secured, @EnableWebSecurity, @PreAuthorize, @PreFilter & @PostFilter etc.,.**
8. **Is it possible to encode password in Spring Security using XML?** Yes, it is possible. Spring security provides the <password-encoder/> tag is used to encode password. This interface supports the use of passwords which are encoded using a digest algorithm such as MD5 or SHA.
9. **How to configure channel security in Spring?** Spring security provides feature to configure allowing a URL pattern to be accessed over HTTPS/HTTP channel, thereby securing the URL pattern. This can be done by setting the requires-channel attribute in the <intercept-url> to the preferred channel name. The value of this attribute can be set to “any” if there is no preference. For example, the following configuration is added to allow a URL pattern to be accessed over HTTPS only:

|  |
| --- |
| <intercept-url pattern="/login" access="ROLE\_USER" requires-channel="https" /> |

1. **What is the role of Authentication Manager in Spring security?** In Spring Security, the authentication manager is the main interface that provides authentication services and assumes the job of establishing a user’s identity. The authenticate method will try to authenticate the user using the org.springframework.security.core.Authentication object (which carries the principal and credentials). If authentication is successful, it returns an Authentication object, complete with information about the granted authorities for the user. If, however, the authentication fails, the method throws an authentication exception. AuthenticationManager is an instance of Spring Security’s ProviderManager class and can be created using the AuthenticationManagerBuilder instance that is available as part of the application context.
2. **What is SecurityContextHolder?** SecurityContextHolder stores request-specific security information and the details of the present security context and includes the details of the principal which is currently interacting with the application.
3. **How is Remember-Me authentication accomplished in Spring security?** Remember-me authentication refers to web sites being able to remember the identity of a principal between sessions. This is typically accomplished by sending a cookie to the browser which is detected during future sessions and causing automated login. Spring Security has two concrete remember-me implementations:

* Using hashing to preserve the security of cookie-based tokens
* Using a database or other persistent storage mechanism to store the generated tokens.

1. **Which filter handles session management in Web Application security?** SessionManagementFilter is the filter used to handle session management and it checks if a user has been authenticated during the current request.
2. **How will you use web security expressions in web.xml? Give an example.** To use web security expressions, you would first need to set the use-expressions attribute in the <http> element to true. Next, the access attributes of the <intercept-url> elements is configured to contain Spring EL expressions. For example, the expression used to make the endpoint /admin accessible to users who have the role “admin” and whose IP address matches a local subnet is:

|  |
| --- |
| <http use-expressions="true">  <intercept-url pattern="/admin\*"  access="hasRole('admin') and hasIpAddress('192.168.1.0')"/>  </http> |

1. **What is the difference between IsAuthenticated() and isFullyAuthenticated()? isFullyAuthenticated()** returns true only if the user has been authenticated using their credentials. It returns false for a remember-me user or an anonymous user. **IsAuthenticated()** returns true if the user is not an anonymous user even if the user was authenticated based on a remember-me authentication method.
2. **What is the difference between ROLE\_USER and ROLE\_ANONYMOUS while configuring intercept url? ROLE\_ANONYMOUS** is enabled by default. It is the role assigned to an anonymous user when a configuration uses the “anonymous authentication” filter. **ROLE\_USER** has meaning, only if you assign this role to your users after authenticating.
3. **What is CSRF? Why should you disable CSRF in Spring security?** CSRF stands for Cross-Site Request Forgery and is also known as one-click attack. It is a type of malicious exploit of a website which tricks the end users to send unauthorized commands to the web application to which the user is authenticated. To prevent this any requests coming to the web-app from any other domains are blocked. However, this problem can also be solved by sending a unique token to the request which has been generated and is stored in the http session, which, any attacker will find hard to duplicate. Spring security provides the capability to auto generate and validate such a token. Hence, CSRF can be disabled in the http security configuration. Also, CSRF protection can be disabled for services that will be entirely consumed by non-browser clients since CSRF attack is not relevant unless a user is accessing the app using a browser.
4. **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.
5. **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.
6. **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\_A'/>
* <intercept-url pattern='/secure/b/\*\*' access='ROLE\_B'/>
* <intercept-url pattern='/secure/\*\*' access='ROLE\_USER'/>

1. **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)

1. **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.
2. **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.
3. **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.
4. **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().getAuthentication().getPrincipal();  if (principal instanceof UserDetails) {  String username = ((UserDetails)principal).getUsername();  } else {  String username = principal.toString();  } |

1. **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.

1. **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="enabled"/>  @PreAuthorize("hasRole('USER')")  public void create(Contact contact); |

**35. Which Filter Class Is Needed For Spring Security?** org.springframework.web.filter.DelegatingFilterProxy.

**36. What Are Access Controls In Spring Security?**

* 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.

**37. How To Restrict Static Resources Processed By Spring Security Filters?**

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

**38. From The Applications Perspective, How Many User Roles Needed In Spring Security?**

* Supervisors
* Tellers
* Plain Users

**39. Will Spring Security Secures All The Applications?** No, in web application, we need to do some more things to secure full application to save from attackers.

**40. How To Add Security To Method Calls Made On Spring Beans In The Application Context?**

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

**41. Which Java And Spring Version Are Needed For Spring Security?** Spring security 3.0 and jdk 1.5.

**42. What Are All Security Layers In Spring Security Framework?**

* Authentication:
* Web request security
* Service layer and domain object security

**43. When I Login In The Application Where Spring Security Is Applied And Got The Messages "bad Credentials". What Is Wrong?** Authentication has failed for the given userid and password.

**44. When I Try To Login, Application Goes In Endless Loop. What Is Wrong?** It happens when login page is secured resource. Login page should not be secured, it should be marked as ROLE\_ANONYMOUS.

**45. What is security principal?** SecurityContextHolder stores the principal currently interacting with the application. The principal is the currently logged in user that you retrieve it through the security context.

|  |
| --- |
| Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();  if (principal instanceof UserDetails) {  String username = ((UserDetails)principal).getUsername();  } else {  String username = principal.toString();  } |

**46. How do I enable Spring Security in Java Web application?** To enable Spring security in Java Web application, you need to do configure three things,

* declare a delegating proxy filter in web.xml,
* add ContextLoaderListener in web.xml,
* and provide actual security constraints on applicationContext-Security.xml file.
* Since Spring security uses a chain of filters to implement various security constraints, also known as security chain filter, it relies on web container for the initialization of delegating filter proxy.

**47. Which filter class is required for spring security?** The DelegatingFilterProxy class from package org.springframework. web.filter is required.

**48. Types of authentication that spring supports.**

* HTTP Basic authentication,
* HTTP digest,
* Form based,
* Using LDAP,
* Using LDAP,
* Using LDAP,
* OAUTH,
* Automatic remember me authentication.

**49. Explain BASIC authentication.** Basic authentication is a simple authentication scheme built into the HTTP protocol. The client sends HTTP requests with the Authorization header that contains the word Basic word followed by a space and a base64-encoded string username:password.

**50. Explain digest authentication.** Digest authentication is an application of MD5 cryptographic hashing with usage of nonce values to prevent replay attacks. It uses the HTTP protocol.

**51. How to restrict static resources using spring security?** The Ant matchers match against the request path and not the path of the resource on the filesystem.So ignore any request that starts with "/resources/".This is similar to configuring http@security=none when using the XML namespace configuration.

|  |
| --- |
| @Override  public void configure(WebSecurity web) throws Exception {  web  .ignoring()  .antMatchers("/resources/\*\*");  } |

**52. Is there a way to set up basic authentication and form login in same application?** Yes. We may need form login for web app and basic for rest services. In that case multiple http configuration is required.

**53. What is JCA in Java?** Java Cryptography Architecture implements security functions for the Java platform. It provides a platform and gives architecture and APIs for encryption and decryption. JCA is used by the developer to combine the application with the security measure. A programmer uses the JCA to meet the security measure. It helps in performing the third-party security rules. It uses the hash table, encryption message digest, etc to implement the security.

**54. Explain mutual authentication.** Mutual authentication, also called two-way authentication, is a process or technology in which both entities in a communications link authenticate each other.

**55. Name an alternative to Spring security.** Apache Shiro is an open-source software security framework that performs authentication, authorization, cryptography and session management. Shiro is an intuitive and easy-to-use framework with robust security features.

**56. How to use Form Login Authentication using Spring Boot**? We make use of Spring Boot Security to get default login page and authentication users.

|  |
| --- |
| @Override  protected void configure(HttpSecurity http) throws Exception {  http.authorizeRequests().antMatchers("/").permitAll().antMatchers("/welcome")  .hasAnyRole("USER", "ADMIN").antMatchers("/getEmployees").hasAnyRole("USER", "ADMIN")  .antMatchers("/addNewEmployee").hasAnyRole("ADMIN").anyRequest().authenticated().and().formLogin()  .permitAll().and().logout().permitAll();  http.csrf().disable();  } |

**57. How to create Custom Login Page using Spring Boot Security?** We can create our own custom login page and use it for authentication.

|  |
| --- |
| @Override  protected void configure(HttpSecurity http) throws Exception {  http.authorizeRequests().antMatchers("/").permitAll().antMatchers("/welcome")  .hasAnyRole("USER", "ADMIN").antMatchers("/getEmployees").hasAnyRole("USER", ADMIN")  .antMatchers("/addNewEmployee")  .hasAnyRole("ADMIN").anyRequest().authenticated()  .and().formLogin().**loginPage("/login").**permitAll()  .and().logout().permitAll();  http.csrf().disable();  } |

**58. How to configure Spring Security with in-memory configuration?**

|  |
| --- |
| @Autowired  public void configureGlobal(AuthenticationManagerBuilder auth)  throws Exception {  auth.inMemoryAuthentication()  .withUser("user").password("password").roles("USER")  .and()  .withUser("admin").password("password").roles("USER", "ADMIN"); |

**59.** **What is the use of Spring Boot Security AuthenticationHandler class?** In some scenarios we might want to redirect different users to different pages depending on the roles assigned to the users. For example we might want users with role USER to be redirected to the welcome page, while users with role ADMIN to be redirected to the add employee page. We will be making use of the AuthenticationSuccessHandler.

|  |
| --- |
| @Override  protected void configure(HttpSecurity http) throws Exception {  http.authorizeRequests().antMatchers("/").permitAll().antMatchers("/welcome")  .hasAnyRole("USER", "ADMIN")  .antMatchers("/getEmployees").hasAnyRole("USER", "ADMIN")  .antMatchers("/addNewEmployee")  .hasAnyRole("ADMIN").anyRequest().authenticated()  .and().formLogin().**successHandler(successHandler**)  .loginPage("/login").permitAll().and().logout().permitAll();  http.csrf().disable();  } |