What is Spring Security?

Spring Security provides **comprehensive security services** for Java EE-based enterprise software applications.

There are **two main areas** that Spring Security targets. **"Authentication"**is the process of establishing a principal is who they claim to be (a "principal" generally means a user, device or some other system which can perform an action in your application).**"Authorization"** refers to the process of deciding whether a principal is allowed to perform an action within your application.

What is Oauth?

OAuth is an open standard for authorization. OAuth provides client applications a 'secure delegated access' to server resources on behalf of a resource owner. It specifies a process for resource owners to authorize third-party access to their server resources without sharing their credentials.

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.

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();

}

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.

Which filter class is required for spring security?

The DelegatingFilterProxy class from packageorg.springframework. web.filter is required.

Minimum java and spring version required for spring security?

Spring security 3.0 and jdk 1.5.

Mention other filters in spring security and its purpose.

**SecurityContextIntegrationFilter**: establishes SecurityContext and maintains between HTTP requests.

**LogoutFilter**: clears SecurityContextHolder when logout requested.

**UsernamePasswordAuthenticationFilter**: places Authentication into the SecurityContext on login request.

**ExceptionTranslationFilter**: converts SpringSecurity exceptions into HTTP response or redirect.

**FilterSecurityInterceptor**: authorize web requests based on config attributes and authorities.

Types of authentication that spring supports.

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

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**.

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.

Does Spring Security support password hashing?

Yes, Spring Security provides support for passwordhashing.

What is salting in spring security?

Salting secure your application from Dictionary-Attack. Using Salt you may add an extra string in password so hacker find it difficult for braking the password.

There are 2 salt methods,

* Global Salt.
* Per User Salt.

In Global Salt there is one single common wordappend to password. In Per User Salt we have to give one user attribute serve as Salt String.

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/\*\*");

}

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.

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.

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.

**How is Security mechanism implemented using Spring?**

Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications. Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. Like all Spring projects, the real power of Spring Security is found in how easily it can be extended to meet custom requirements.   
**Spring makes use of the DelegatingFilterProxy for implementing security mechanisms.** It is a Proxy for standard Servlet Filter, delegating to a Spring-managed bean that implements the Filter interface. Its the starting point in the springSecurityFilterChain which instantiates the Spring Security filters according to the Spring configuration  
Some of the features of Spring Security are

* Comprehensive and extensible support for both Authentication and Authorization
* Protection against attacks like session fixation, clickjacking, cross site request forgery, etc
* Servlet API integration Optional integration with Spring Web MVC

**What is OAuth2? How to implement it using Spring Boot Security?**

OAuth (Open Authorization) is a simple way to publish and interact with protected data.  
It is an open standard for token-based authentication and authorization on the Internet. It allows an end user's account information to be used by third-party services, such as Facebook, without exposing the user's password.  
The OAuth specification describes five grants for acquiring an access token:

* Authorization code grant
* Implicit grant
* Resource owner credentials grant
* Client credentials grant
* Refresh token grant

Consider the use case of Quora. Go to Quora.com.  
If you are a new user you need to signup. You can signup using google or facebook account. When doing so you are authorizing Google or Facebook to allow quora to access you profile info with Quora. **This authorizing is done using OAuth**. Here you have in no way shared your credentials with Quora.

**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();

}

**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();

}

**How to do authentication against database tables using Spring Boot Security?**

Spring Authentication using username, password and authorization using roles can be done using either

* In Memory Configuration -
* @Autowired
* public void configureGlobal(AuthenticationManagerBuilder authenticationMgr) throws Exception {
* authenticationMgr.inMemoryAuthentication().withUser("employee").password("employee")
* .authorities("ROLE\_USER").and().withUser("javainuse").password("javainuse")
* .authorities("ROLE\_USER", "ROLE\_ADMIN");
* }

Database Authentication- 

@Autowired

public void configAuthentication(AuthenticationManagerBuilder auth) throws Exception {

auth.jdbcAuthentication().dataSource(dataSource);

}

**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");

**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();

}

**What is the difference between ROLE\_USER and ROLE\_ANONYMOUS in a Spring intercept url configuration?**

* **ROLE\_ANONYMOUS** is the default role assigned to an unauthenticated (anonymous) user when a configuration uses Spring Security's "anonymous authentication" filter . This is enabled by default. However, it is probably clearer if you use the expression isAnonymous() instead, which has the same meaning.
* **ROLE\_USER** has no meaning unless you assign this role to your users when they are authenticated (you are in charge of loading the roles (authorities) for an authenticated user). It isn't a name that is built in to Spring Security's infrastructure. In the given example, presumably that role is assigned to an authenticated user.

**How to configure DelegatingFilterProxy ?**

In the web.xml we add the DelegatingFilterProxy which is delegating proxy to automatically intercept a URL with a particular pattern to apply 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>

**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,ROLE\_B'/>

**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'/>

**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.