**🔐 Spring Security Interview Questions and Answers:**

**✅ Basic Level**

**1. What is Spring Security?**  
Spring Security is a powerful and customizable framework for authentication, authorization, and protection against common security attacks (like CSRF, XSS, etc.) in Java applications. It is part of the Spring ecosystem.

**2. How does Spring Security work?**  
Spring Security uses filters in the filter chain to intercept HTTP requests. It checks:

* If the request is authenticated
* If the user has the right permissions/roles
* Then allows or denies access

**3. What are the default login credentials in Spring Security?**

* Default user: user
* Password: A random password printed in the console when the app starts  
  (Only if you haven’t defined your own user)

**4. How do you define custom login credentials?**  
Using application.properties:

spring.security.user.name=admin

spring.security.user.password=admin123

**🔒 Authentication & Authorization**

**5. What is the difference between Authentication and Authorization?**

* **Authentication**: Who you are (login)
* **Authorization**: What you're allowed to access (roles/permissions)

**6. How do you implement role-based access control (RBAC) in Spring Security?**

http.authorizeRequests()

.antMatchers("/admin/\*\*").hasRole("ADMIN")

.antMatchers("/user/\*\*").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated();

**7. How do you secure a REST API in Spring Boot?**

* Use HttpSecurity to define which endpoints need authentication
* Example:

http.csrf().disable()

.authorizeHttpRequests()

.requestMatchers("/api/public").permitAll()

.requestMatchers("/api/private").authenticated()

.and()

.httpBasic();

**🔑 JWT (JSON Web Token)**

**8. What is JWT and how is it used in Spring Security?**  
JWT is a stateless way to secure APIs. Once a user logs in:

* Server generates a signed token
* Client stores the token (usually in localStorage/cookie)
* For every API call, client sends this token in the Authorization header
* Server validates the token and grants/denies access

**9. Where do you store the JWT token in the frontend?**

* In the **Authorization Header** using:

Authorization: Bearer <token>

* On client-side: In **localStorage** or **cookies**

**10. How do you validate a JWT token in Spring Security?**

* Parse the token using a JWT library like **jjwt**
* Validate:
  + Signature
  + Expiry
  + Subject/Claims

**🛡️ Advanced Topics**

**11. What is the use of @PreAuthorize and @Secured?**

* @PreAuthorize("hasRole('ADMIN')") checks before entering the method
* @Secured("ROLE\_ADMIN") also limits access at the method level

**12. What is CSRF and how does Spring Security handle it?**

* **CSRF (Cross-Site Request Forgery)** is an attack that tricks a user into executing unwanted actions.
* Spring Security enables CSRF protection by default for web apps.
* For REST APIs, you often disable it using:

http.csrf().disable();

**13. What is the DelegatingFilterProxy?**  
It is a servlet filter that delegates to a Spring bean that implements Filter. It connects the servlet container's filter chain with Spring Security's filter chain.

**14. How can you implement OAuth2 in Spring Boot?**

* Use spring-security-oauth2 dependency
* Configure client details and providers (like Google, GitHub)
* Secure endpoints with .oauth2Login() and scopes

**🧪 Testing Security**

**15. How do you test secured endpoints in Spring Boot?**

* Use @WithMockUser in test methods

@WithMockUser(username = "admin", roles = {"ADMIN"})

@Test

public void testSecureEndpoint() throws Exception {

mockMvc.perform(get("/admin/dashboard"))

.andExpect(status().isOk());

}

**JWT Interview Questions and Answers :**

**✅ 1. What is JWT?**

**Answer:**  
JWT (JSON Web Token) is a compact, URL-safe means of representing claims to be transferred between two parties. It is commonly used for **authentication and authorization** in stateless applications like RESTful APIs.

**✅ 2. What are the components of a JWT token?**

**Answer:** A JWT has three parts:

1. **Header** – Contains the token type and signing algorithm (e.g., HS256)
2. **Payload** – Contains claims like username, roles, expiration time
3. **Signature** – Used to verify the integrity and authenticity of the token

Example:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. --> Header

eyJzdWIiOiJ1c2VyQGVtYWlsLmNvbSIsInJvbGUiOiJVU0VSIiwiZXhwIjoxNzAwMDAwMDB9 --> Payload

SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c --> Signature

**✅ 3. Why is JWT considered stateless?**

**Answer:**  
JWT tokens store all user data (claims) inside the token itself and are **self-contained**. The server doesn’t need to store session information — making it **stateless and scalable**.

**✅ 4. Where do you store JWT on the client side?**

**Answer:**

* **LocalStorage** – Easy to use, but vulnerable to XSS
* **SessionStorage** – Clears on tab close
* **HttpOnly Cookie** – More secure; avoids XSS, but needs CSRF protection

**✅ 5. What happens when JWT expires?**

**Answer:**  
The token becomes invalid. Any request with an expired token gets rejected with 401 Unauthorized. You then:

* Re-authenticate the user
* Or use a **refresh token** mechanism to issue a new access token

**✅ 6. How do you secure a REST API using JWT in Spring Boot?**

**Answer:**

1. Authenticate user and generate JWT
2. Store it client-side
3. Send it in Authorization header with every request
4. Validate token in Spring Security filter
5. If valid, allow access to secured endpoints

**✅ 7. How do you validate JWT in Spring Boot?**

**Answer:**  
Use a library like jjwt to:

* Verify the **signature**
* Check **expiration time**
* Extract **claims** to get the user info or roles

**✅ 8. What are the types of claims in JWT?**

**Answer:**

* **Registered claims** – Standard ones like iss, exp, sub, etc.
* **Public claims** – Custom claims with public meaning
* **Private claims** – Custom claims shared between parties (e.g., userId, roles)

**✅ 9. Can JWT be tampered with?**

**Answer:**  
Yes, if not signed. But when signed with a **secret** (HMAC) or **private key** (RSA), tampering changes the signature — making the token invalid.

**✅ 10. What are the common algorithms used in JWT?**

**Answer:**

* **HMAC** – HS256 (shared secret)
* **RSA / ECDSA** – RS256 (asymmetric keys: private/public)

**✅ 11. Difference between access token and refresh token?**

| **Access Token** | **Refresh Token** |
| --- | --- |
| Short-lived | Long-lived |
| Used to access resources | Used to get a new access token |
| Sent on every request | Sent only to refresh token endpoint |

**✅ 12. How do you revoke a JWT token?**

**Answer:**

* JWTs are **stateless**, so you can't revoke them server-side directly
* Workarounds:
  + Use **token blacklist**
  + Maintain a token version or issue time in DB
  + Use **short expiry** + **refresh token** pattern

**13 .How do you enable method-level security in Spring Security?**

@EnableMethodSecurity(prePostEnabled = true)

Use annotations like:

@PreAuthorize("hasRole('ADMIN')")

@Secured("ROLE\_USER")

**14. How does Spring Security handle password encryption?**  
**Answer:**  
It uses PasswordEncoder interface. Use BCryptPasswordEncoder for strong hashing:

PasswordEncoder encoder = new BCryptPasswordEncoder();