Spring Security - Practice Sheet

10 Q&A on filter chain, JWT, OAuth2, CSRF, and method security

Q1. What is the Spring Security filter chain and how does it process requests?

A: Spring Security registers a chain of servlet filters via DelegatingFilterProxy. Each filter handles a concern: SecurityContextPersistenceFilter restores session state, UsernamePasswordAuthenticationFilter processes form logins, and

Q2. What is the difference between Authentication and Authorization?

A: Authentication verifies who the user is (username/password, JWT, OAuth2 token). Authorization decides what they can do (can this principal access this resource?). Spring Security does auth first, then authz on the resolved principal.

Q3. How do you implement stateless JWT authentication in Spring Security?

A: Disable sessions (SessionCreationPolicy.STATELESS), add a custom OncePerRequestFilter that parses the JWT, builds a UsernamePasswordAuthenticationToken, and stores it in SecurityContextHolder. Protect routes with .authorizeHttpRequests().

Q4. What does UserDetailsService do in Spring Security?

A: UserDetailsService.loadUserByUsername(username) returns a UserDetails object containing the username, hashed password, and granted authorities. DaoAuthenticationProvider calls it during form-login to authenticate against a database user.

Q5. What are hasRole(), hasAuthority(), and isAuthenticated() security expressions?

A: hasRole('ADMIN') checks for ROLE_ADMIN in GrantedAuthorities (Spring auto-prefixes). hasAuthority('SCOPE_read') checks the exact authority. isAuthenticated() is true for any non-anonymous principal. Use them in @PreAuthorize or requestMatchers.

Q6. How does method-level security work in Spring Security?

A: Add @EnableMethodSecurity to a config class. Annotate service methods with @PreAuthorize('hasRole("ADMIN")') or @PostAuthorize. Spring wraps the bean in a proxy that evaluates the SpEL expression before or after the method runs.

Q7. What is an OAuth2 Resource Server and how do you configure one?

A: A Resource Server validates Bearer tokens on incoming requests. Configure it with .oauth2ResourceServer(oauth2 -> oauth2.jwt(...)) and spring.security.oauth2.resourceserver.jwt.issuer-uri pointing to your authorization server for JWK key discovery.

Q8. How do you configure CORS in a Spring Security application?

A: Define a CorsConfigurationSource bean or use @CrossOrigin, then call .cors(cors -> cors.configurationSource(source)) in the security filter chain BEFORE other filters. Without this, Spring Security blocks CORS preflight OPTIONS requests.

Q9. What is CSRF and when can you safely disable it?

A: CSRF forges state-changing requests using a victim's authenticated session. Disable it only for stateless REST APIs that authenticate via headers (JWT, API keys) rather than cookies browsers cannot set custom headers from attacker pages.

Q10. How do you test Spring Security with @WithMockUser?

A: @WithMockUser('user') populates the SecurityContext with a mock principal for the duration of the test. @WithSecurityContext lets you build a fully custom context. Use MockMvc with springSecurity() configurer to apply filter chain logic.