CODING SHUTTLE

Spring Security

Internal working of Spring Security

Adding Spring Security

In a Spring boot application, we only need to include the spring-boot-starter-security dependency and Spring boot auto-configured the security with sensible defaults defined in WebSecurityConfiguration class.

```xml
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Authentication and Authorization

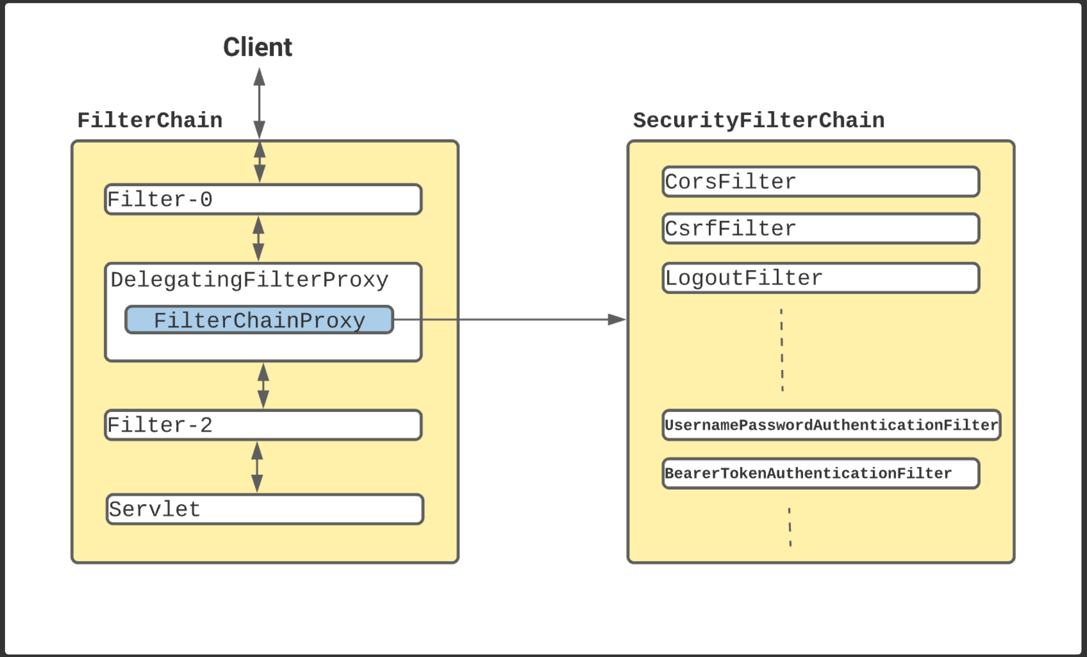
Authentication is the process of verifying the identity of a user. It ensures that the user is who they claim to be. Authentication typically involves validating credentials, such as a username and password, and creating a security context for the user.

Authorization is the process of determining whether an authenticated user has the necessary permissions to access a particular resource or perform an action. It controls what an authenticated user can or cannot do.

Internal working of Spring-Security

- In a Spring Boot application, SecurityFilterAutoConfiguration automatically registers the DelegatingFilterProxy filter with the name springSecurityFilterChain.

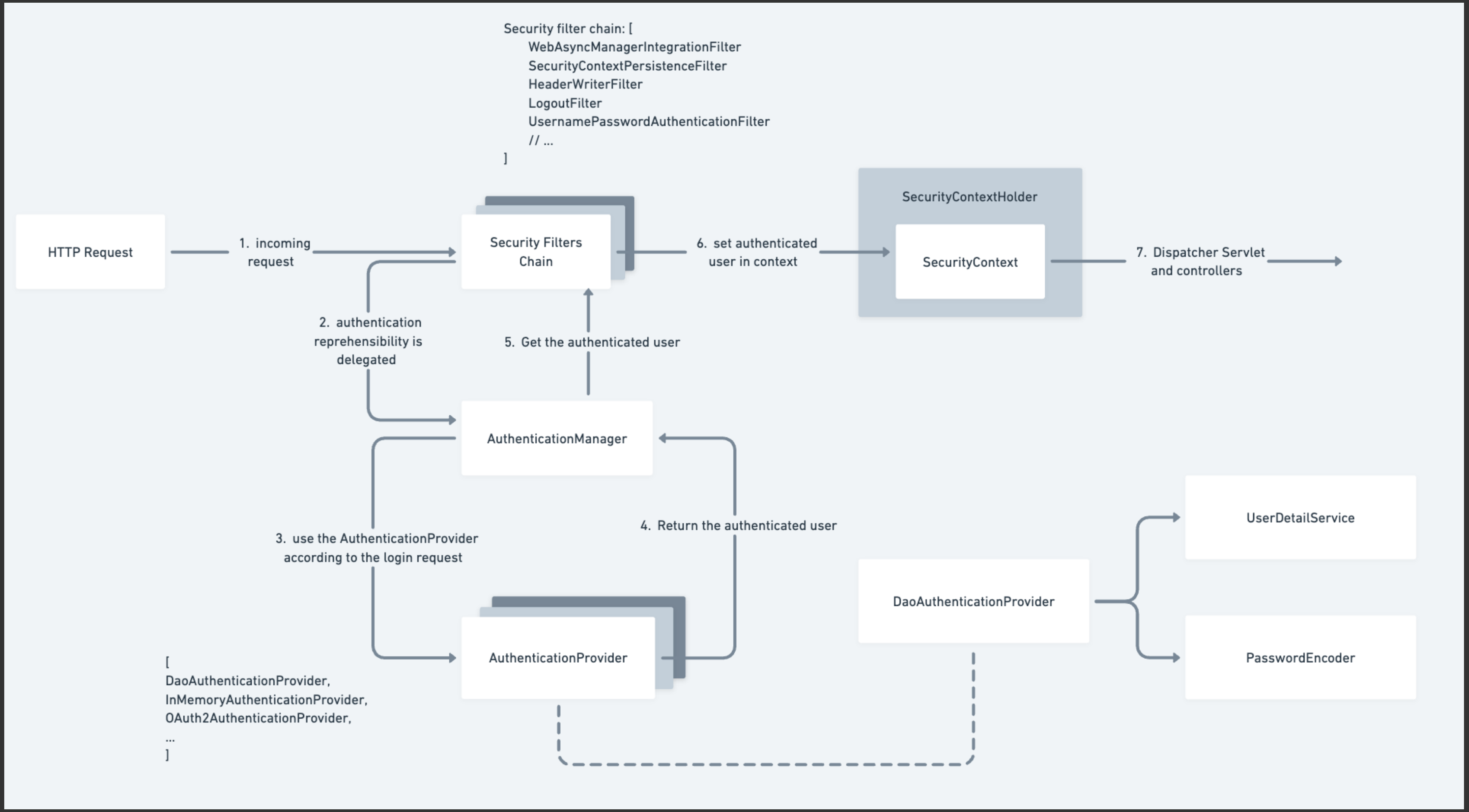
- Once the request reaches to DelegatingFilterProxy, Spring delegates the processing to FilterChainProxy bean that utilizes the SecurityFilterChain to execute the list of all filters to be invoked for the current request.



Default behaviour of Spring-Security

- Creates a bean named springSecurityFilterChain. Registers the Filter with a bean named springSecurityFilterChain with the Servlet container for every request.
- HTTP basic authentication for authenticating requests made with remoting protocols and web services.
- Generate a default login form.
- Creates a user with a username of user and a password that is logged to the console.
- Protects the password storage with BCrypt.
- Enables logout feature.
- Other features such as protection from CSRF attacks and session fixation

Internal Flow of Spring-Security



Security filter chain: [
WebAsyncManagerIntegrationFilter
SecurityContextPersistenceFilter
HeaderWriterFilter
LogoutFilter
UsernamePasswordAuthenticationFilter
// ...
]
HTTP Request
1. incoming request
Security Filters Chain
6. set authenticated user in context
SecurityContextHolder
SecurityContext
7. Dispatcher Servlet and controllers
2. authentication reprehensibility is delegated
5. Get the authenticated user
AuthenticationManager
3. use the AuthenticationProvider according to the login request
4. Return the authenticated user
AuthenticationProvider
[
DaoAuthenticationProvider,
InMemoryAuthenticationProvider,
OAuth2AuthenticationProvider,
...
]
DaoAuthenticationProvider
UserDetailService
PasswordEncoder