Spring Security

Spring Security

Spring Security is a framework that focuses on providing both authentication and authorization (access control) to java web application and SOAP/REST web services.

Spring framework supports integration with many technologies

- HTTP basic authentication
- LDAP
- OpenID providers
- JAAS API
- And customized authentication system (by yourself)

Spring Security

Terminologies

Principal

User that performs the action

Authentication

Confirming truth of credentials

Authorization

Define access policy for principal

GrantedAuthority

Application permission granted to a principal

SecurityContext

Hold the authentication and other security information

SecurityContextHolder

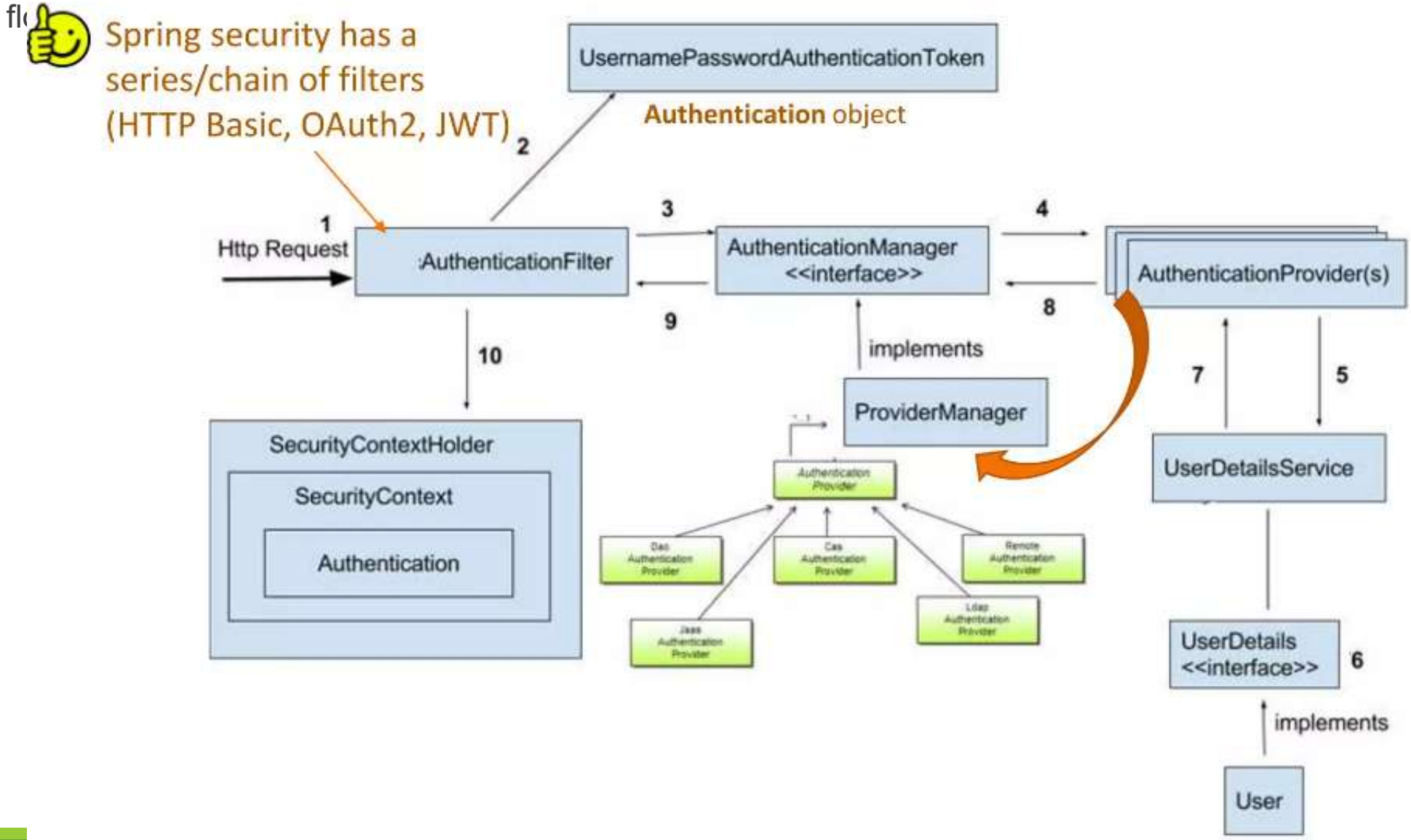
Provides access to SecurityContext

Spring Security

Terminologies

- AuthenticationManager
  - Controller in the authentication process
- AuthenticationProvider
  - Interface that maps to a data store which stores your user data.
- Authentication Object
  - Object is created upon authentication, which holds the login credentials.
- UserDetails
  - Data object which contains the user credentials, but also the Roles of the user.
- UserDetailsService
    - Collects the user credentials, authorities(roles) and build an UserDetails object.

Spring Security



Spring Security

In Spring 6.x, by adding the dependency to project, when you access any resources, you should provide the credential.

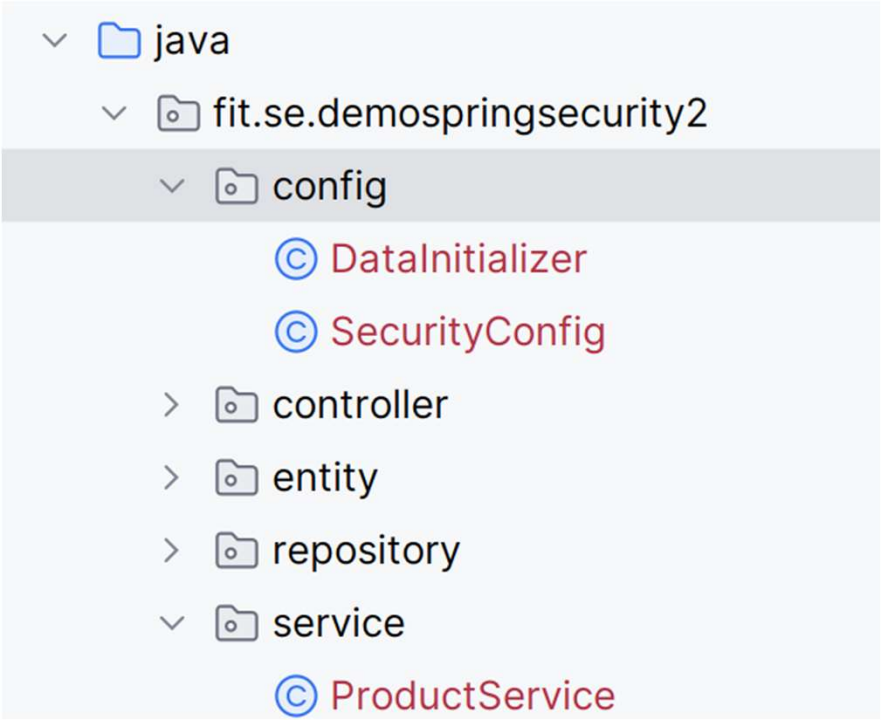
To manage the resources grant/deny permission, you should configure them using XML or by code.

There are many way to implements the security in Spring.

Spring Security

Dependency

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


java
fit.se.demospringsecurity2
config
DataInitializer
SecurityConfig
controller
entity
repository
service
ProductService

Spring Security

Configuration

Sample steps:

1. Create any class with the @Configuration annotation.
2. Add @EnableWebSecurity

```
@Configuration
@EnableWebSecurity
@EnableMethodSecurity(prePostEnabled = true)
public class SecurityConfig {
```

Spring Security

Configuration

Sample steps:

3. Create a bean with return type SecurityFilterChain and a parameter with type HttpSecurity

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http)
    ...
```

```
.authorizeHttpRequests(authz -> authz
        .requestMatchers("/login", "/css/**", "/js/**", "/images/**",
"/test").permitAll()
        .requestMatchers("/", "/products").hasAnyRole("ADMIN",
"CUSTOMER")
        .requestMatchers("/products/view/**").hasAnyRole("ADMIN",
"CUSTOMER")
        .requestMatchers("/products/add", "/products/edit/**",
"/products/delete/**").hasRole("ADMIN")
        .anyRequest().authenticated()
)
```

```
.formLogin(form -> form
        .loginPage("/login")
        .defaultSuccessUrl("/products", true)
        .permitAll()
)
.logout(logout -> logout
        .logoutUrl("/logout")
        .logoutSuccessUrl("/products")
        .invalidateHttpSession(true)// Xóa session
        .deleteCookies("JSESSIONID")//Xóa cookies
        .permitAll()
);
```

Spring Security

4. Controller: Setup @PreAuthorize("hasRole(‘ .... ‘)”) for method

```
@GetMapping("/view/{id}")
public String viewProduct(@PathVariable Long id, Model model) {
   Product product = productService.getProductById(id).orElse(null);
   if (product == null) {
      return "redirect:/products";
   }
   model.addAttribute("product", product);
   return "products/view";
}

@GetMapping("/add")
@PreAuthorize("hasRole('ADMIN')")
public String showAddForm(Model model) {
   model.addAttribute("product", new Product());
   return "products/add";
}
```

5. View: check for ROLE

```
<div sec:authorize="hasRole('ADMIN')">
  <a th:href="@{/products/edit/{id}(id=${product.id})}"
    class="btn btn-outline-warning btn-sm">
     <i class="fas fa-edit"></i> Edit
  </a>
  <a th:href="@{/products/delete/{id}(id=${product.id})}"
    class="btn btn-outline-danger btn-sm"
    onclick="return confirm('Are you sure you want to delete this
product?')">
     <i class="fas fa-trash"></i> Delete
  </a>
</div>
```