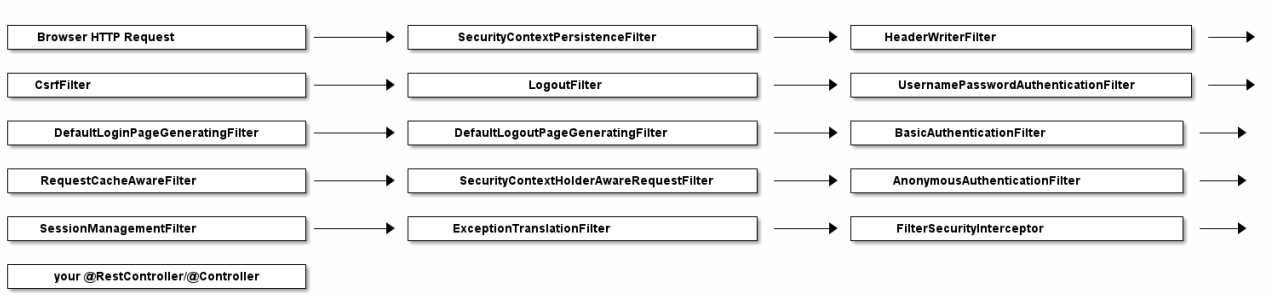
Spring security

Spring security é usado para autenticação e autorização de requests, na é basicamente um conjunto de filters, cada um com uma responsabilidade (filter de login, filter de autenticação, filter de autorização, etc), que são chamados em uma sequência formando o que é chamado de FilterChain.

Quando spring security é iniciado com configurações padrão ele carrega um total de 15 filters, sendo eles:



Os mais relevantes são:

- BasicAuthenticationFilter: Tenta encontrar Basic Auth no header Authorization, se encontrado tenta autenticar o usuário
- UsernamePasswordAuthenticationFilter: Tenta encontrar um usuário e senha no request param/body e se encontrar tenta usá-los para autenticação
- DefaultLoginPageGeneratingFilter: Gera uma página de login padrão (precisa ser desabilitado via config caso não seja desejado)
- DefaultLogoutPageGeneratingFilter: Gera uma página de logout, mesmo caso de DefaultLoginPageGeneratingFilter
- FilterSecurityInterceptor: Responsável por verificar a autorização do usuário logado

Esse é o core de spring-security, o resto é configurado via classes de configuração/XML, a config padrão é:

- Todas requests precisam de autenticação
- Página de login default habilitada
- Http basic habilitado

Autenticação

Autenticação com spring-security tem 3 cenários:

- A senha do usuário é acessível: O mais comum, onde se tem acesso aos detalhes do usuário (login e senha) em db local, nesse caso é usada a interface UserDetailsService

- A senha do usuário não é acessível: Normalmente a senha do usuário está em uma API externa e é necessário autenticá-lo por lá, nesse caso são usadas as interfaces AuthenticationProvider e AuthenticationManager
- OAuth2: A autenticação é feita via OAuth2 em uma outra plataforma

Autenticação com UserDetailsService

- UserDetails: É uma interface que define o contrato de uma classe que carrega os dados do usuário, além de ser possível implementar a interface na mão, ela possui as implementações padrão:
  - JdbcUserDetailsManager: É uma implementação baseada em JDBC, usada para mapear uma tabela SQL com os dados do usuário
  - InMemoryUserDetailsManager: Implementação com os dados do usuário em memória, é útil para testes
  - User: É uma implementação padrão que é construída usando dados de outro objeto, como uma entidade JPA.
- UserDetailsService: Uma interface que só possui um método obrigatório que é usado carregar os dados de usuário e retorna uma implementação de UserDetails
- PasswordEncoder: Interface que define o contrato para uma classe usada para criptografar e comparar as senhas salvas, o bean padrão usa Bcrypt

AuthenticationProvider e AuthenticationManager

- AuthenticationProvider: É uma interface que define o contrato para autenticar o usuário para um método de autenticação específico, ela possui os métodos:
  - supports: Recebe um objeto do tipo Class e verifica seu tipo para ver se ele é suportado no método authenticate (isso é chamado no AuthenticationManager)
  - authenticate: Recebe um objeto da que implementa a interface Authentication e faz a autenticação do usuário usando os dados desse objeto podendo retornar null quando não suportar o objeto passado, lançar uma exception em caso de erro ou retornando um objeto Authentication caso tudo dê certo.
- AuthenticationManager: É uma interface que define possui um método chamado authenticate, sua implementação padrão é ProviderManager, que delega para uma lista beans AuthenticationProvider a autenticação. Por padrão usa o bean DaoAuthenticationProvider
  - Caso uma exception AuthenticationException seja lançada ou null seja retornado a iteração continuará para o próximo AuthenticationProvider, de modo até todos serem percorridos e será considerada a última exception lançada

Autorização

A autorização em spring é baseada em authorities/roles, que nada mais é do que um identificador usado para identificar um tipo de usuário (ex: admin) e que são verificadas na hora de acessar um recurso, também existem roles, que são authorities com o prefixo role (ex: role_admin) antes. Essa diferenciação entre role/authorities não é muito bem explicada, mas ela existe.

Authorities são definidos pela interface GrantedAuthority que possui um método get e diversas implementações, sendo a mais comum SimpleGrantedAuthority.

UserDetails pode ter n authorities, que podem ser armazenadas em uma coluna e separadas por ; ou em uma tabela própria. E um objeto Authentication possui um método getAuthorities que precisa ser populado durante a construção do objeto.

CSRF

Por padrão CSRF vem habilitado para qualquer request POST, PUT, DELETE e PATCH, o token é esperado no header X-XSRF-TOKEN, sendo necessário desabilitar a proteção nas configs caso não faça sentido usá-lo (uma API stateless por exemplo). É possível desabilitar CSRF para apenas alguns endpoints por meio do método ignoringAntMatchers(<ant matchers>);

Interações de spring security

Method security

É possível usar anotações para fazer a autorização de roles direto no controller desde que os atributos da anotação @EnableGlobalMethodSecurity estejam com a flag true, as anotações suportadas são:

- @Secured e @RollesAllowed: Recebem uma lista de roles que podem chamar um endpoint no controller. A diferença entre as duas é que @Secured é específico do spring, enquanto @RolesAllowed é de javax.security.
- @PreAuthorize e @PostAuthorize: Além de receber os roles também podem receber expressões SpEL (uma linguagem que o spring usa para expressar condições de acesso mais específicas como por IP).

Injeção de objetos

É possível receber os seguintes objetos em uma classe anotada com @Controller ou @RestController

- O principal (username, client-id, etc) do usuário por meio da anotação @AuthenticationPrincipal. Se o usuário não estiver autenticado será retornado null
- O objeto CurrentSecurityContext que possui o objeto Authenticate retornado na autenticação do usuário por meio da anotação @CurrentSecurityContext. Esse objeto também é acessível de qualquer lugar pela classe SecurityContextHolder

- O objeto CSRFToken

Spring Authorization Server

É um projeto recente do spring, baseado em spring security e usado para implementar um servidor de autorização usando o padrão oauth2, ele tem como objetivo substituir uma parte de um projeto deprecado chamado spring security oauth, sendo que a outra parte desse projeto foi para o spring security.