Spring Security: Architecture Principles

Daniel Garnier-Moiroux

Spring I/O, 2024-05-30

Daniel Garnier-Moiroux

Software Engineer @ Broadcom

- Spring + Tanzu
- @Kehrlann@hachyderm.io
- @Kehrlann
- https://garnier.wf/
- github.com/Kehrlann/
- contact@garnier.wf

Spring Security

😬 🤯 🤕 😱 😵‍💫

I have a complex scenario. What could be wrong?

You need an understanding of the technologies you intend to use before you can successfully build applications with them. Security is complicated. Setting up a simple configuration [...] is reasonably straightforward.

However, **if you try to jump straight to a complicated [configuration], you are almost certain to be frustrated**. [...] So you need to take things one step at a time.

source: Spring Security reference docs, FAQ

Spring Security

😬 🤯 🤕 😱 😵‍💫

Spring Security

❤️ ❤️ ❤️ ❤️ ❤️

Contents

Contents

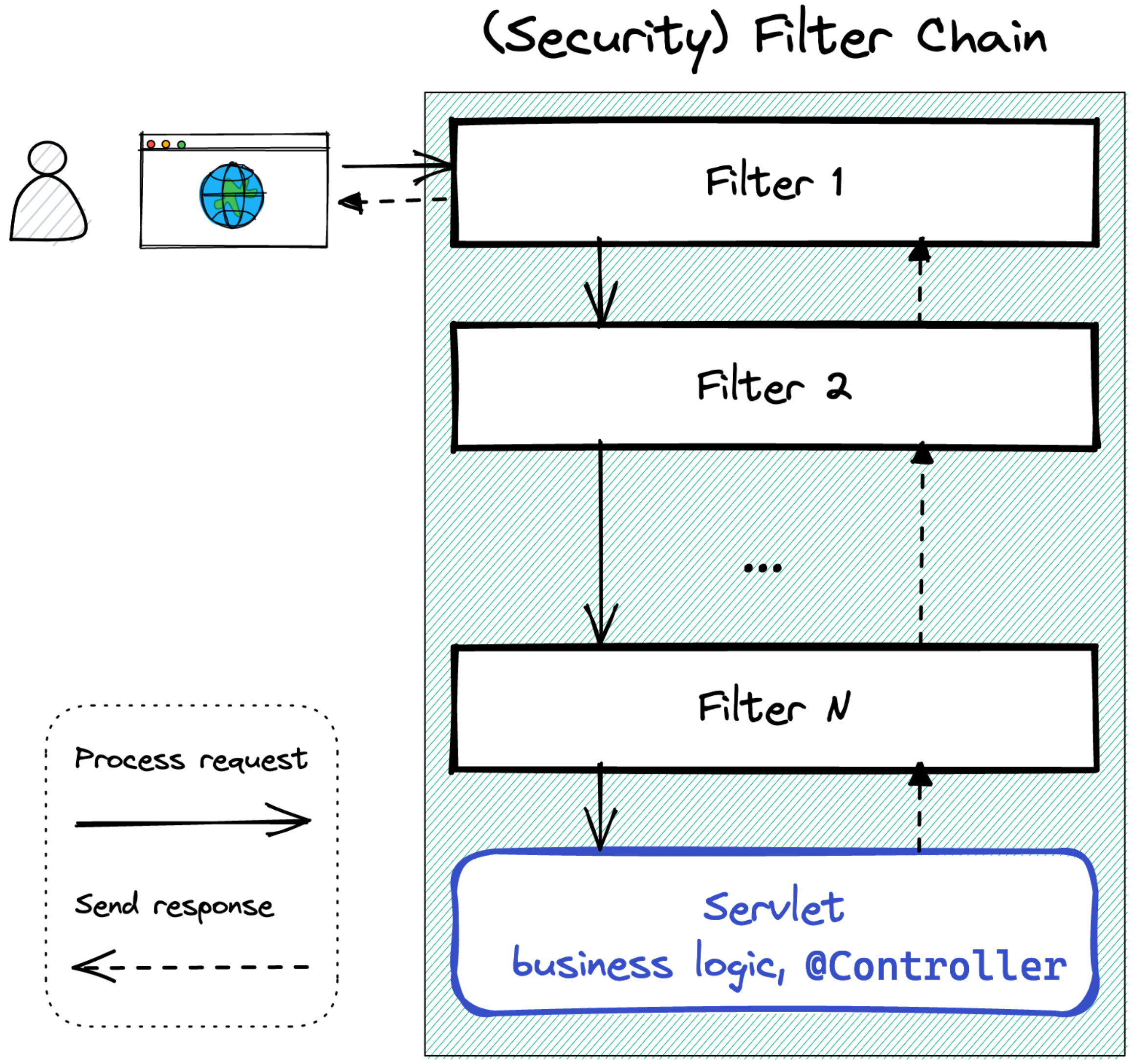
Demo

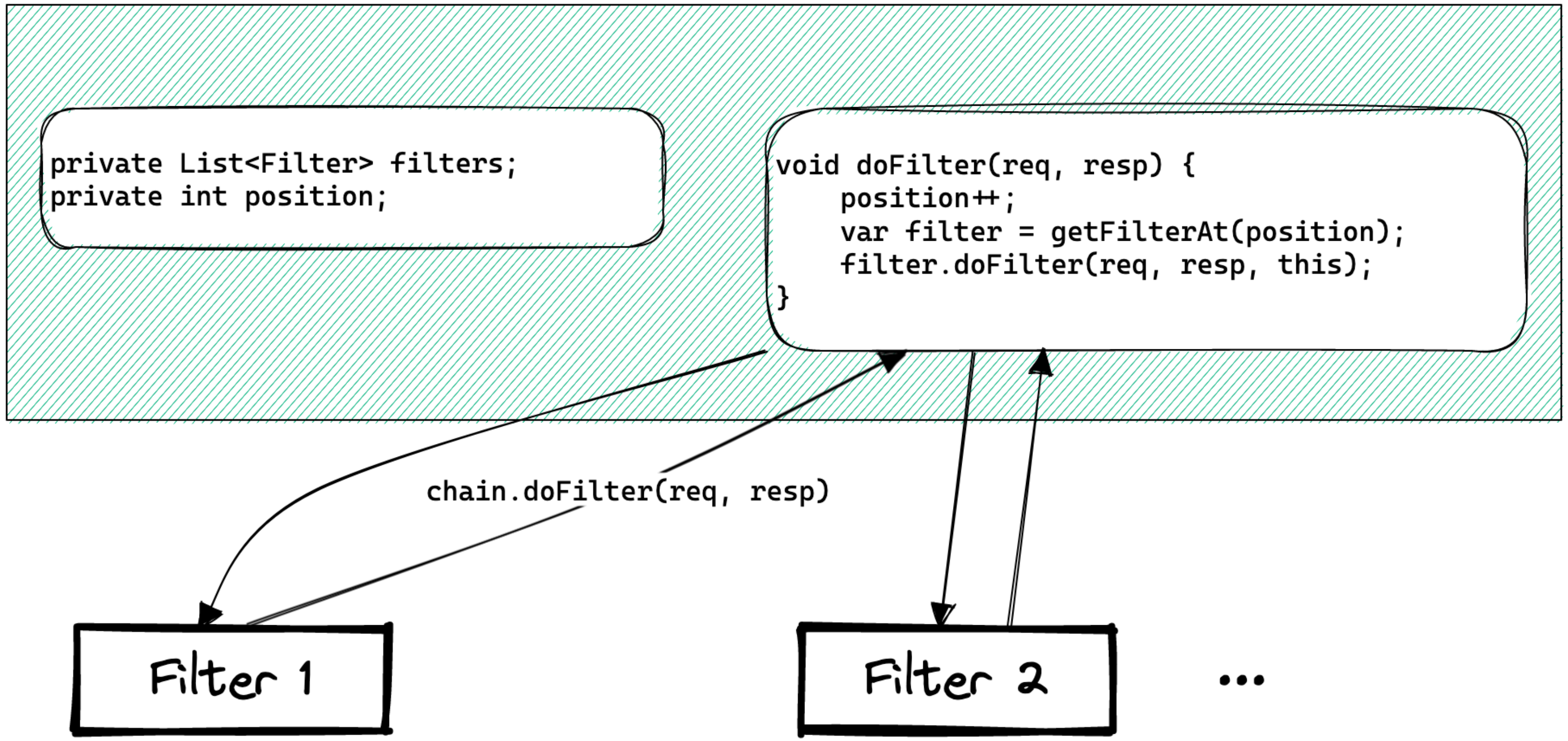
🍃🔐 A basic, secured app

Contents

Spring Security Filter

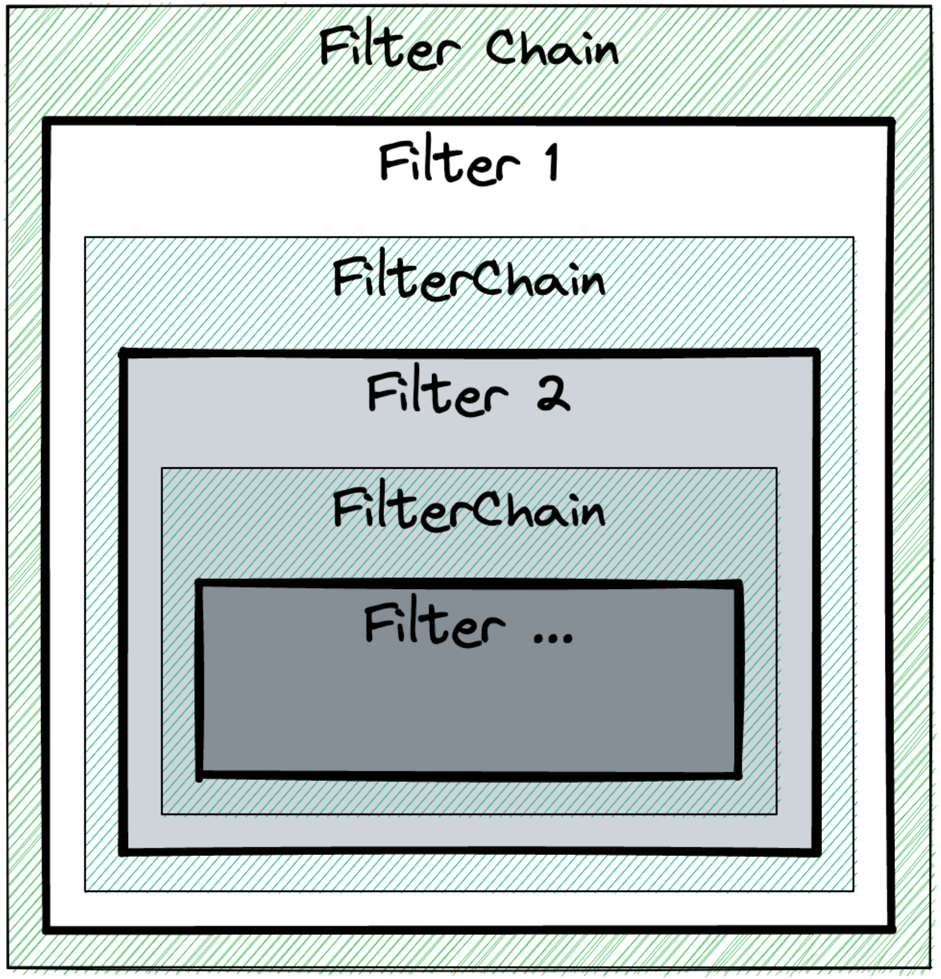
```java
  public void doFilter(
    HttpServletRequest request,
    HttpServletResponse response,
    FilterChain chain
    ) {
      // 1. Before the request proceeds further (e.g. authentication or reject req)
      //  ...

      // 2. Invoke the "rest" of the chain
      chain.doFilter(request, response);

      // 3. Once the request has been fully processed (e.g. cleanup)
      //  ...
  }
```

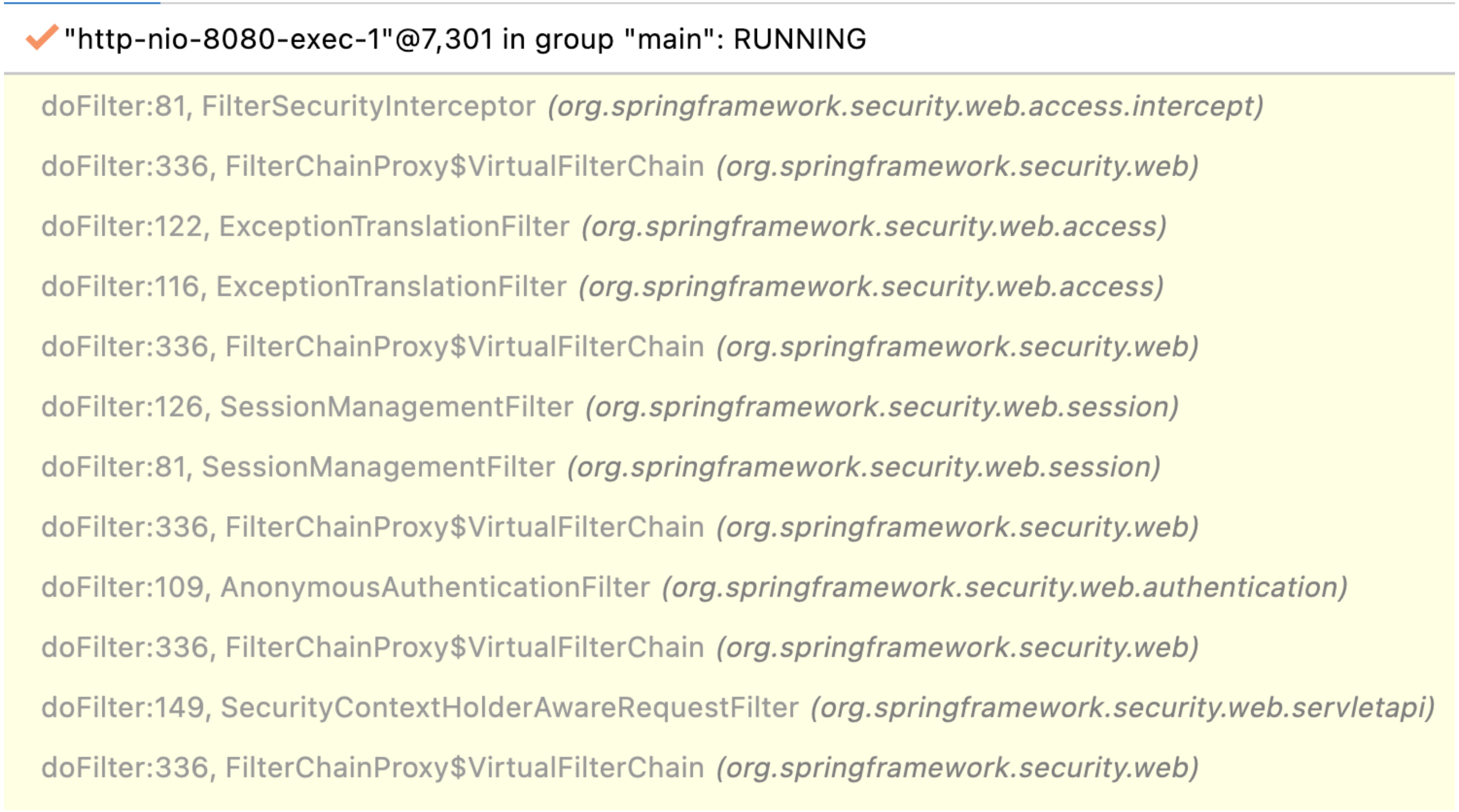

(Security) Filter Chain
Filter 1
Filter 2
...
Filter N
Servlet
business logic, @Controller
Process request
Send response



```
private List<Filter> filters;
private int position;
```

```
void doFilter(req, resp) {
    position++;
    var filter = getFilterAt(position);
    filter.doFilter(req, resp, this);
}
```


Filter Chain
Filter 1
FilterChain
Filter 2
FilterChain
Filter ...



```
"http-nio-8080-exec-1"@7,301 in group "main": RUNNING
doFilter:81, FilterSecurityInterceptor (org.springframework.security.web.access.intercept)
doFilter:336, FilterChainProxy$VirtualFilterChain (org.springframework.security.web)
doFilter:122, ExceptionTranslationFilter (org.springframework.security.web.access)
doFilter:116, ExceptionTranslationFilter (org.springframework.security.web.access)
doFilter:336, FilterChainProxy$VirtualFilterChain (org.springframework.security.web)
doFilter:126, SessionManagementFilter (org.springframework.security.web.session)
doFilter:81, SessionManagementFilter (org.springframework.security.web.session)
doFilter:336, FilterChainProxy$VirtualFilterChain (org.springframework.security.web)
doFilter:109, AnonymousAuthenticationFilter (org.springframework.security.web.authentication)
doFilter:336, FilterChainProxy$VirtualFilterChain (org.springframework.security.web)
doFilter:149, SecurityContextHolderAwareRequestFilter (org.springframework.security.web.servletapi)
doFilter:336, FilterChainProxy$VirtualFilterChain (org.springframework.security.web)
```

Demo

⛔ Our first filter

A detailed example

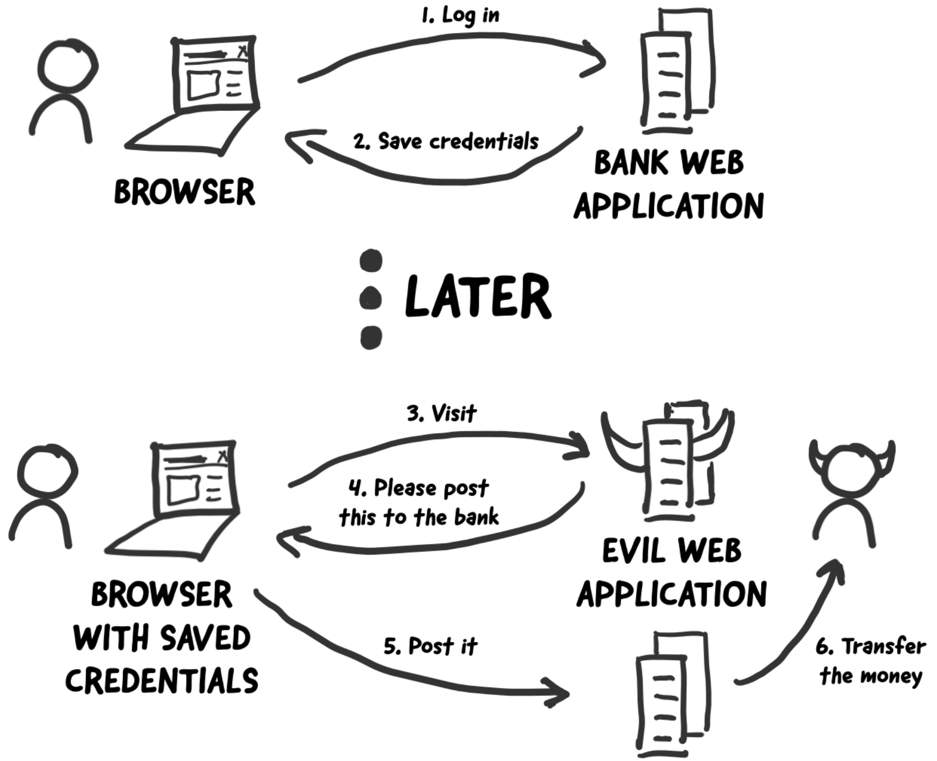
`CsrfFilter.java`

Cross

Site

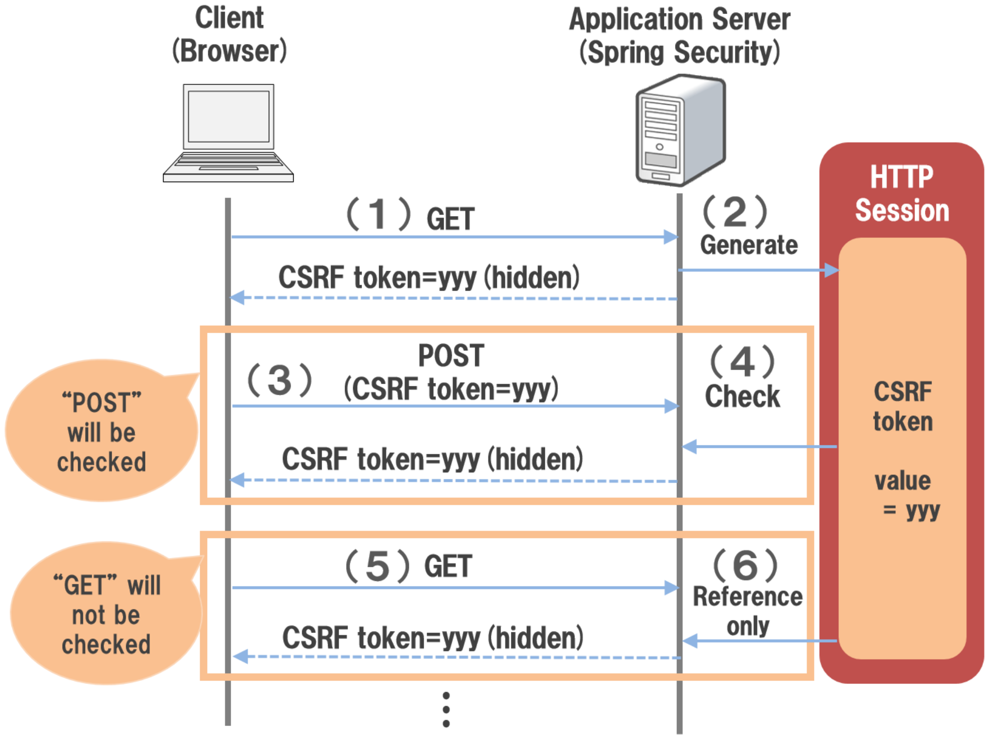
Request

Forgery



Protection

```
<form ... >
 <!-- visible inputs -->
  <input
      type="hidden"
      name="_csrf"
      value="yyy" />
</form>
```



A "real" example

`CsrfFilter.java`

Other filters?

Static, on startup: `DefaultSecurityFilterChain`

Dynamic, at runtime:

```yaml
logging.level:
  org.springframework.security: TRACE
```

Recap

1. Basic interface `Filter`, specifically `OncePerRequestFilter`
    1. Takes HttpServletRequest, HttpServletResponse
    2. Reads from request
        1. Sometimes writes to Response
        2. Sometimes does nothing!
    3. If request is "secure", calls `filterChain.doFilter( ... )`
2. Filters are registered `SecurityFilterChain`
    1. Order matters
    2. *Before* `AuthorizationFilter.class`

Contents

Authentication objects

Spring Security produces `Authentication` objects. They are used for:

- Authentication (`authn`): *who* is the user?
- Authorization (`authz`): is the user *allowed to perform* XYZ?

Vocabulary

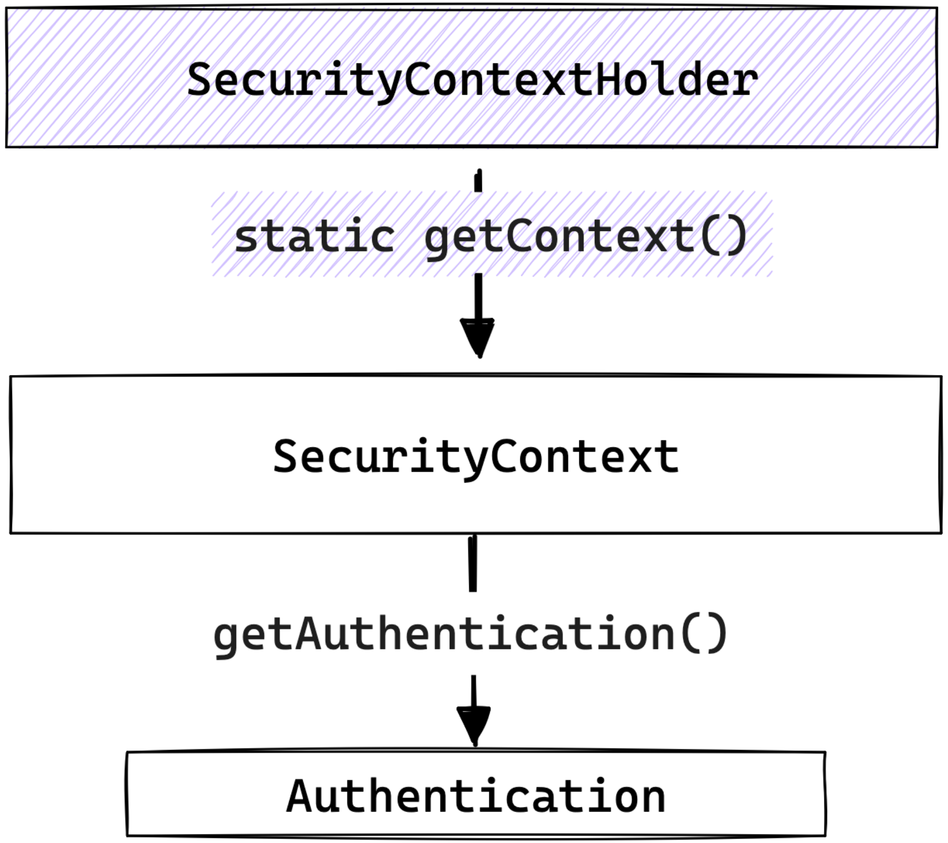
- **Authentication**: represents the user. Contains:
  - **Principal**: user "identity" (name, email…)
  - **GrantedAuthorities**: "permissions" (`roles` , …)

Vocabulary (cont')

- **Authentication** also contains:
  - **.isAuthenticated()**: almost always `true`
  - **details**: details about the *request*
  - (Credentials): "password", often `null`

SecurityContext

- Thread-local
- Not propagated to child threads
- Cleared after requests is processed



What’s the most common `Authentication` implementation?

Good practice

DO NOT

Use `UsernamePasswordAuthenticationToken` everywhere

INSTEAD

Create your own `Authentication` subclasses

Remember our filter?

```java
public void doFilter(
  HttpServletRequest request,
  HttpServletResponse response,
  FilterChain chain
  ) {
    // 1. Before the request proceeds further (e.g. authentication or reject req)
    //  ...

    // 2. Invoke the "rest" of the chain
    chain.doFilter(request, response);

    // 3. Once the request has been fully processed (e.g. cleanup)
    //  ...
}
```

More like this

```java
public void doFilter(
  HttpServletRequest request,
  HttpServletResponse response,
  FilterChain chain
  ) {
    // 1. Decide whether the filter should be applied

    // 2. Apply filter: authenticate or reject request

    // 3. Invoke the "rest" of the chain
    chain.doFilter(request, response);

    // 4. No cleanup
}
```

Demo

🤖 Robot wants Auth

Recap

1. Some filters produce an `Authentication`
    1. Read the request ("convert" to domain object)
    2. Authenticate (are the credentials valid?)
    3. Save the `Authentication` in the `SecurityContext`
    4. Or reject the request when creds invalid
2. There's more than just `UsernamePasswordAuthenticationToken`!

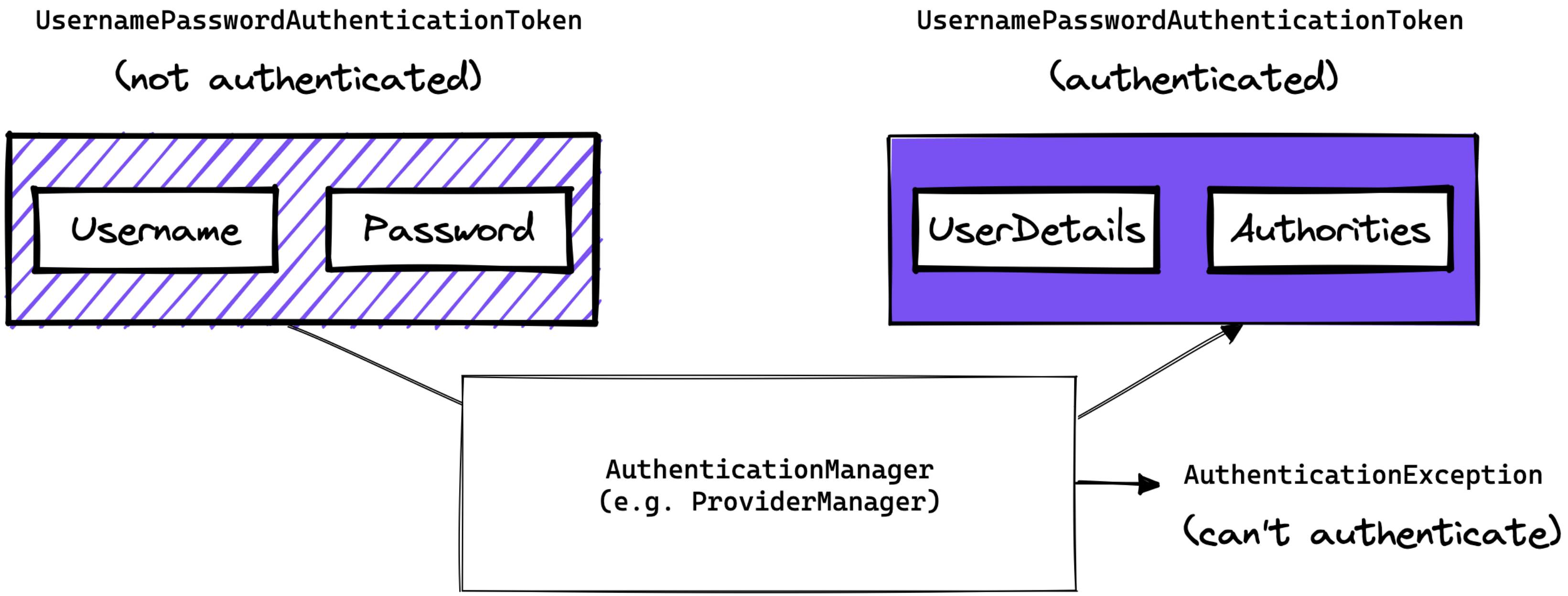
Contents

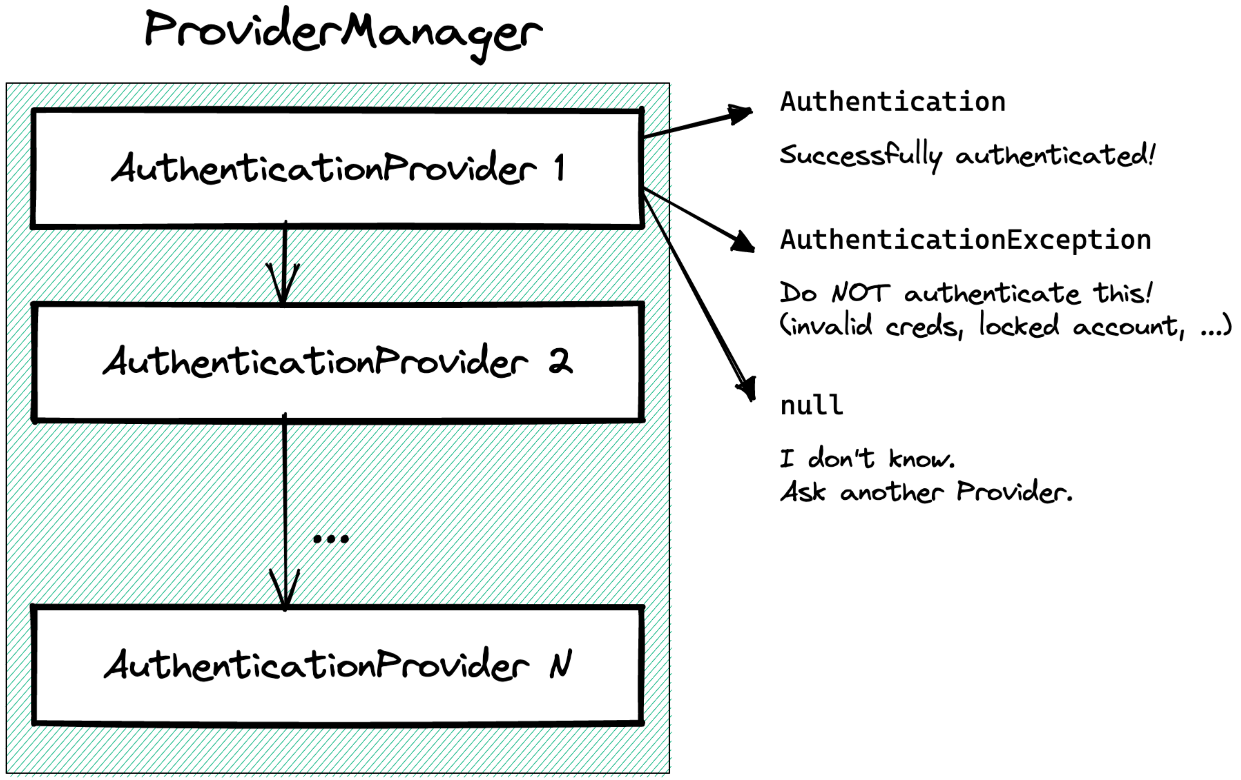
Authentication

Muahaha I lied 😈

`Authentication` objects are both:

- The result of a *authentication action*
- An *authentication request*


UsernamePasswordAuthenticationToken
(not authenticated)
Username
Password
UsernamePasswordAuthenticationToken
(authenticated)
UserDetails
Authorities
AuthenticationManager
(e.g. ProviderManager)
AuthenticationException
(can't authenticate)


ProviderManager
AuthenticationProvider 1
AuthenticationProvider 2
...
AuthenticationProvider N
Authentication
Successfully authenticated!
AuthenticationException
Do NOT authenticate this!
(invalid creds, locked account, ...)
null
I don't know.
Ask another Provider.

Demo

🧑🏻 Daniel's edge-case

Recap

1. `Authentication` is both an auth request and a successful auth result
2. `AuthenticationProvider` validate credentials
    1. Operates only within the "auth" domain (no HTTP, HTML, …)
3. `AuthenticationProvider` leverages Spring Security infrastructure

Contents

Wrapping up

1. `Filter` for security decisions on HTTP requests
2. `Authentication` is the domain language of Spring Security
3. `AuthenticationProvider` to validate credentials
4. `Filter` + `AuthenticationProvider` for custom login

Repo:

- https://github.com/Kehrlann/spring-security-the-good-parts

Reach out:

- @Kehrlann@hachyderm.io
- @Kehrlann
- https://garnier.wf/
- contact@garnier.wf

😄 Thank you!