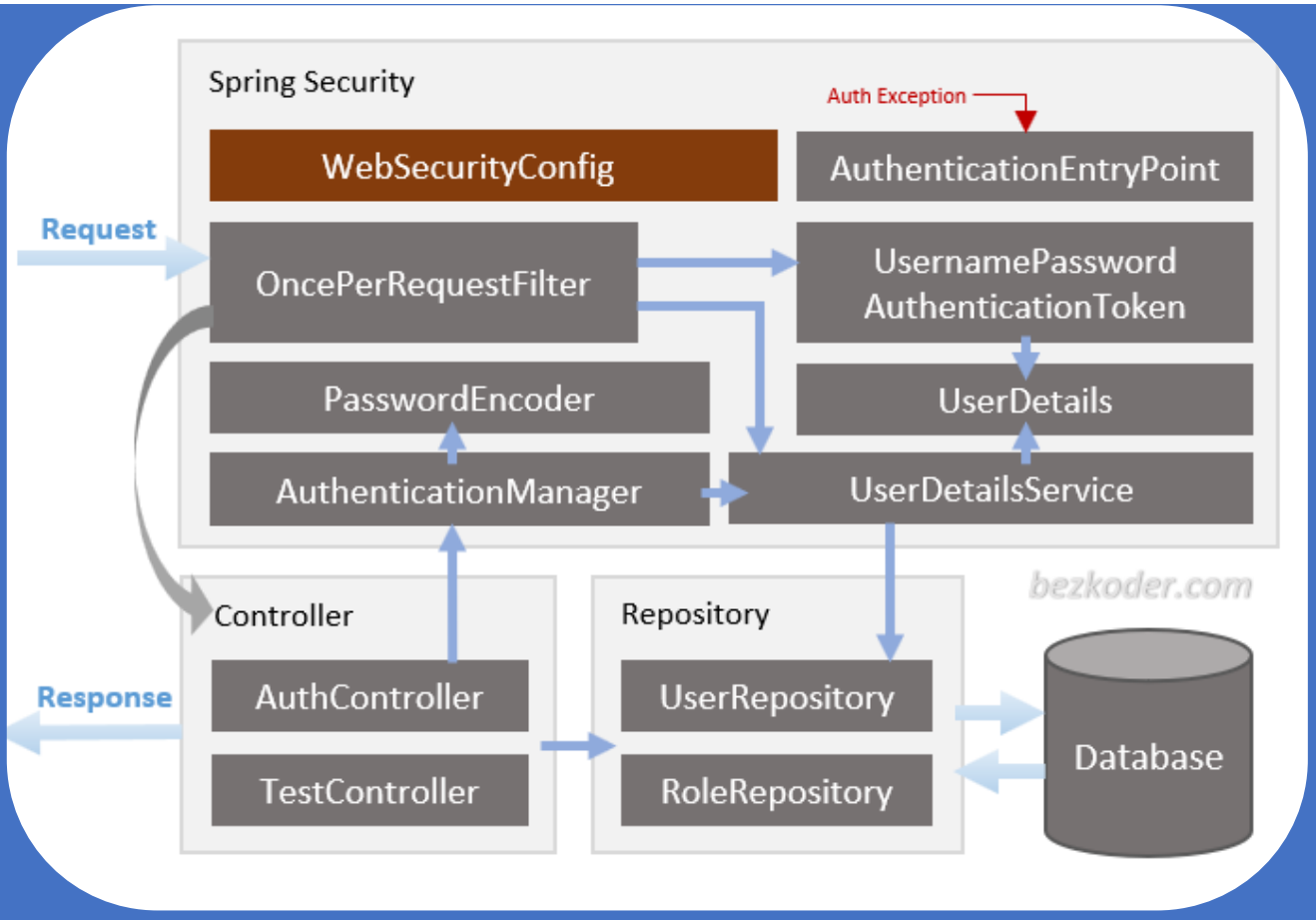
SPRING SECURITY



Aron Haubner

WISS [Firmenadresse]

In Wörtern vom Ersteller

Now I will explain it briefly.

Spring Security

– `WebSecurityConfig` is the crux of our security implementation. It configures cors, csrf, session management, rules for protected resources. We can also extend and customize the default configuration that contains the elements below.
(`WebSecurityConfigurerAdapter` is deprecated from Spring 2.7.0, you can check the source code for update. More details at:
WebSecurityConfigurerAdapter Deprecated in Spring Boot)

– `UserDetailsService` interface has a method to load User by *username* and returns a `UserDetails` object that Spring Security can use for authentication and validation.

– `UserDetails` contains necessary information (such as: username, password, authorities) to build an Authentication object.

– `UsernamePasswordAuthenticationToken` gets {username, password} from login Request, `AuthenticationManager` will use it to authenticate a login account.

– `AuthenticationManager` has a `DaoAuthenticationProvider` (with help of `UserDetailsService` & `PasswordEncoder`) to validate `UsernamePasswordAuthenticationToken` object. If successful, `AuthenticationManager` returns a fully populated Authentication object (including granted authorities).

– `OncePerRequestFilter` makes a single execution for each request to our API. It provides a `doFilterInternal()` method that we will implement parsing & validating JWT, loading User details (using `UserDetailsService`), checking Authorizaion (using `UsernamePasswordAuthenticationToken`).

– `AuthenticationEntryPoint` will catch authentication error.

Repository contains `UserRepository` & `RoleRepository` to work with Database, will be imported into **Controller**.

Controller receives and handles request after it was filtered by `OncePerRequestFilter`.

– `AuthController` handles signup/login requests

– `TestController` has accessing protected resource methods with role based validations.

Understand the architecture deeply and grasp the overview more easier:
Spring Boot Architecture for JWT with Spring Security

Technology

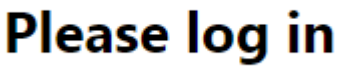
- Java 17 / 11 / 8
- Spring Boot 3 / 2 (with Spring Security, Spring Web, Spring Data JPA)
- jjwt-api 0.11.5
- PostgreSQL/MySQL
- Maven

In meinen Wörtern

Spring Security

`WebSecurityConfig` ist sehr wichtig, weil es fast alles konfiguriert und manage.

Schritt #1 Login



Schritt #2 User Details Objekt

> `UserDetailsService` erstellt ein User Objekt von jedem User, welches dann „User Details“ heisst

Schritt #3 Authentikation

> Bei einem Login request holt `UsernamePasswordAuthenticationToken` die eingegebenen Daten und schickt sie an den `AuthenticationManager`

Schritt #4 Manager

> Nvm ich weis schon wie es geht….