Spring Security

What is Spring Security?

- Spring Security is a framework that enables a programmer to impose security restrictions to Spring-framework–based Web applications through JEE components

Spring Security

Spring Security operates in two major areas

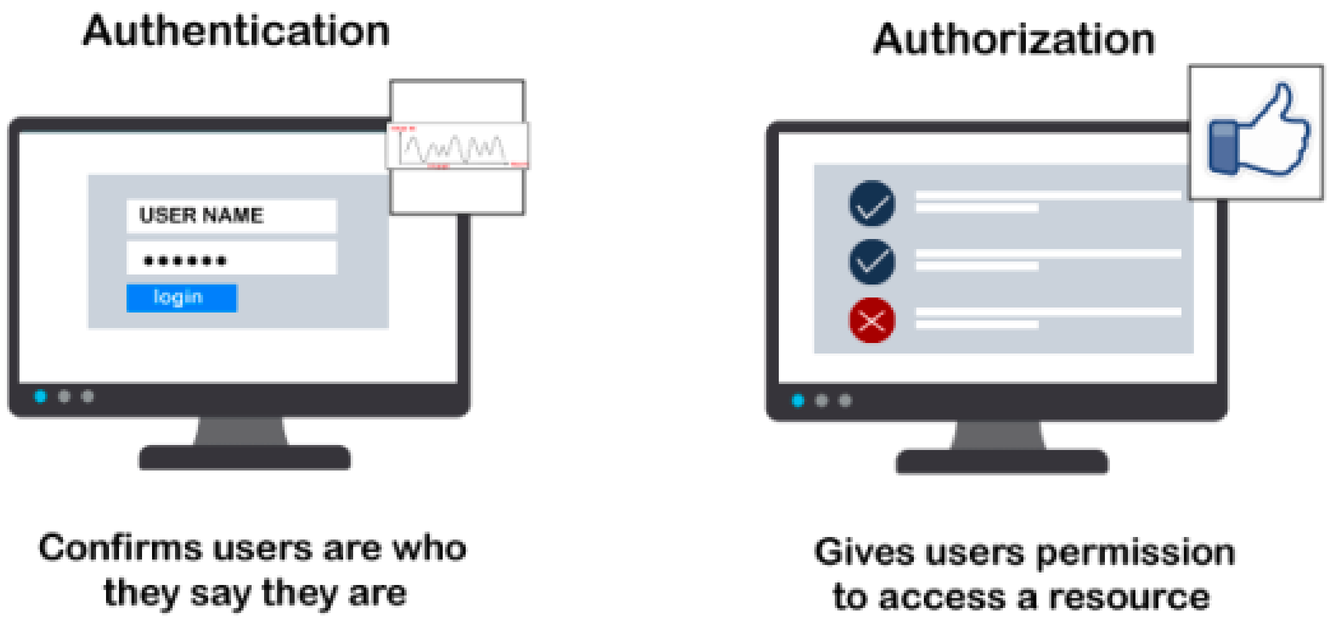
- Authentication
- Authorization

Authentication

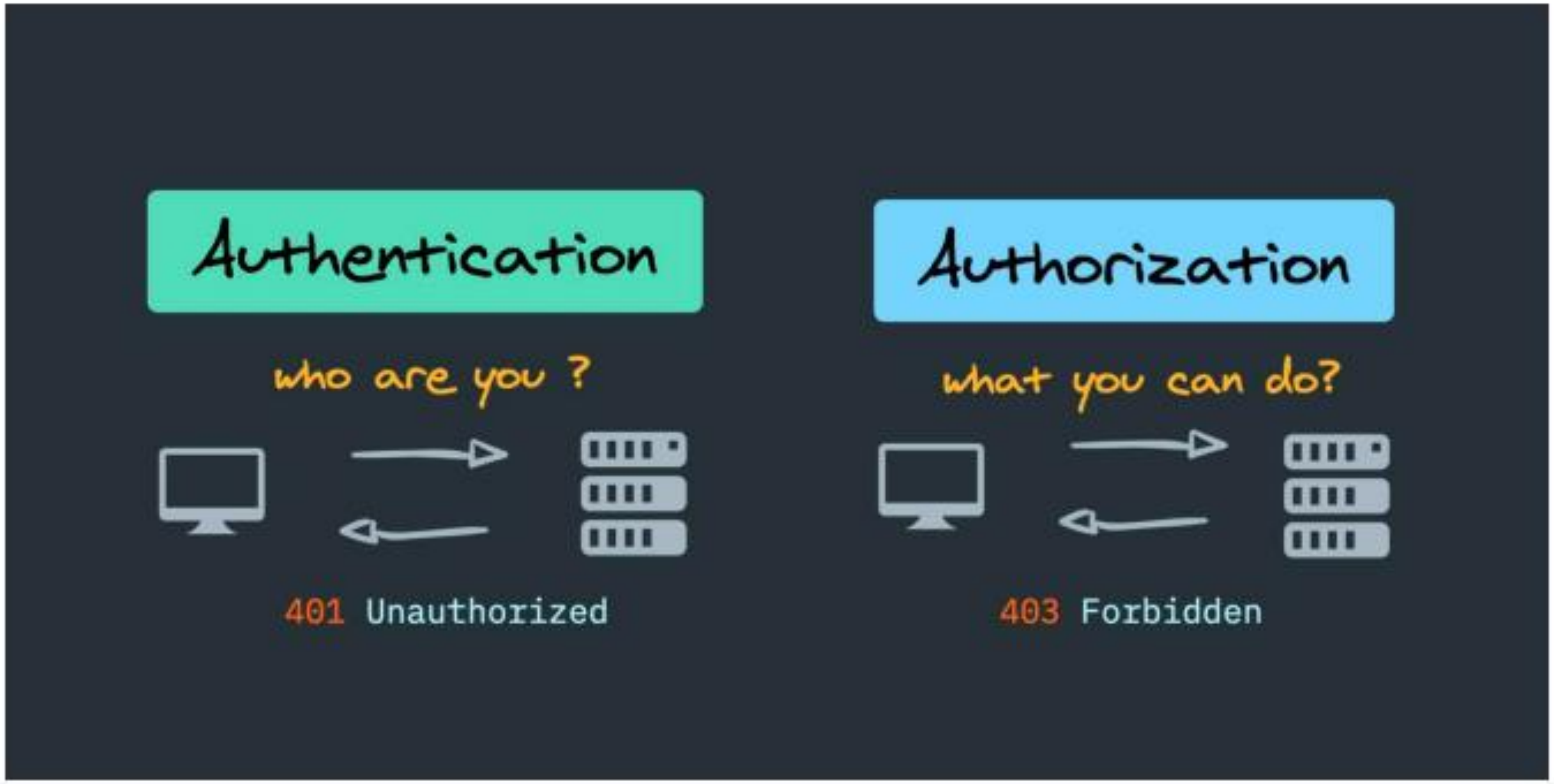
- Authentication is the process of verifying who a user is

Authorization

- Authorization is the process of verifying what they have access to


Authentication
USER NAME
••••••
login
Confirms users are who they say they are
Authorization
Gives users permission to access a resource

Error Codes


Authentication
who are you ?
401 Unauthorized
Authorization
what you can do?
403 Forbidden

Spring Security components

- **Filters**
- **Authentication Manager**
- **Authentication Providers**
- **UserDetails and UserDetailsService**
- **Security Context and Security Context Holder**
- **Authentication Token**
- **Access Decision Manager**
- **Granted Authority**
- **Session Management**

Filters

- **SecurityFilterChain**: A filter chain that intercepts requests and applies security measures. It is configured via the HttpSecurity object in modern Spring Security setups

Authentication Manager

- The central interface for managing authentication. It processes authentication requests and returns an Authentication object if the authentication is successful or throws an exception if it fails.

Authentication Providers

- Responsible for performing a specific type of authentication (e.g., username/password, token-based).

- **Common Implementations**:
  - **DaoAuthenticationProvider**: Authenticates based on user details stored in a database.
  - **JwtAuthenticationProvider**: Used for JWT (JSON Web Token) based authentication.
  - **LdapAuthenticationProvider**: For LDAP-based authentication.

UserDetails and UserDetailsService

- **UserDetails**: An interface that represents a user's information, including username, password, and granted authorities (roles).
- **UserDetailsService**: A service interface for loading user-specific data. The loadUserByUsername method is used to retrieve a UserDetails object.

Security Context and Security Context Holder

- **SecurityContext**: Holds the security information of the current thread of execution, including the authenticated user's details.
- **SecurityContextHolder**: A helper class that provides access to the SecurityContext. It is the primary interface to interact with the security context.

Authentication Token

- **Authentication**: The principal interface representing an authentication token. It contains the principal (usually the user), credentials (e.g., password), and granted authorities.

Access Decision Manager

- **AccessDecisionManager**: Makes final authorization decisions based on the security policy, the user's granted authorities, and the secured object (e.g., a method or URL).

Granted Authority

- **GrantedAuthority**: Represents an authority granted to the user, typically in the form of roles like ROLE_USER or ROLE_ADMIN.

Enabling spring security on the spring boot application

- By adding spring security dependency in pom.xml

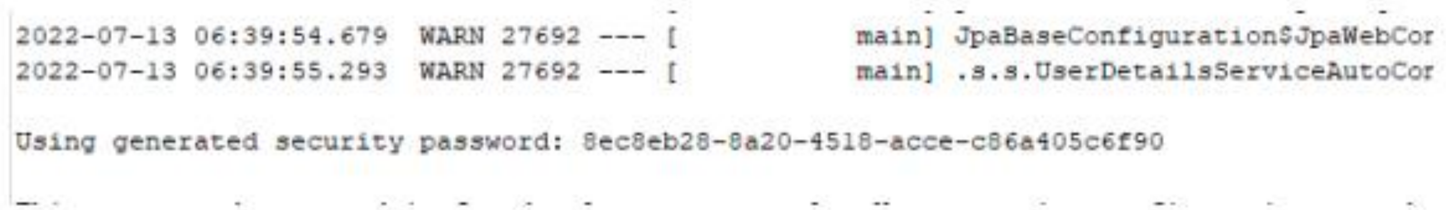
  - `<dependency>`
  - `<groupId>org.springframework.boot</groupId>`
  - `<artifactId>spring-boot-starter-security</artifactId>`
  - `</dependency>`

Rest Controller

- `@RestController`
- `public class MyController {`
- `@GetMapping("/")`
- ` public String home() {`
- ` return "<h1> Home works";`
- ` }`
- `}`

Start the App

- While starting the app u will notice a password is generated in default

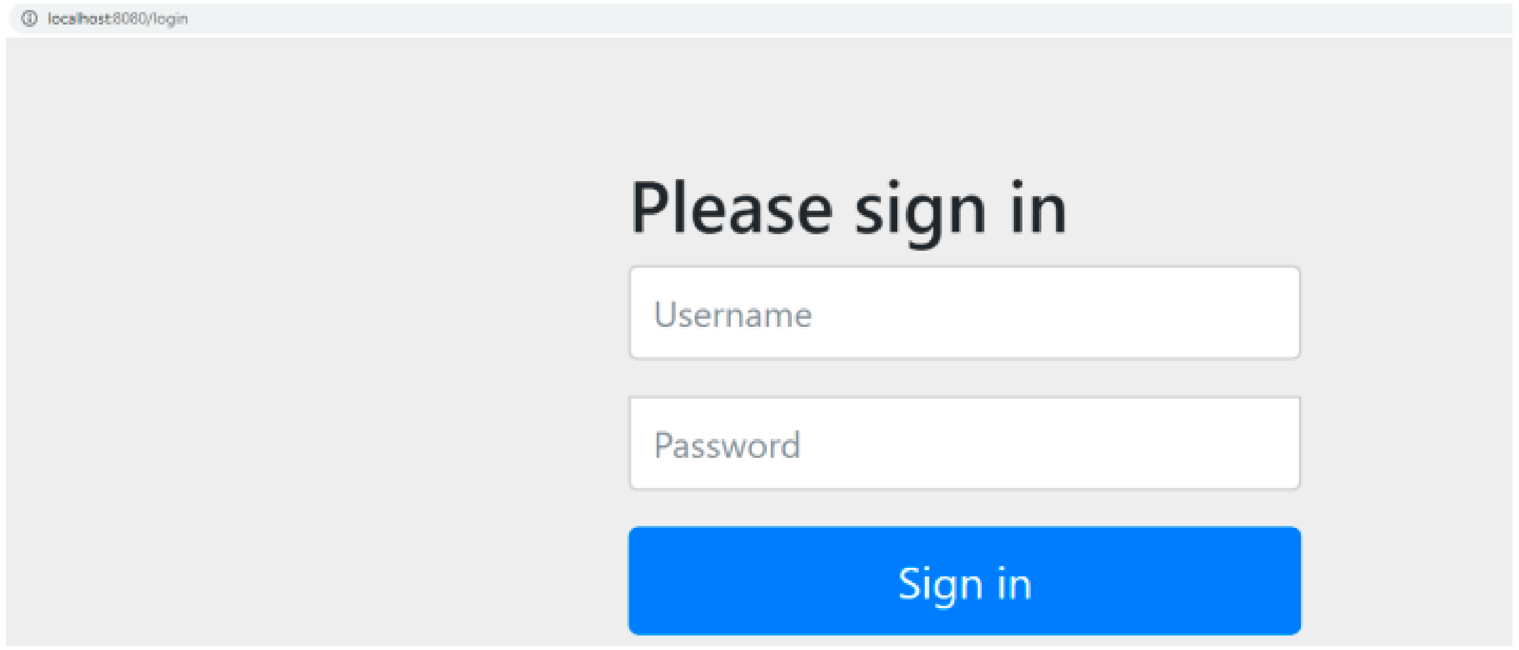


```
2022-07-13 06:39:54.679  WARN 27692 --- [           main] JpaBaseConfiguration$JpaWebCor
2022-07-13 06:39:55.293  WARN 27692 --- [           main] .s.s.UserDetailsServiceAutoCor

Using generated security password: 8ec8eb28-8a20-4518-acce-c86a405c6f90
```

Login form

- Just access the URL in browser http://localhost:8080, it will redirect to login form http://localhost:8080/login



Default credentials

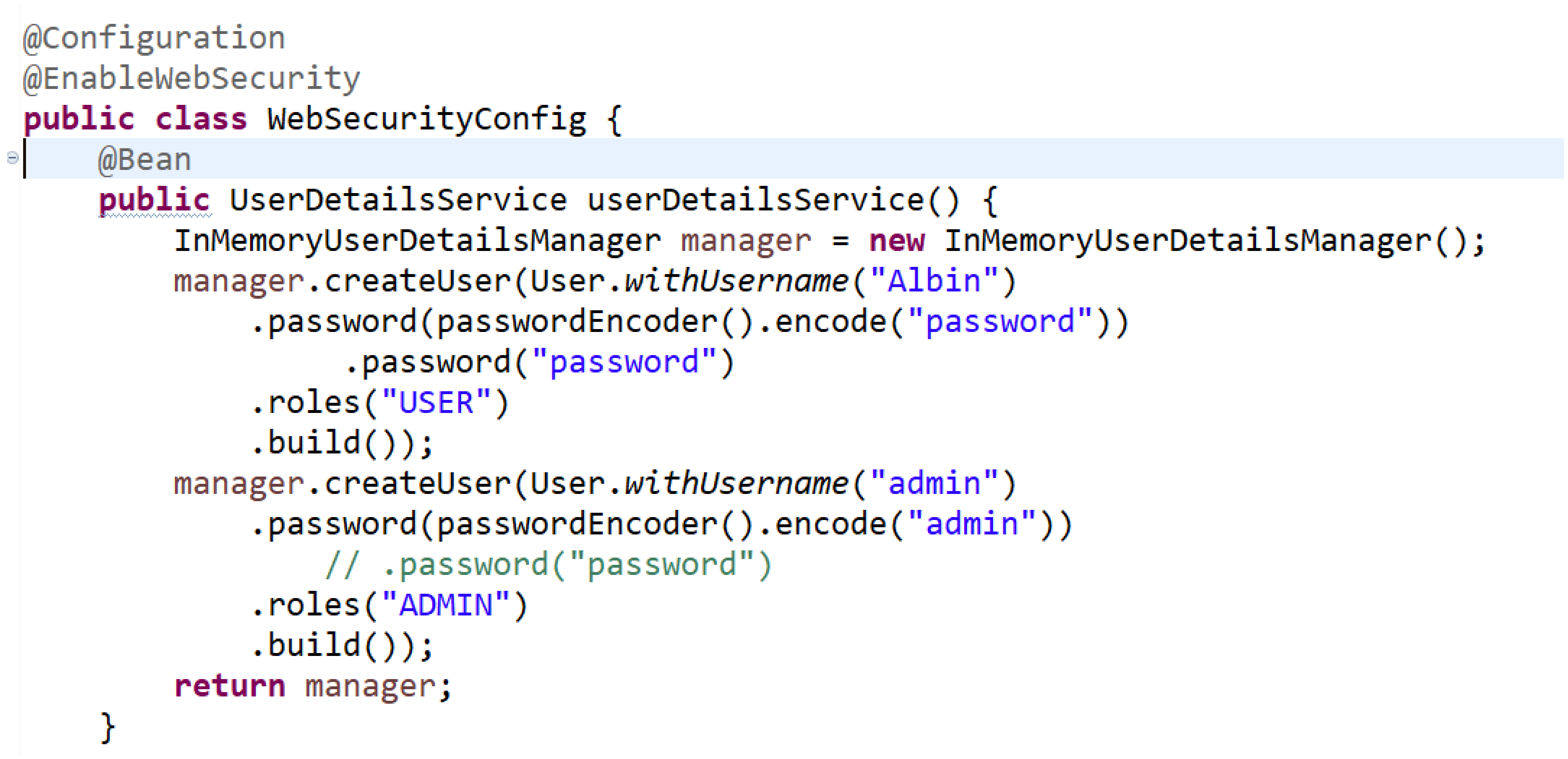
- Default username : user
- Password : provide the auto generated password

Override default credentials

- To override default credentials

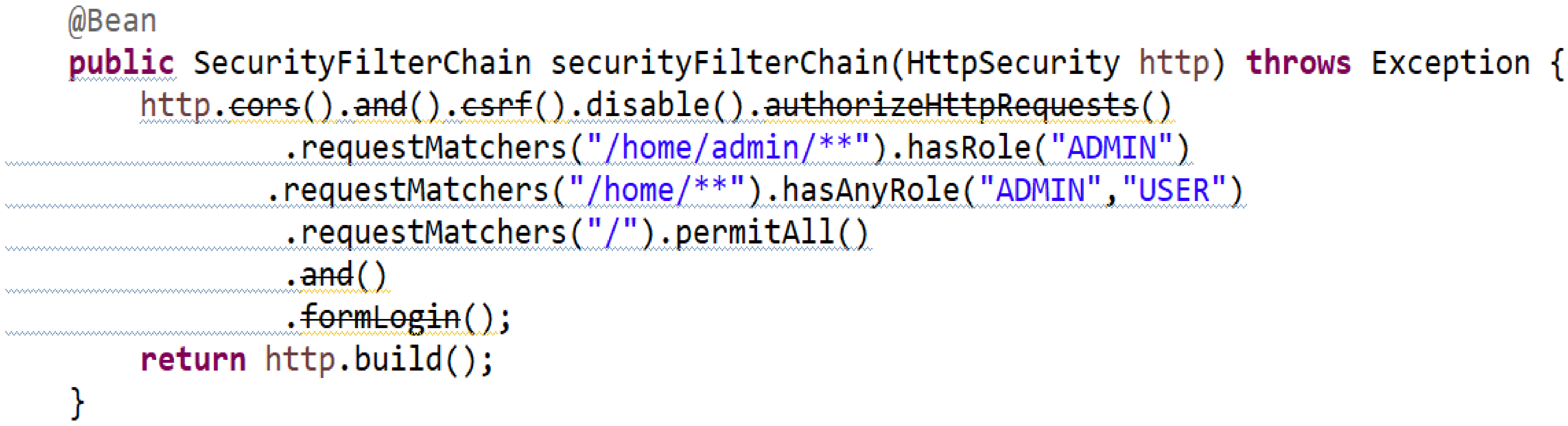
- application.properties file
- `spring.security.user.name=albin`
- `spring.security.user.password=xyz123456`

Configuring Security – In Memory

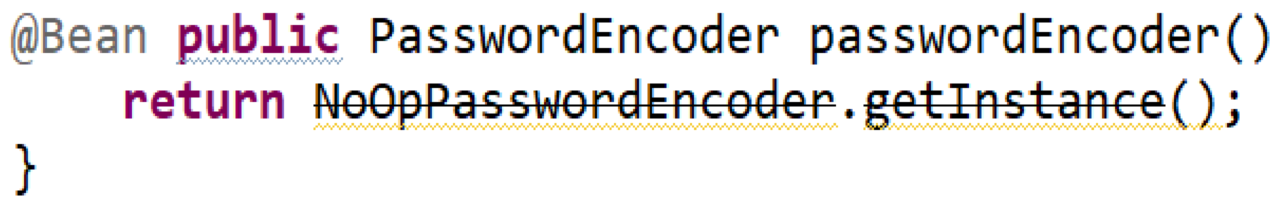


```java
@Configuration
@EnableWebSecurity
public class WebSecurityConfig {
    @Bean
    public UserDetailsService userDetailsService() {
        InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();
        manager.createUser(User.withUsername("Albin")
            .password(passwordEncoder().encode("password"))
                 .password("password")
            .roles("USER")
            .build());
        manager.createUser(User.withUsername("admin")
            .password(passwordEncoder().encode("admin"))
                // .password("password")
            .roles("ADMIN")
            .build());
        return manager;
    }
```

Configuring Security

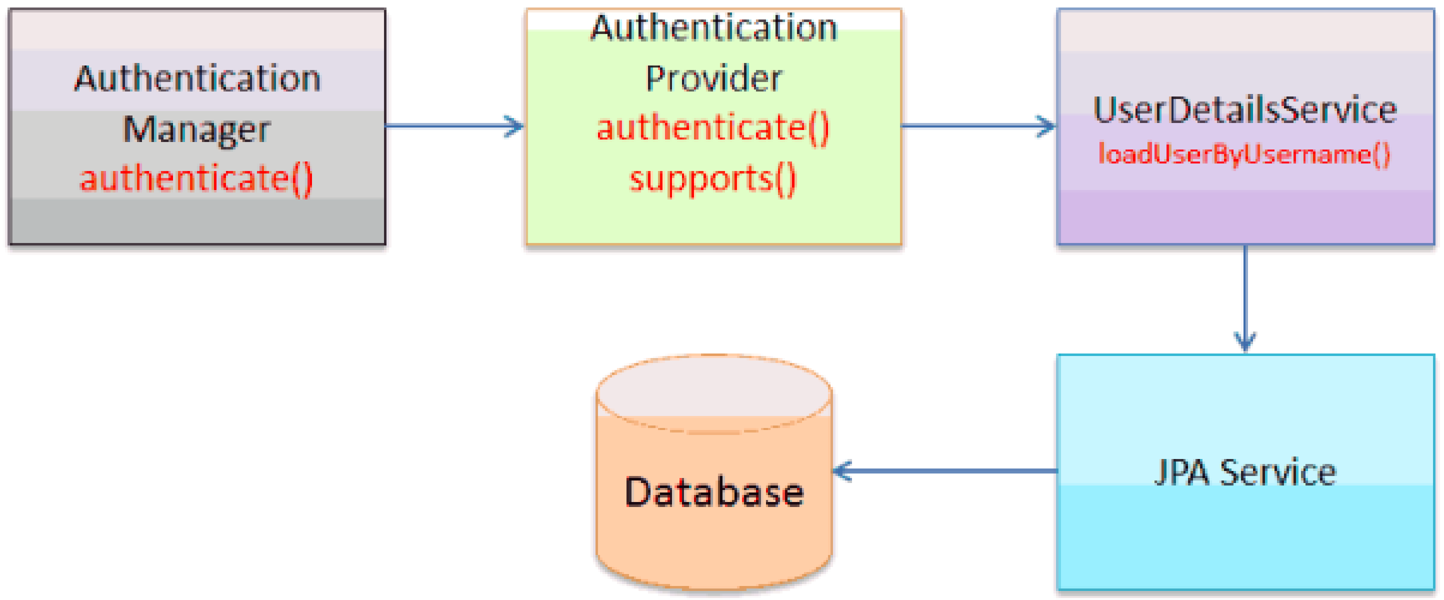


```
@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    http.~~cors~~().~~and~~().~~csrf~~().disable().~~authorizeHttpRequests~~()
            .requestMatchers("/home/admin/**").hasRole("ADMIN")
           .requestMatchers("/home/**").hasAnyRole("ADMIN","USER")
            .requestMatchers("/").permitAll()
            .~~and~~()
            .~~formLogin~~();
    return http.build();
}
```



```
@Bean public PasswordEncoder passwordEncoder() {
    return ~~NoOpPasswordEncoder~~.~~getInstance~~();
}
```

Spring security with JPA Authentication



THANK YOU!