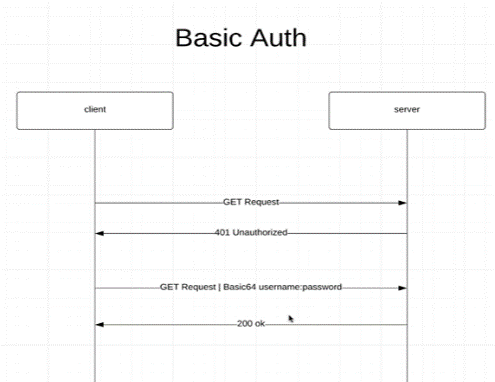
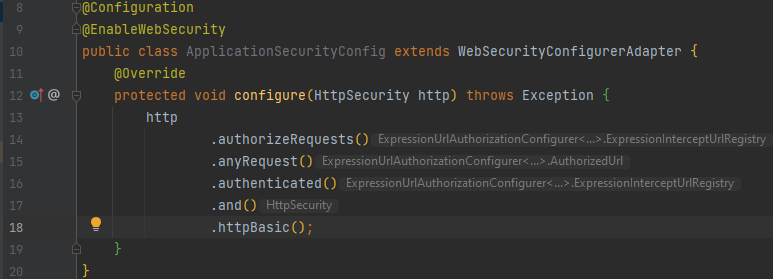
Spring Security

## Form-Based Auth



## Basic Auth

* Username, password must be specified in Request header
* in every single request!
* encoded as Base64
* no logout

Here we will configure all security aspects

1. new **package** security
2. new **class** ApplicationSecurityConfig
3. @Configuration
4. @EnableWebSecurity
5. **extends** WebSecurityConfigurerAdapter
6. **override method** configure(http:HttpSecurity) 🡪 authorize requests, any requests must be authenticated and the method used is basic64

Test API with Postman

* normal GETrequest gives back 401-Unauthorized
* we have to send user, password in header
  + click: “Authorization 🡪 Type: Basic 🡪 enter username, pass

Whitelist specific URLs – Antmatchers

* possible for html, css, js files
* .antMatchers(..patterns:”/”…)
* .permitALL()

Create in memory user

1. in security.ApplicationSecurityConfig override method UserDetailsService
2. @Bean
   1. will be instantiated when we start the app

UserDetailsService (interface) 🡪 how we retrieve users from db

Add BCrypt

1. new class PasswordConfig in package security
2. @Configuration
3. method: PasswordEncoder(interface)

min 56

Roles and Permissions

* a role (e.g admin) has a set of permissions (e.g student:read, student:write)

Admin User

* new user in ApplicationSecurityConfig-UserDetailsService
* build()

User roles and permissions with enums

Role based Authentication

* restrict APIs with role-based-authentification
* define second antmatchers in httpSecurity (first: .permitAll, second: .hasRole)

Permission based Authentication

* e.g. admin(read and write) and admintrainee (read only)
* has Authorities, hasRoles

Adding Authorities to users