Spring Security

It is a powerful and flexible framework designed to handle the authentication, authorization, and protection of Spring-based applications

1. Form Based
2. Basic Auth
3. In Memory
4. Database
5. Token Based Auth (JWT)

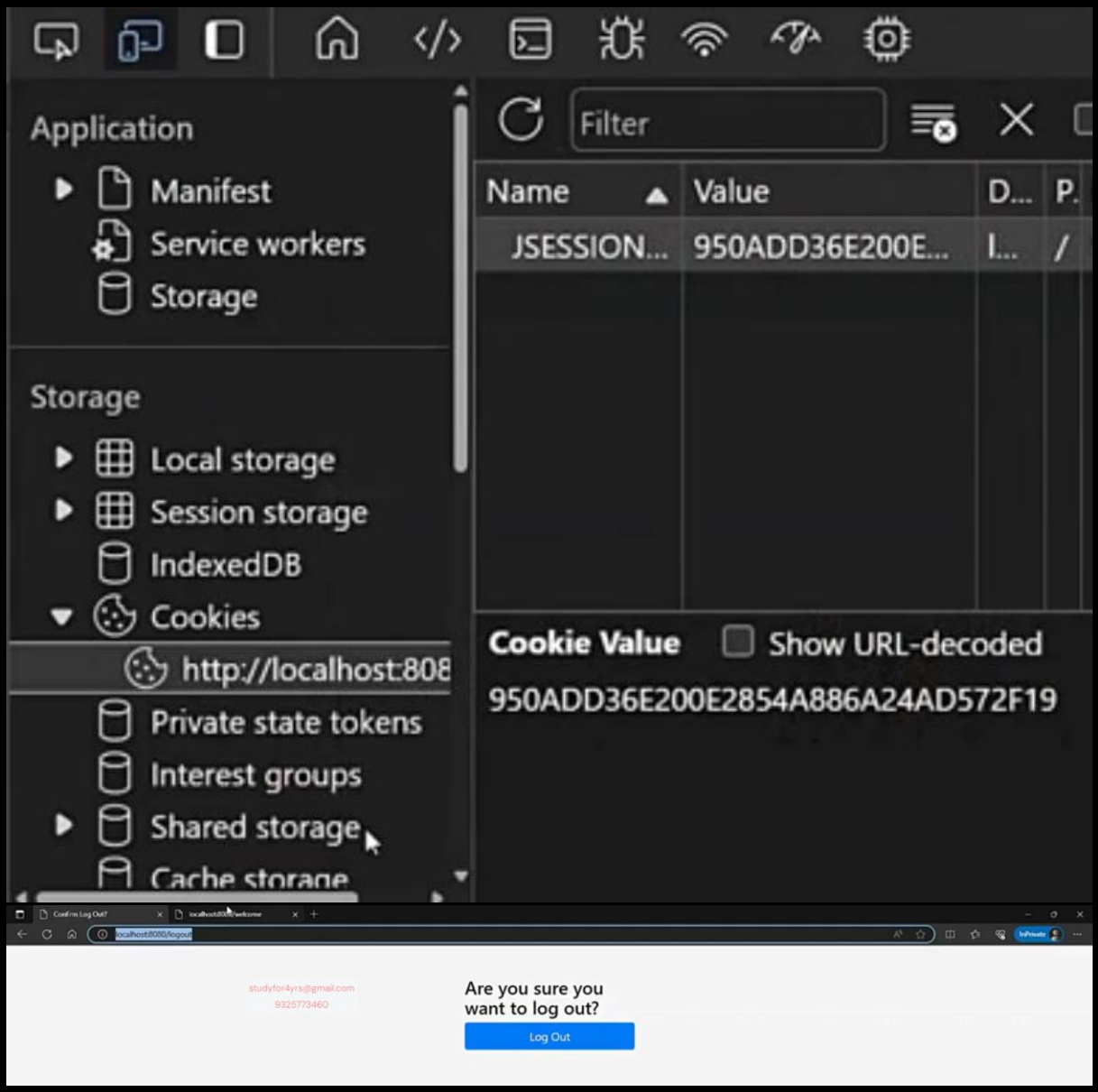
Role Based + Authorization

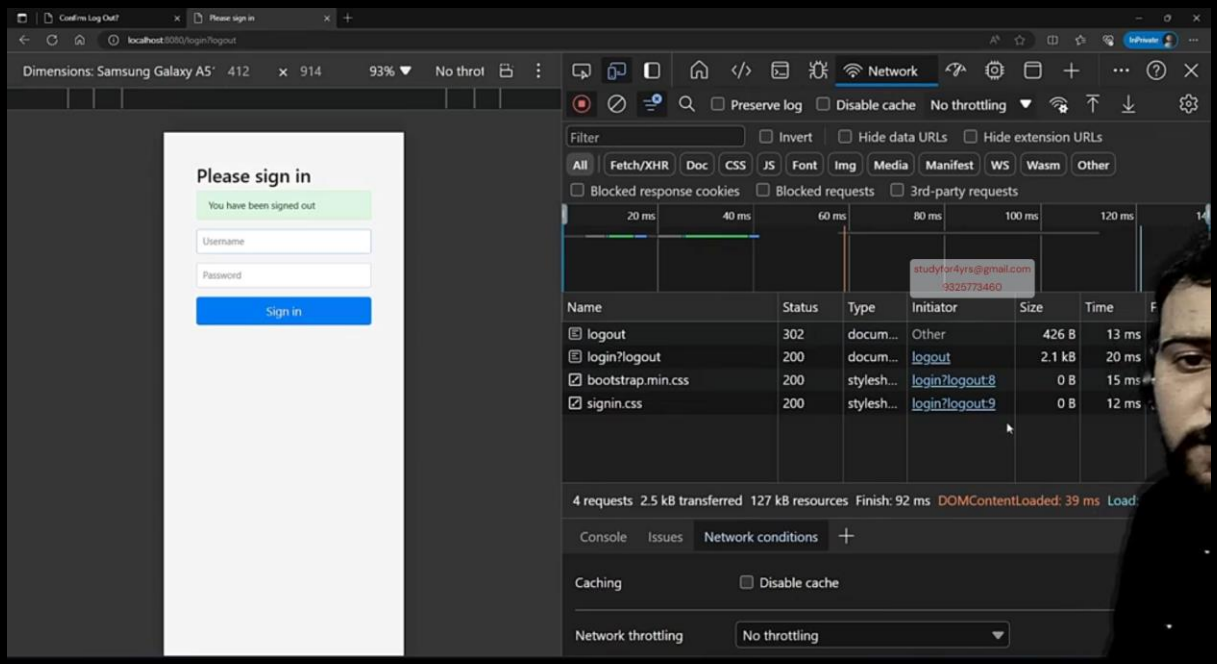
Spring Security SECURITY

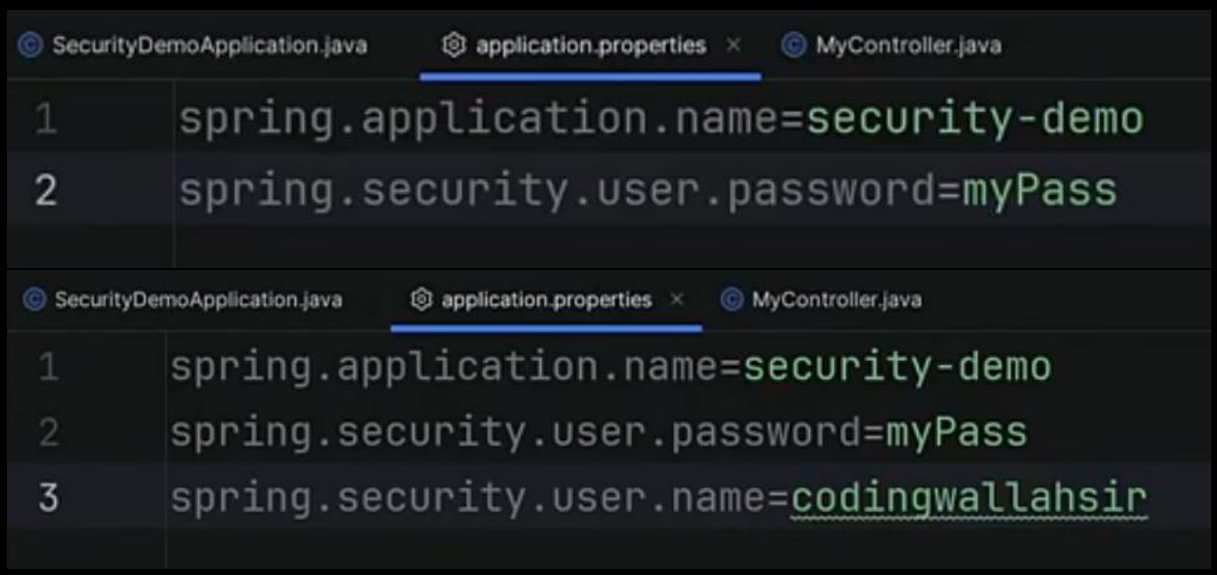
Highly customizable authentication and access-control framework for Spring applications.

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

```
Using generated security password: ad10d8e7-6107-4ff2-821d-74b205aae9e0
```

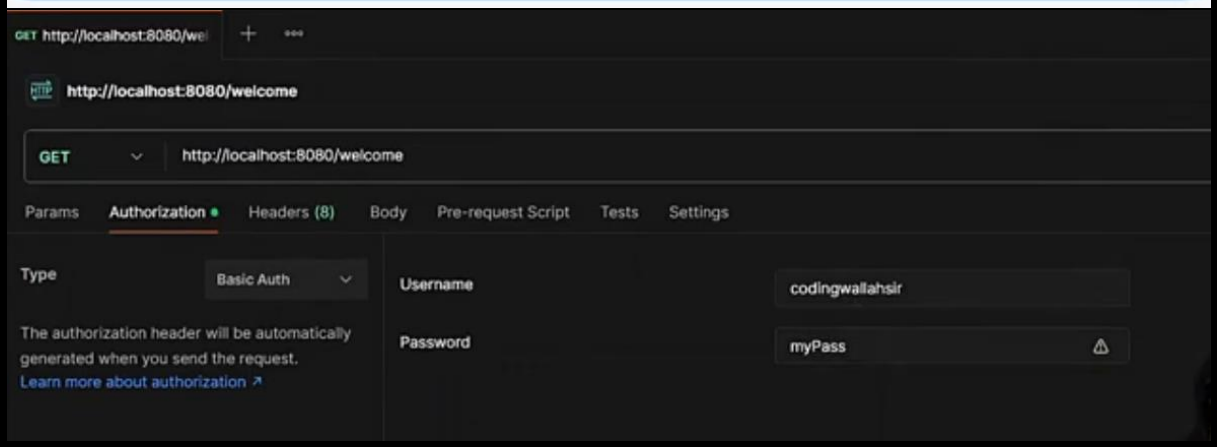

Application
Manifest
Service workers
Storage
Storage
Local storage
Session storage
IndexedDB
Cookies
http://localhost:808
Private state tokens
Interest groups
Shared storage
Cache storage
Filter
Name
Value
D...
P.
JSESSION...
950ADD36E200E...
I...
/
Cookie Value
Show URL-decoded
950ADD36E200E2854A886A24AD572F19
Are you sure you want to log out?
Log Out

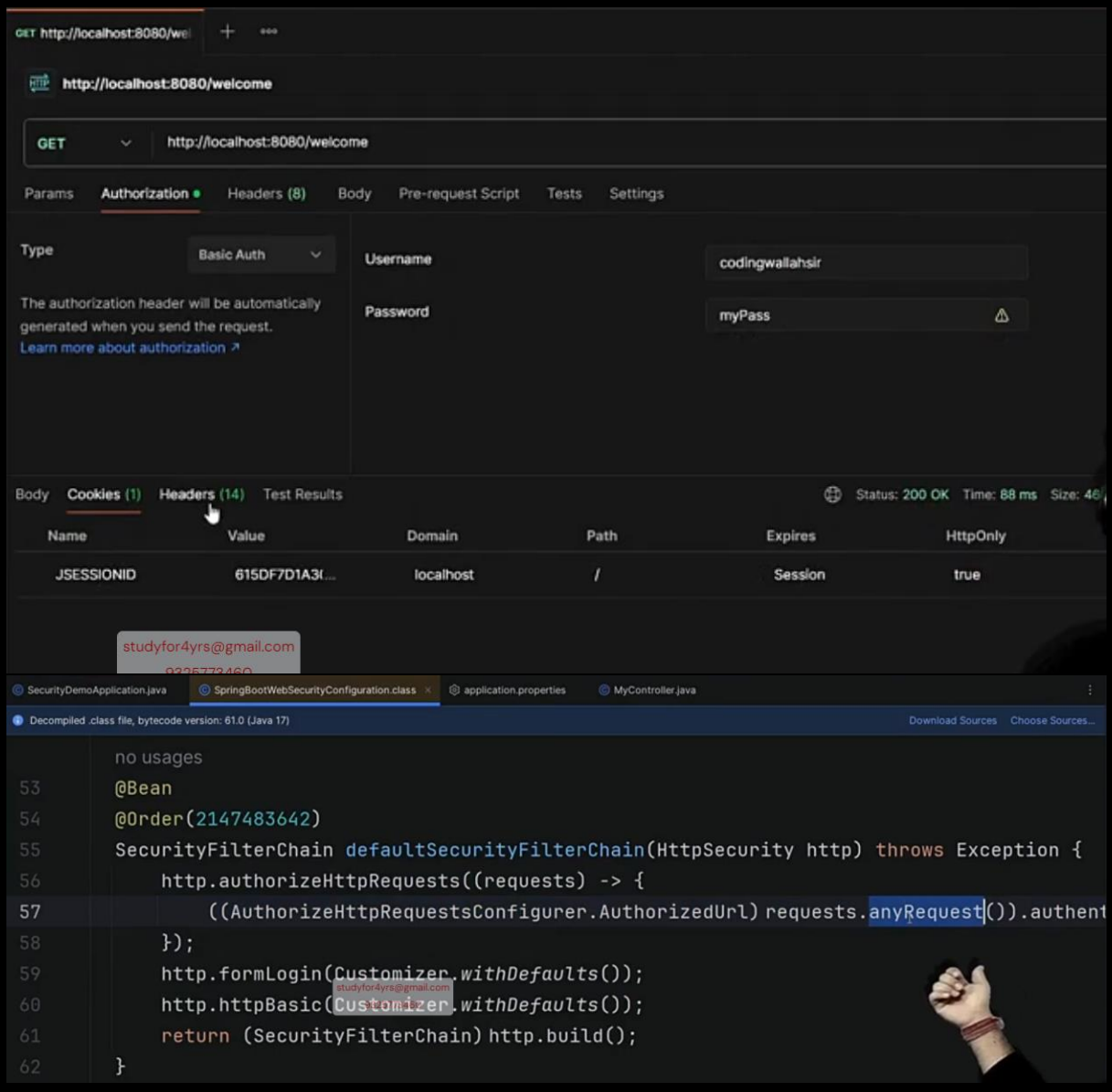

Please sign in
You have been signed out
Username
Password
Sign in
Network
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
Hide extension URLs
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
WS
Wasm
Other
Blocked response cookies
Blocked requests
3rd-party requests
20 ms
40 ms
60 ms
80 ms
100 ms
120 ms
Name
Status
Type
Initiator
Size
Time
logout 302 docum... Other 426 B 13 ms
login?logout 200 docum... logout 2.1 kB 20 ms
bootstrap.min.css 200 stylesh... login?logout:8 0 B 15 ms
signin.css 200 stylesh... login?logout:9 0 B 12 ms
4 requests 2.5 kB transferred 127 kB resources Finish: 92 ms DOMContentLoaded: 39 ms
Console
Issues
Network conditions
Caching
Disable cache
Network throttling
No throttling

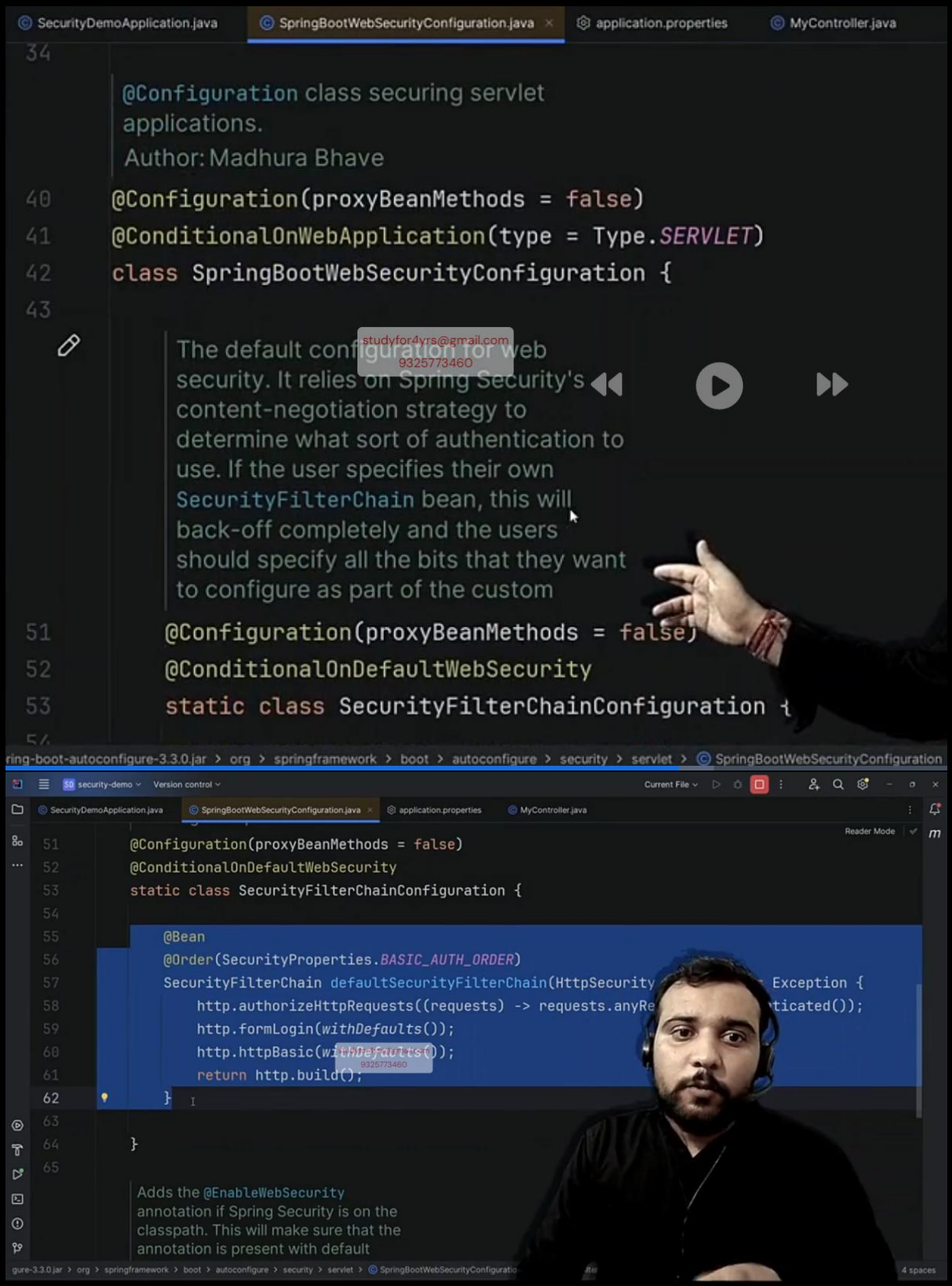

SecurityDemoApplication.java
application.properties
MyController.java
1 spring.application.name=security-demo
2 spring.security.user.password=myPass

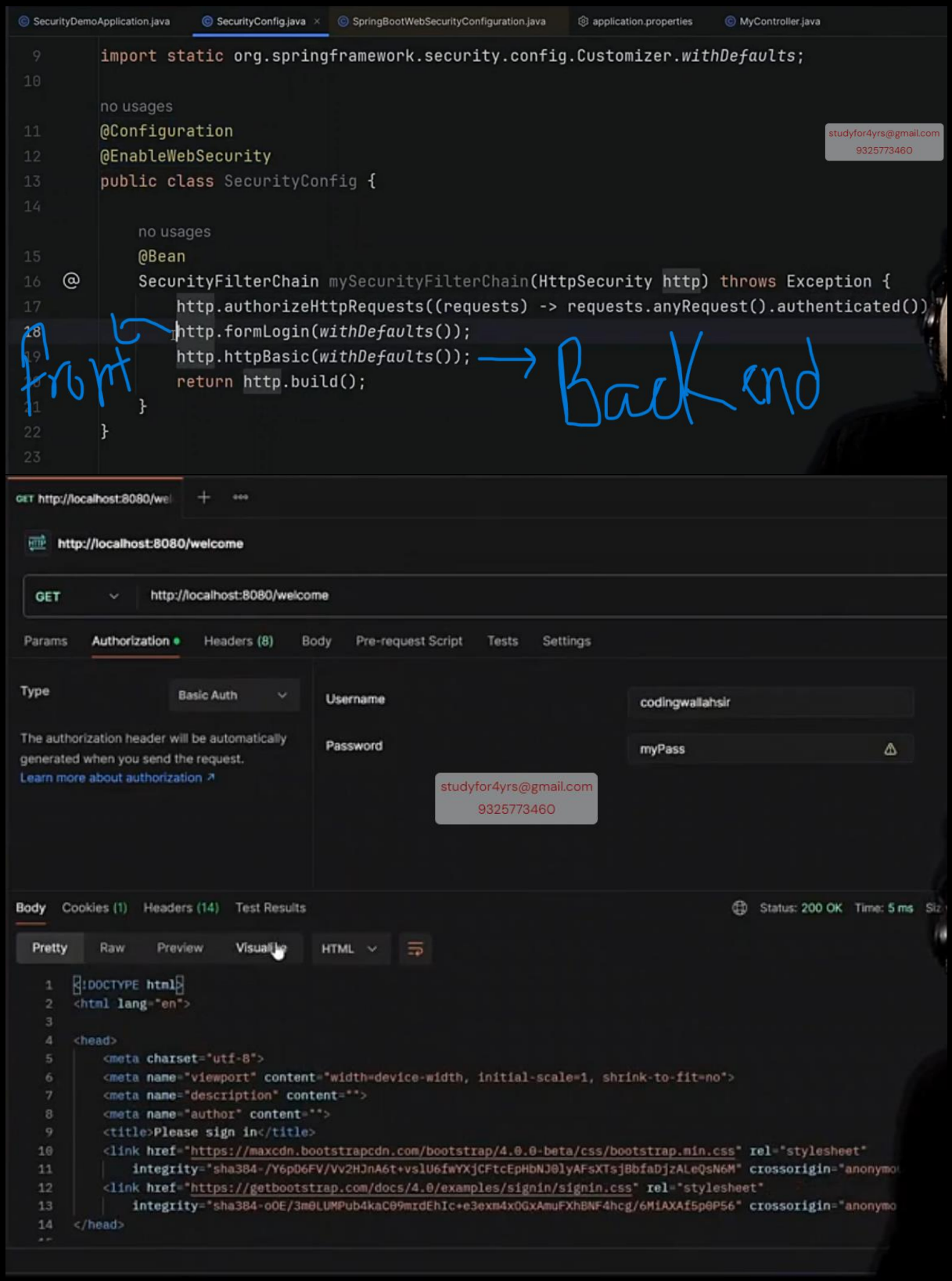
SecurityDemoApplication.java
application.properties
MyController.java
1 spring.application.name=security-demo
2 spring.security.user.password=myPass
3 spring.security.user.name=codingwallahsir

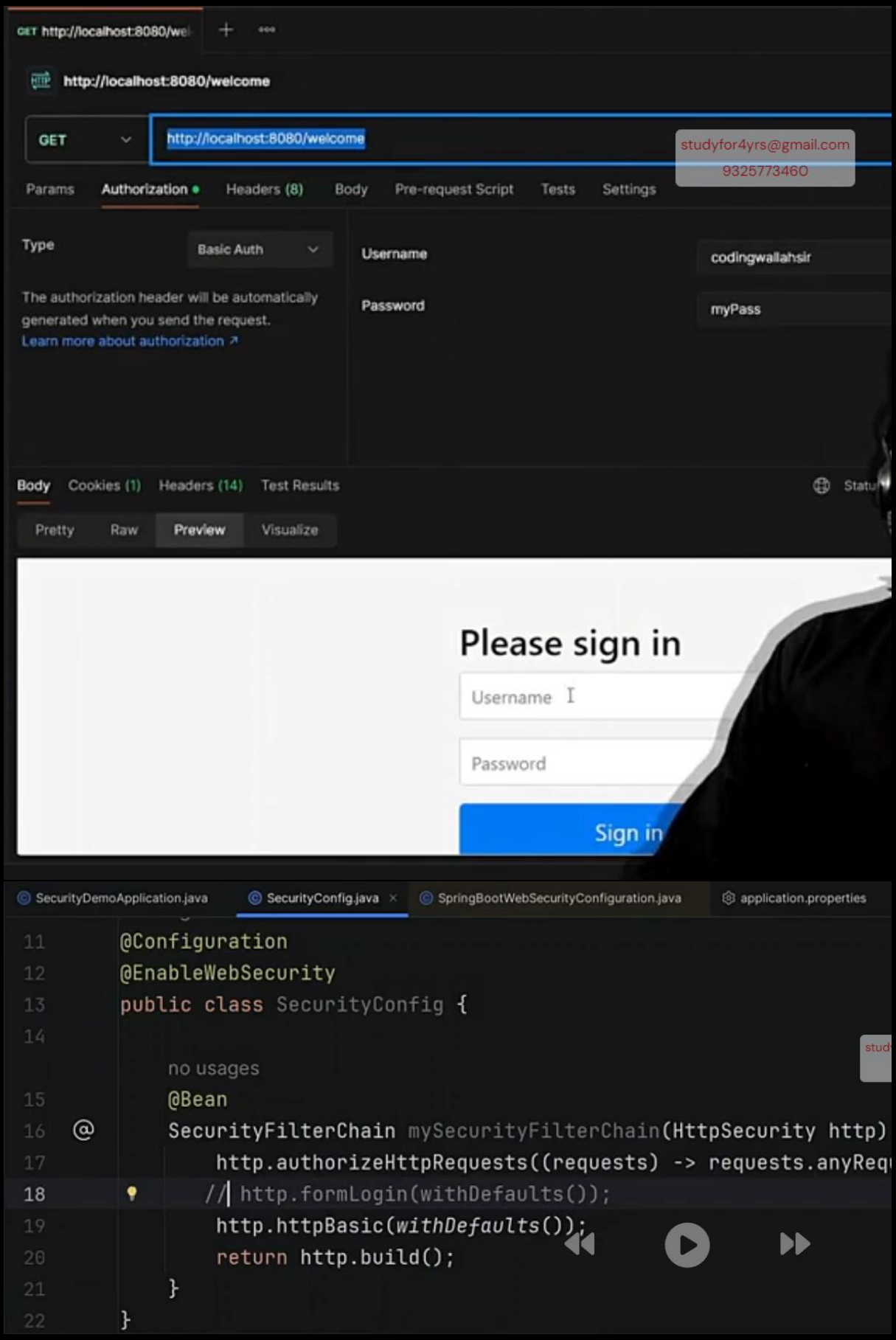

Please sign in
You have been signed out
codingwallahsir
Sign in

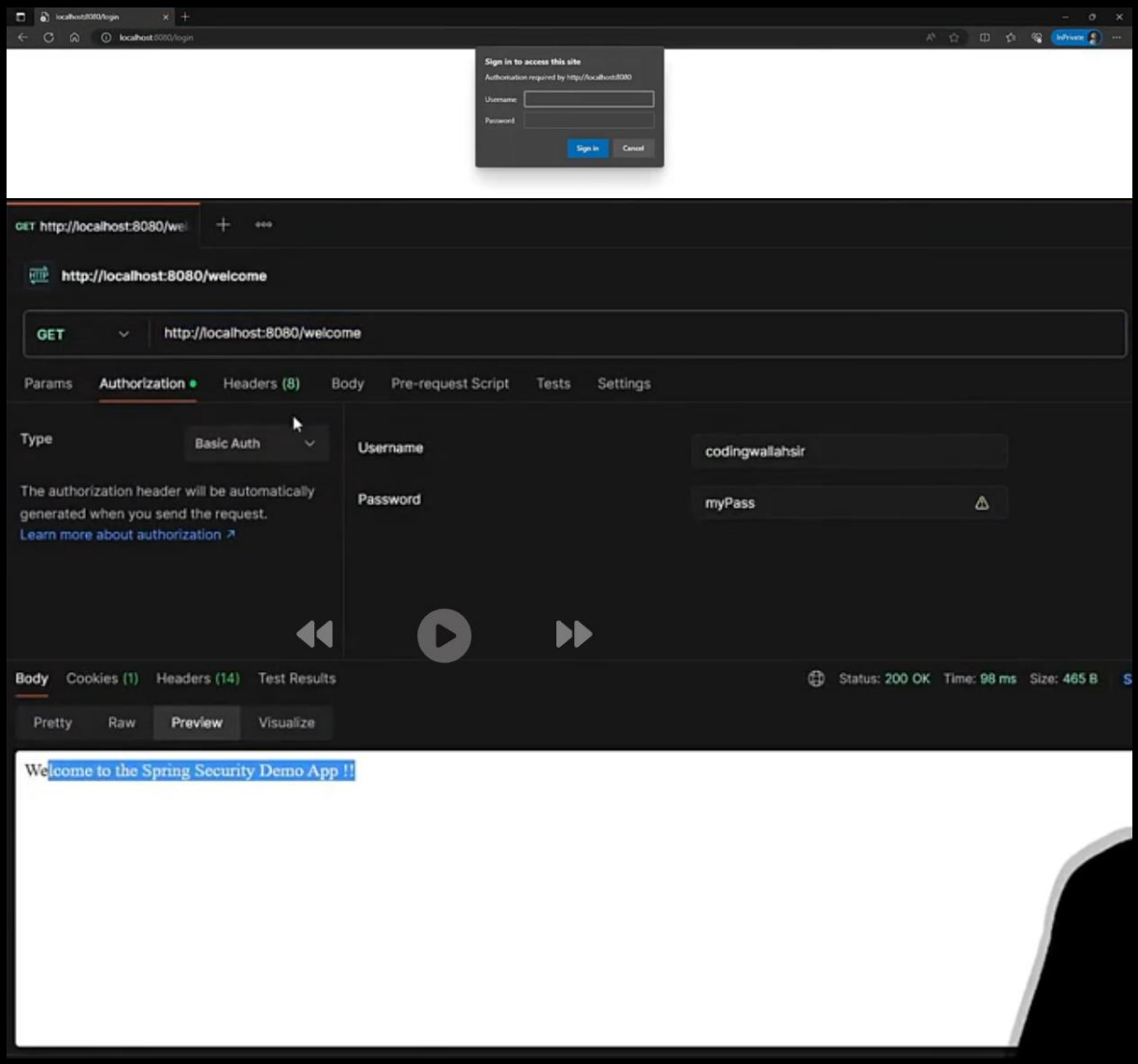

GET http://localhost:8080/wel
http://localhost:8080/welcome
GET
http://localhost:8080/welcome
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Type
Basic Auth
The authorization header will be automatically generated when you send the request.
Learn more about authorization
Username
codingwallahsir
Password
myPass

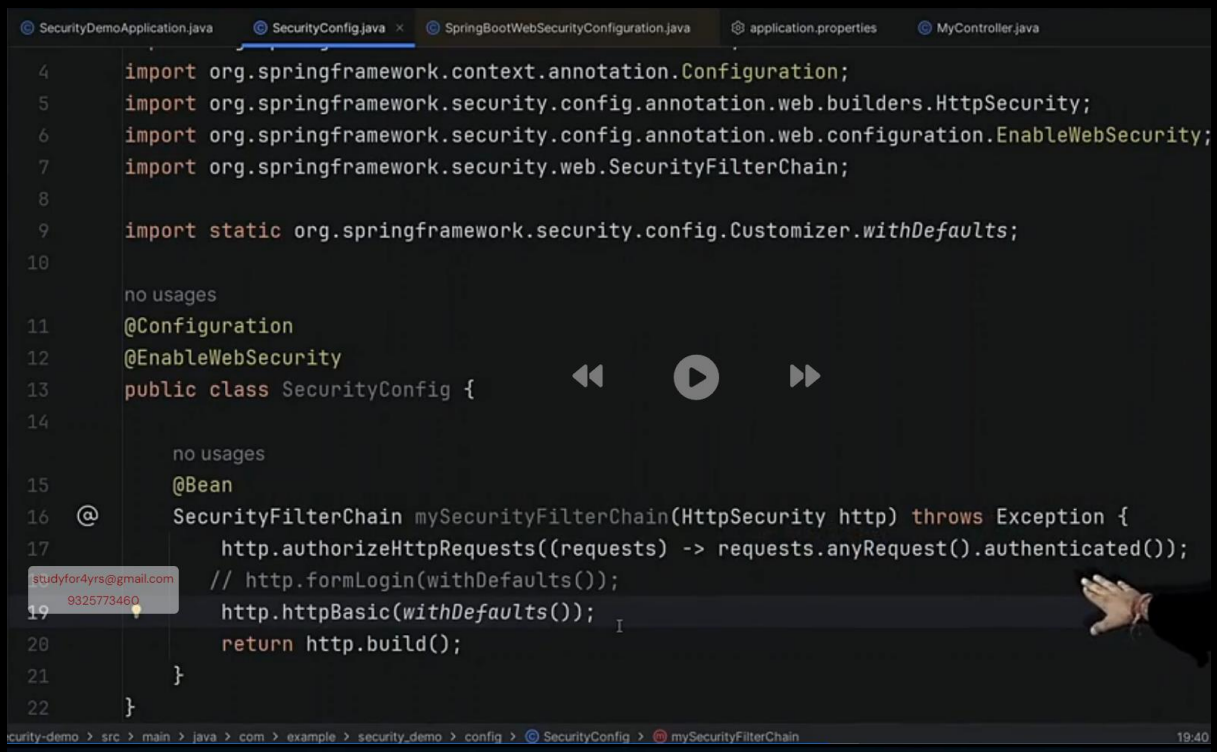

GET http://localhost:8080/wel
http://localhost:8080/welcome
GET
http://localhost:8080/welcome
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Type
Basic Auth
Username
codingwallahsir
The authorization header will be automatically generated when you send the request.
Learn more about authorization
Password
myPass
Body
Cookies (1)
Headers (14)
Test Results
Status: 200 OK
Time: 88 ms
Name
Value
Domain
Path
Expires
HttpOnly
JSESSIONID
615DF7D1A3(...
localhost
/
Session
true
studyfor4yrs@gmail.com
SecurityDemoApplication.java
SpringBootWebSecurityConfiguration.class
application.properties
MyController.java
Decompiled .class file, bytecode version: 61.0 (Java 17)
Download Sources
Choose Sources...
no usages
@Bean
@Order(2147483642)
SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http) throws Exception {
http.authorizeHttpRequests((requests) -> {
((AuthorizeHttpRequestsConfigurer.AuthorizedUrl) requests.anyRequest()).authent
});
http.formLogin(Customizer.withDefaults());
http.httpBasic(Customizer.withDefaults());
return (SecurityFilterChain) http.build();
}

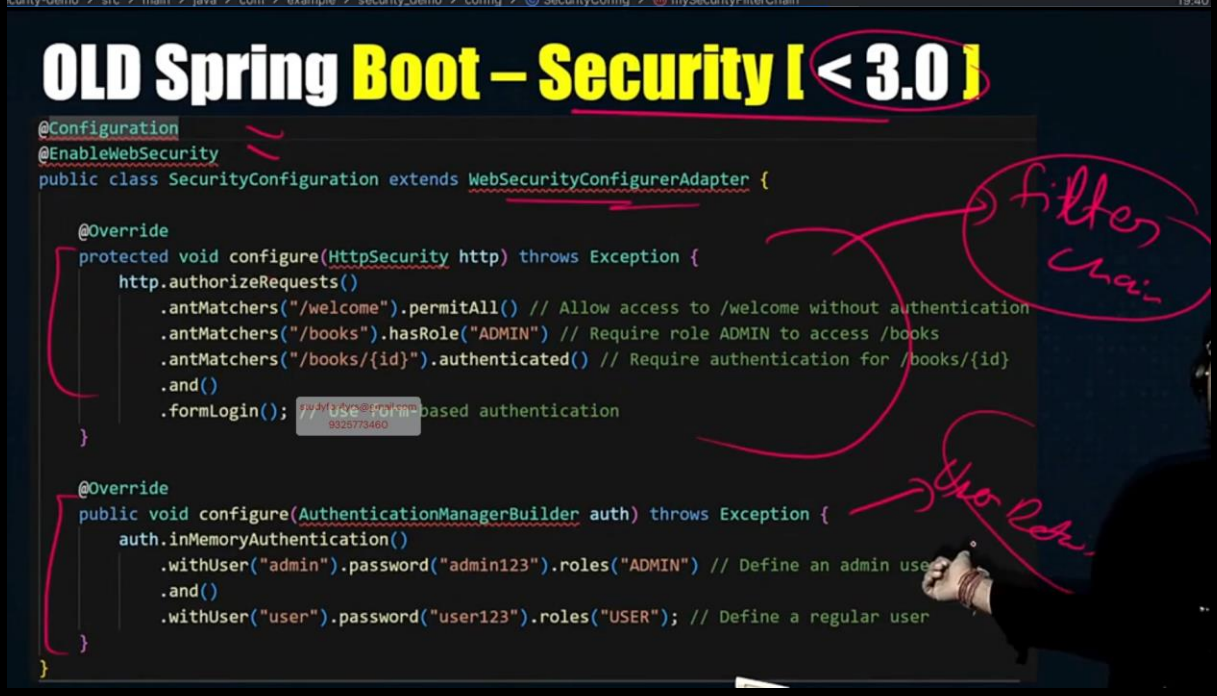

SecurityDemoApplication.java
SpringBootWebSecurityConfiguration.java
application.properties
MyController.java
@Configuration class securing servlet applications.
Author: Madhura Bhave
@Configuration(proxyBeanMethods = false)
@ConditionalOnWebApplication(type = Type.SERVLET)
class SpringBootWebSecurityConfiguration {
The default configuration for web security. It relies on Spring Security's content-negotiation strategy to determine what sort of authentication to use. If the user specifies their own SecurityFilterChain bean, this will back-off completely and the users should specify all the bits that they want to configure as part of the custom
@Configuration(proxyBeanMethods = false)
@ConditionalOnDefaultWebSecurity
static class SecurityFilterChainConfiguration {
studyfor4yrs@gmail.com
9325773460
@Bean
@Order(SecurityProperties.BASIC_AUTH_ORDER)
SecurityFilterChain defaultSecurityFilterChain(HttpSecurity
Exception {
http.authorizeHttpRequests((requests) -> requests.anyRe
ticated());
http.formLogin(withDefaults());
http.httpBasic(withDefaults());
return http.build();
}
}
Adds the @EnableWebSecurity annotation if Spring Security is on the classpath. This will make sure that the annotation is present with default


SecurityDemoApplication.java
SecurityConfig.java
SpringBootWebSecurityConfiguration.java
application.properties
MyController.java
import static org.springframework.security.config.Customizer.withDefaults;
no usages
@Configuration
@EnableWebSecurity
public class SecurityConfig {
no usages
@Bean
SecurityFilterChain mySecurityFilterChain(HttpSecurity http) throws Exception {
http.authorizeHttpRequests((requests) -> requests.anyRequest().authenticated());
http.formLogin(withDefaults());
http.httpBasic(withDefaults());
return http.build();
}
}
studyfor4yrs@gmail.com
9325773460
Front
Backend
GET http://localhost:8080/welcome
http://localhost:8080/welcome
GET
http://localhost:8080/welcome
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Type
Basic Auth
The authorization header will be automatically generated when you send the request.
Learn more about authorization
Username
codingwallahsir
Password
myPass
studyfor4yrs@gmail.com
9325773460
Body
Cookies (1)
Headers (14)
Test Results
Status: 200 OK
Time: 5 ms
Pretty
Raw
Preview
Visualize
HTML
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="">
<meta name="author" content="">
<title>Please sign in</title>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-/Y6pD6FV/Vv2HJnA6t+vslU6fwYXjCFtcEpHbNJ0lyAFsXTsjBbfaDjzALeQsN6M" crossorigin="anonymo
<link href="https://getbootstrap.com/docs/4.0/examples/signin/signin.css" rel="stylesheet"
integrity="sha384-oOE/3m0LUMPub4kaC09mrdEhIc+e3exm4xOGxAmuFXhBNF4hcg/6MiAXAf5p0P56" crossorigin="anonymo
</head>


GET http://localhost:8080/welcome
http://localhost:8080/welcome
GET
http://localhost:8080/welcome
studyfor4yrs@gmail.com
9325773460
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Type
Basic Auth
Username
codingwallahsir
The authorization header will be automatically generated when you send the request.
Learn more about authorization
Password
myPass
Body
Cookies (1)
Headers (14)
Test Results
Pretty
Raw
Preview
Visualize
Please sign in
Username
Password
Sign in
SecurityDemoApplication.java
SecurityConfig.java
SpringBootWebSecurityConfiguration.java
application.properties
@Configuration
@EnableWebSecurity
public class SecurityConfig {
no usages
@Bean
SecurityFilterChain mySecurityFilterChain(HttpSecurity http)
http.authorizeHttpRequests((requests) -> requests.anyReq
// http.formLogin(withDefaults());
http.httpBasic(withDefaults());
return http.build();
}
}


Sign in to access this site
Authorisation required by http://localhost:8080
Username
Password
Sign in
Cancel
GET http://localhost:8080/we
http://localhost:8080/welcome
GET
http://localhost:8080/welcome
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Type
Basic Auth
The authorization header will be automatically generated when you send the request.
Learn more about authorization
Username
codingwallahsir
Password
myPass
Body
Cookies (1)
Headers (14)
Test Results
Status: 200 OK
Time: 98 ms
Size: 465 B
Pretty
Raw
Preview
Visualize
Welcome to the Spring Security Demo App !!

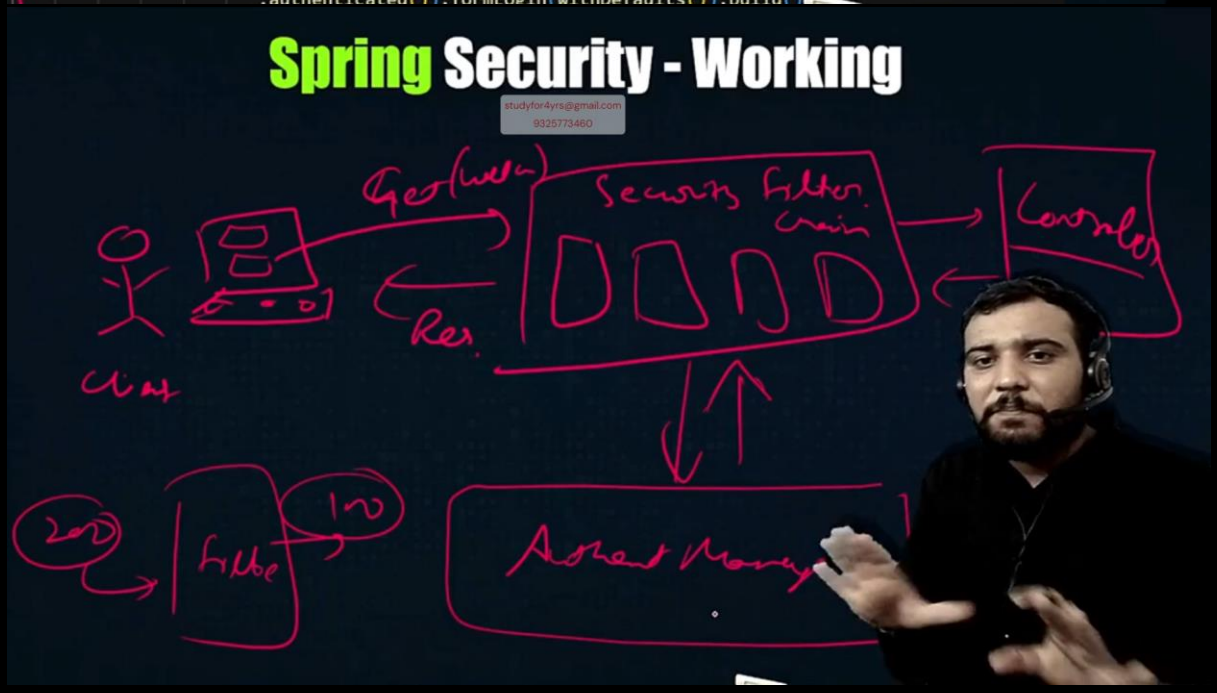

SecurityDemoApplication.java
SecurityConfig.java
SpringBootWebSecurityConfiguration.java
application.properties
MyController.java
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.web.SecurityFilterChain;
import static org.springframework.security.config.Customizer.withDefaults;
no usages
@Configuration
@EnableWebSecurity
public class SecurityConfig {
no usages
@Bean
SecurityFilterChain mySecurityFilterChain(HttpSecurity http) throws Exception {
http.authorizeHttpRequests((requests) -> requests.anyRequest().authenticated());
// http.formLogin(withDefaults());
http.httpBasic(withDefaults());
return http.build();
}
}
curity-demo > src > main > java > com > example > security_demo > config > SecurityConfig > mySecurityFilterChain
19:40

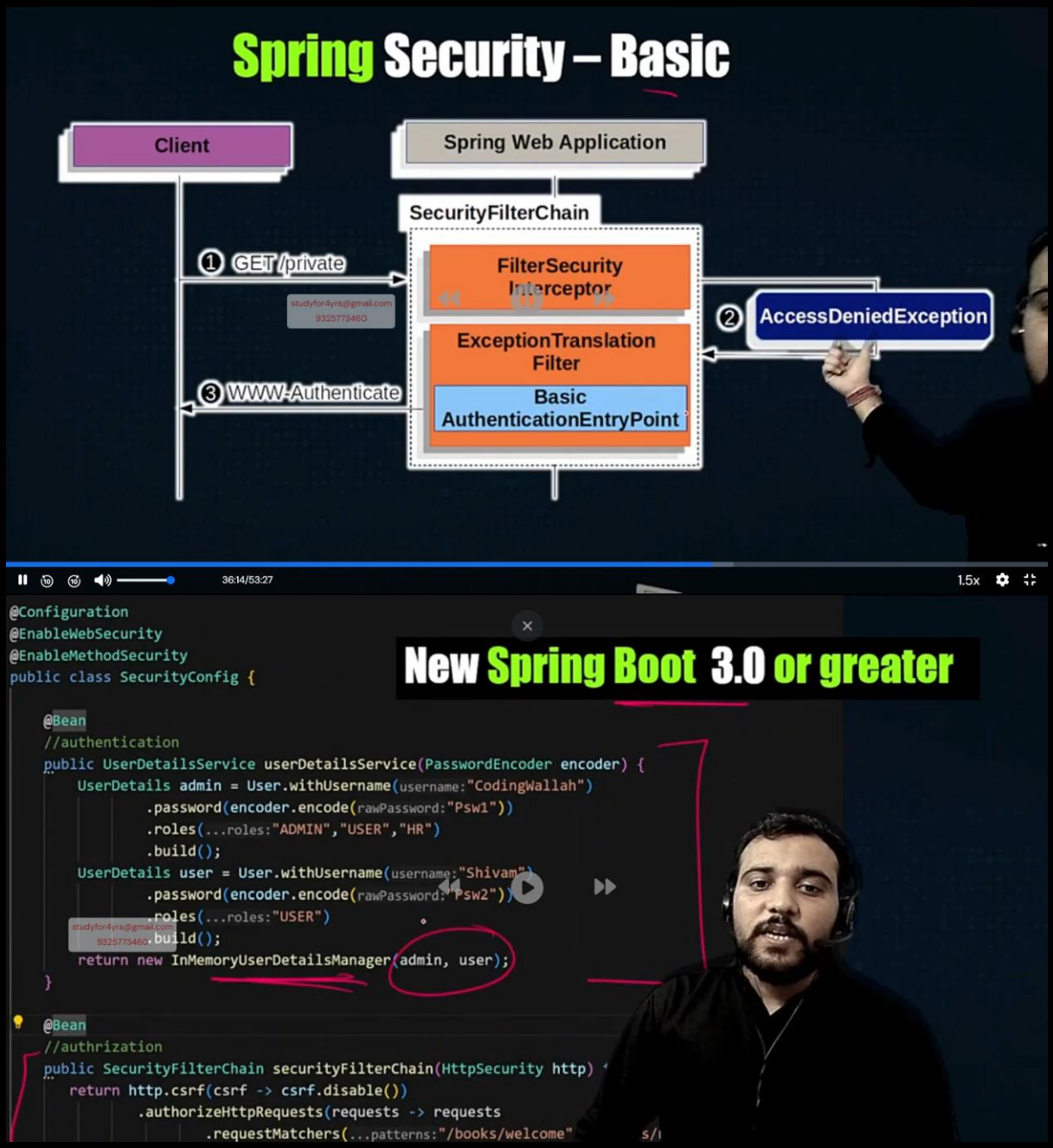

OLD Spring Boot – Security [< 3.0]
@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
@Override
protected void configure(HttpSecurity http) throws Exception {
http.authorizeRequests()
.antMatchers("/welcome").permitAll() // Allow access to /welcome without authentication
.antMatchers("/books").hasRole("ADMIN") // Require role ADMIN to access /books
.antMatchers("/books/{id}").authenticated() // Require authentication for /books/{id}
.and()
.formLogin(); // Use form-based authentication
}
@Override
public void configure(AuthenticationManagerBuilder auth) throws Exception {
auth.inMemoryAuthentication()
.withUser("admin").password("admin123").roles("ADMIN") // Define an admin user
.and()
.withUser("user").password("user123").roles("USER"); // Define a regular user
}
}
filter chain

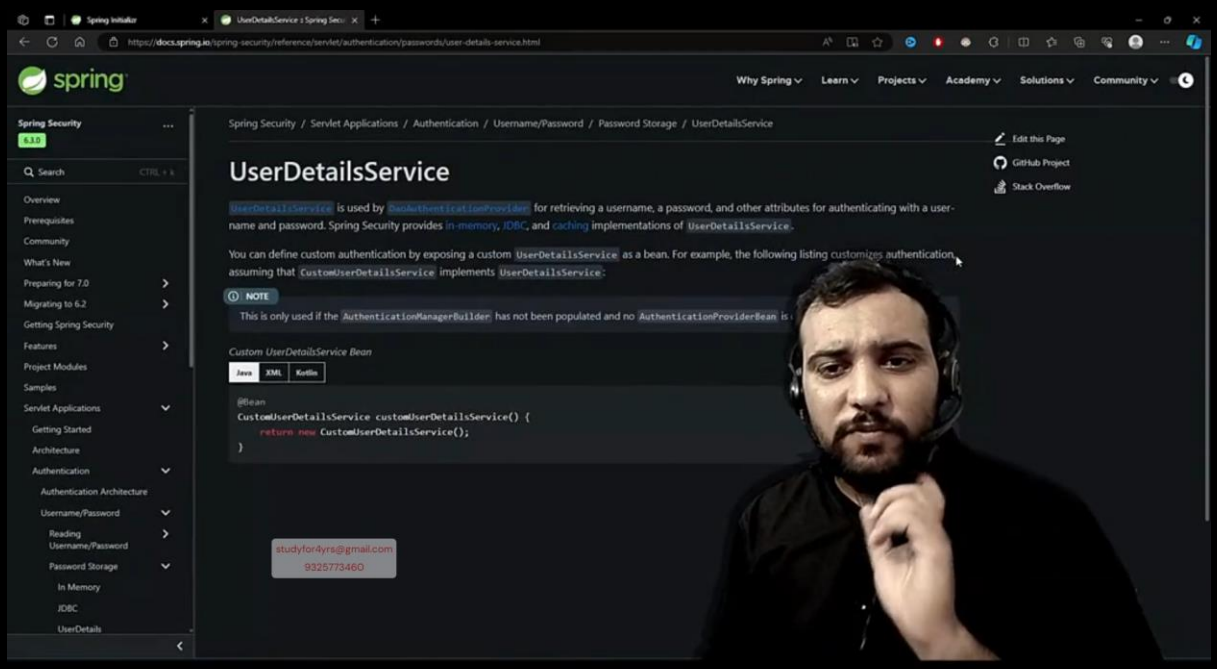
```java
@Configuration
@EnableWebSecurity
@EnableMethodSecurity
public class SecurityConfig {

    @Bean
    //authentication
    public UserDetailsService userDetailsService(PasswordEncoder encoder) {
        UserDetails admin = User.withUsername(username:"CodingWallah")
                .password(encoder.encode(rawPassword:"Psw1"))
                .roles(...roles:"ADMIN","USER","HR")
                .build();
        UserDetails user = User.withUsername(username:"Shivam")
                .password(encoder.encode(rawPassword:"Psw2"))
                .roles(...roles:"USER")
                .build();
        return new InMemoryUserDetailsManager(admin, user);
    }

    @Bean
    //authrization
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        return http.csrf(csrf -> csrf.disable())
                .authorizeHttpRequests(requests -> requests
                        .requestMatchers(...patterns:"/books/welcome", "/books/new").permitAll())
                .authorizeHttpRequests(requests -> requests.requestMatchers(...patterns:"/books/**")
                        .authenticated()).formLogin(withDefaults()).build()
```

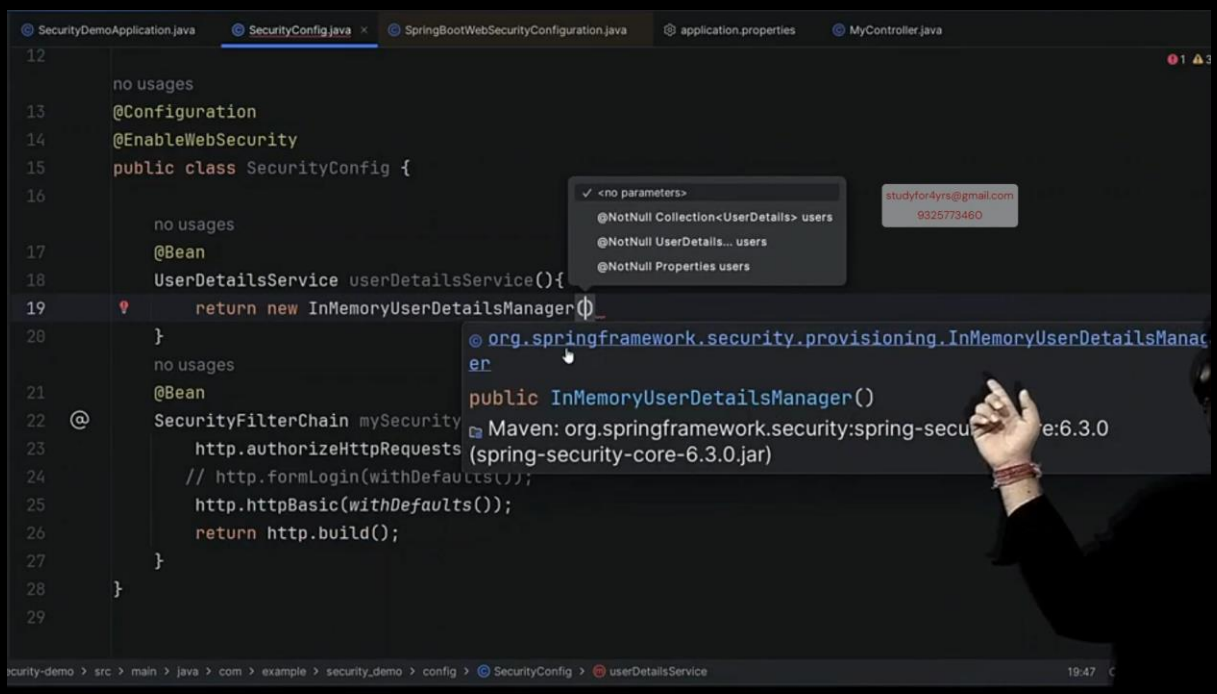
New Spring Boot 3.0 or greater

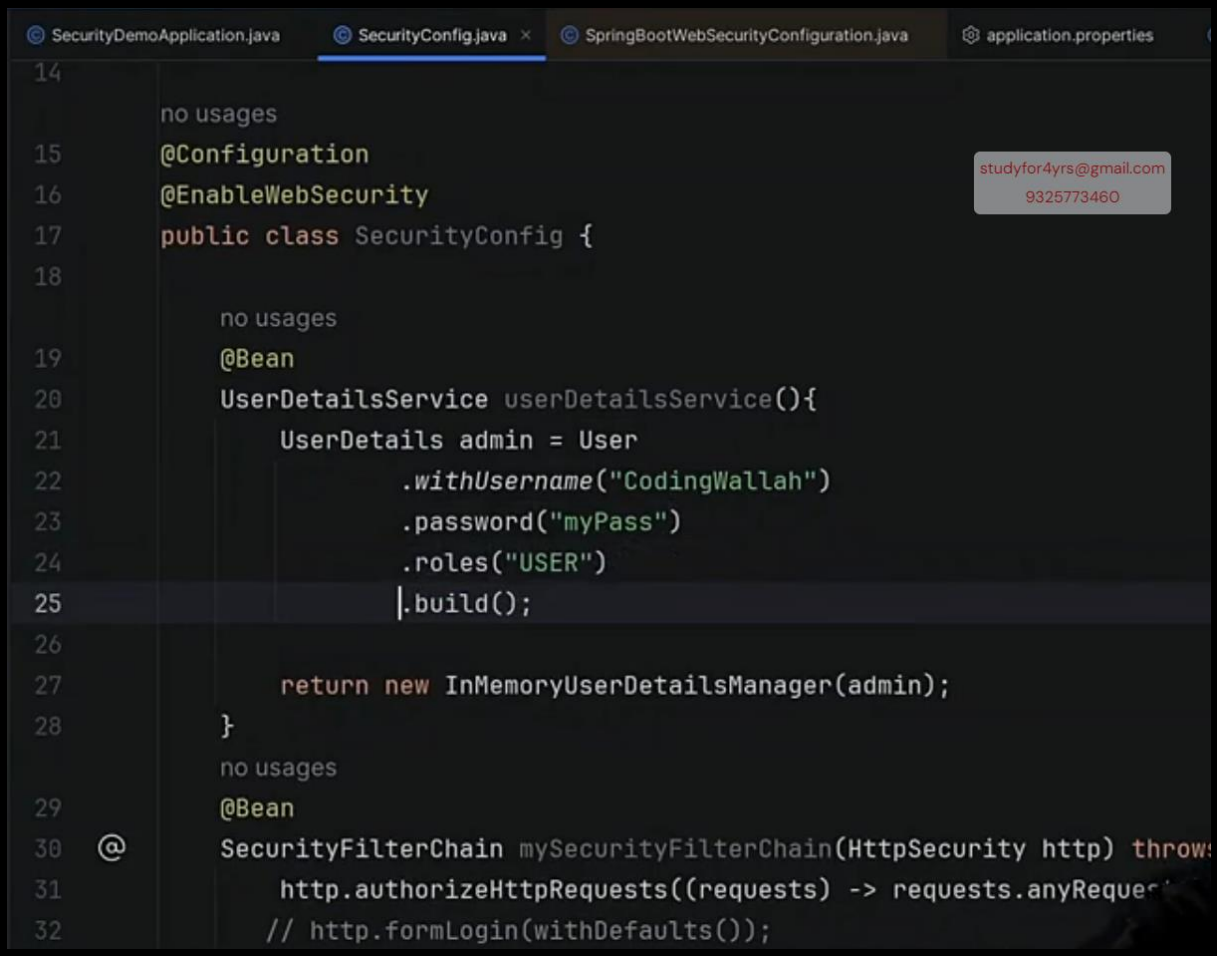
Spring Security - Working

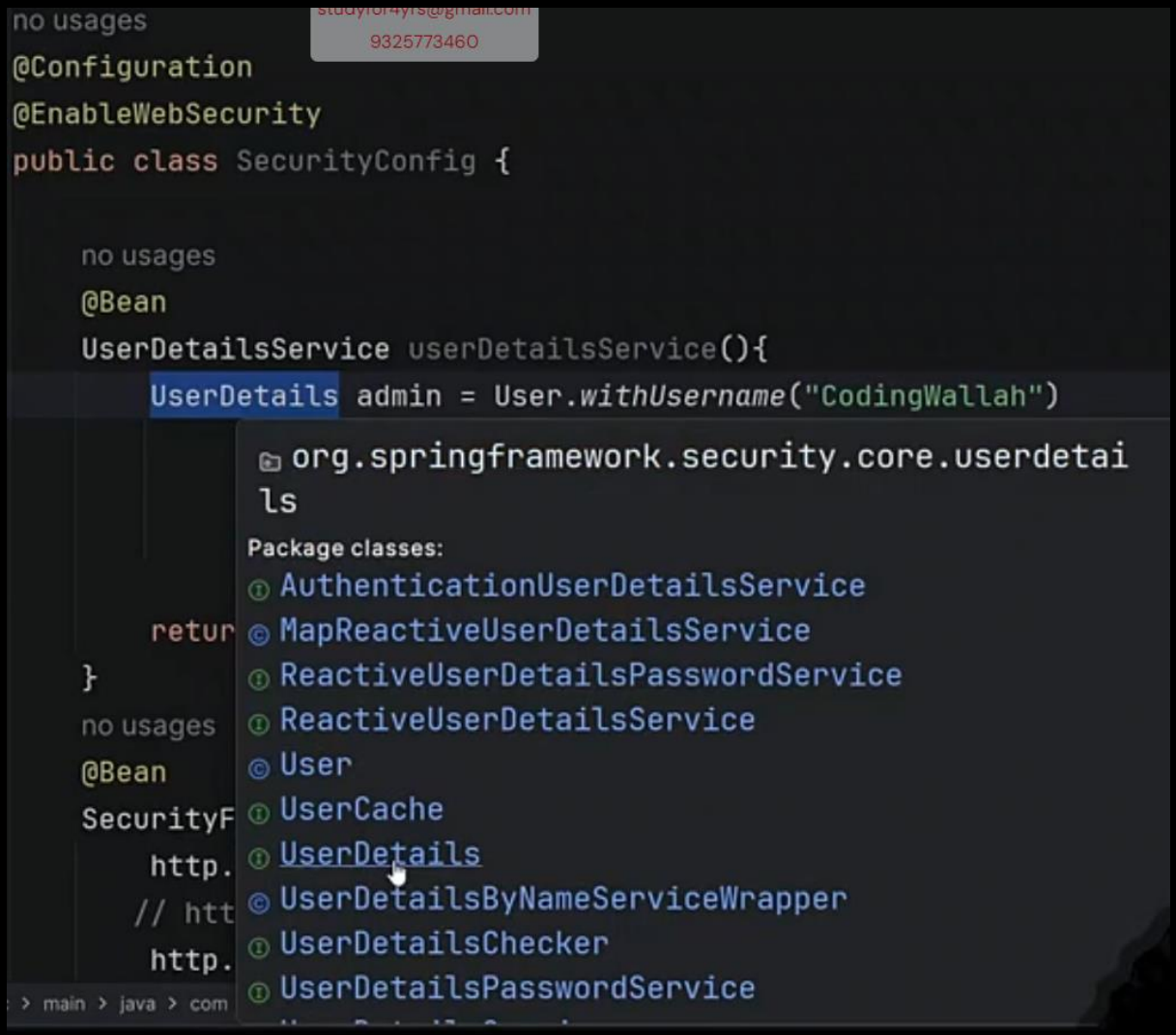



Spring Security – Basic
Client
Spring Web Application
SecurityFilterChain
1 GET /private
FilterSecurity Interceptor
2 AccessDeniedException
ExceptionTranslation Filter
3 WWW-Authenticate
Basic AuthenticationEntryPoint
36:14/53:27
1.5x
New Spring Boot 3.0 or greater
@Configuration
@EnableWebSecurity
@EnableMethodSecurity
public class SecurityConfig {
@Bean
//authentication
public UserDetailsService userDetailsService(PasswordEncoder encoder) {
UserDetails admin = User.withUsername(username:"CodingWallah")
.password(encoder.encode(rawPassword:"Psw1"))
.roles(...roles:"ADMIN","USER","HR")
.build();
UserDetails user = User.withUsername(username:"Shivam")
.password(encoder.encode(rawPassword:"Psw2"))
.roles(...roles:"USER")
.build();
return new InMemoryUserDetailsManager(admin, user);
}
@Bean
//authrization
public SecurityFilterChain securityFilterChain(HttpSecurity http)
return http.csrf(csrf -> csrf.disable())
.authorizeHttpRequests(requests -> requests
.requestMatchers(...patterns:"/books/welcome"


spring
Why Spring
Learn
Projects
Academy
Solutions
Community
Spring Security
Search
Overview
Prerequisites
Community
What's New
Preparing for 7.0
Migrating to 6.2
Getting Spring Security
Features
Project Modules
Samples
Servlet Applications
Getting Started
Architecture
Authentication
Authentication Architecture
Username/Password
Reading Username/Password
Password Storage
In Memory
JDBC
UserDetails
Spring Security / Servlet Applications / Authentication / Username/Password / Password Storage / UserDetailsService
Edit this Page
GitHub Project
Stack Overflow
UserDetailsService
UserDetailsService is used by DaoAuthenticationProvider for retrieving a username, a password, and other attributes for authenticating with a user-name and password. Spring Security provides in-memory, JDBC, and caching implementations of UserDetailsService.
You can define custom authentication by exposing a custom UserDetailsService as a bean. For example, the following listing customizes authentication, assuming that CustomUserDetailsService implements UserDetailsService:
NOTE
This is only used if the AuthenticationManagerBuilder has not been populated and no AuthenticationProviderBean is
Custom UserDetailsService Bean
Java
XML
Kotlin
@Bean
CustomUserDetailsService customUserDetailsService() {
return new CustomUserDetailsService();
}
studyfor4yrs@gmail.com
9325773460


SecurityDemoApplication.java
SecurityConfig.java
SpringBootWebSecurityConfiguration.java
application.properties
MyController.java
no usages
@Configuration
@EnableWebSecurity
public class SecurityConfig {
no usages
@Bean
UserDetailsService userDetailsService(){
return new InMemoryUserDetailsManager()
}
no usages
@Bean
SecurityFilterChain mySecurity
http.authorizeHttpRequests
// http.formLogin(withDefaults());
http.httpBasic(withDefaults());
return http.build();
}
}
<no parameters>
@NotNull Collection<UserDetails> users
@NotNull UserDetails... users
@NotNull Properties users
studyfor4yrs@gmail.com
9325773460
org.springframework.security.provisioning.InMemoryUserDetailsManager
public InMemoryUserDetailsManager()
Maven: org.springframework.security:spring-secu e:6.3.0
(spring-security-core-6.3.0.jar)
curity-demo > src > main > java > com > example > security_demo > config > SecurityConfig > userDetailsService
19:47


SecurityDemoApplication.java
SecurityConfig.java
SpringBootWebSecurityConfiguration.java
application.properties
no usages
@Configuration
@EnableWebSecurity
public class SecurityConfig {
no usages
@Bean
UserDetailsService userDetailsService(){
UserDetails admin = User
.withUsername("CodingWallah")
.password("myPass")
.roles("USER")
.build();
return new InMemoryUserDetailsManager(admin);
}
no usages
@Bean
SecurityFilterChain mySecurityFilterChain(HttpSecurity http) throws
http.authorizeHttpRequests((requests) -> requests.anyReque
// http.formLogin(withDefaults());
studyfor4yrs@gmail.com
9325773460



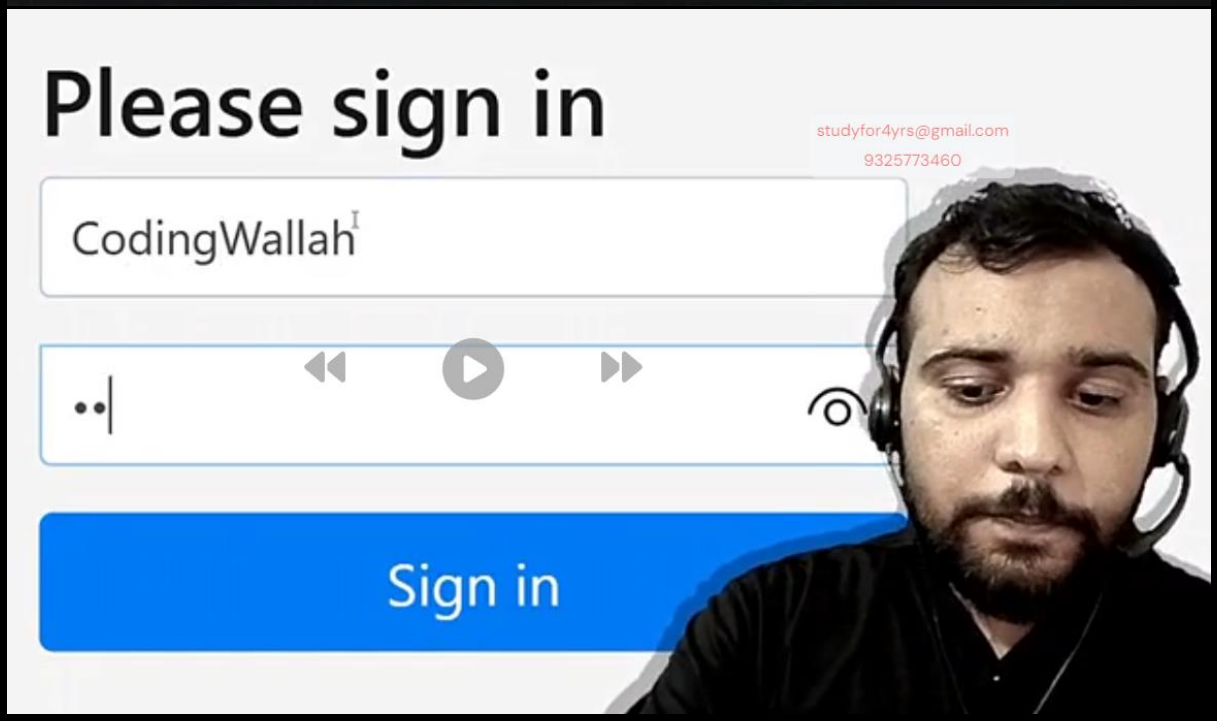

```
@Bean
UserDetailsService userDetailsService(){
    UserDetails admin = User.withUsername("CodingWallah")
                            .password("{noop}myPass")
                            .roles("ADM
                            .build();

    return new InMemoryUserDetailsManager(admin);
}
```

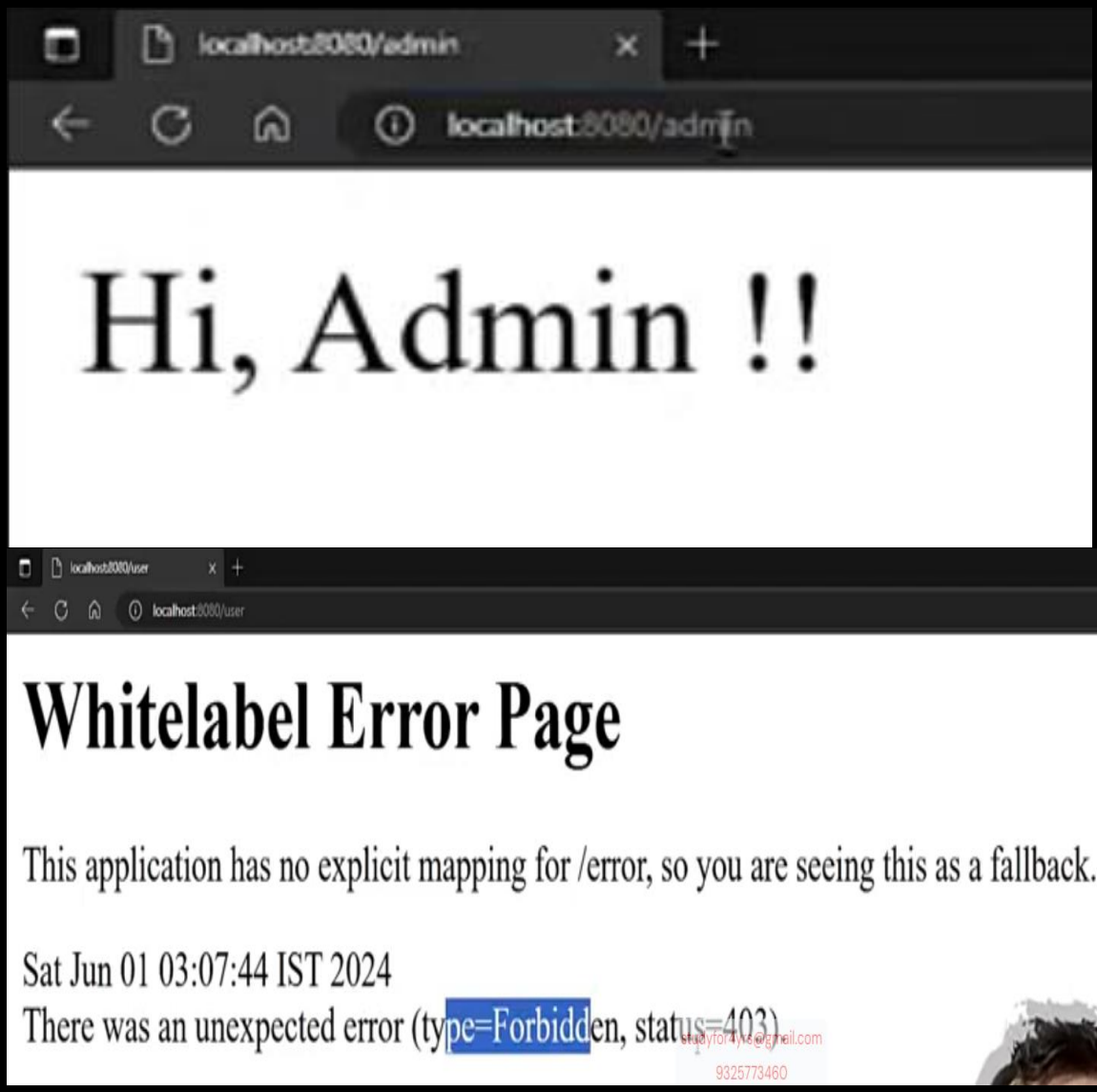
```
no usages
@Bean
UserDetailsService userDetailsService(){
    UserDetails admin = User.withUsername("CodingWallah")
                            .password("{noop}myPass")
                            .roles("ADMIN")
                            .build();

    UserDetails user = User.withUsername("shivam")
            .password("{noop}1234")
            .roles("USER")
            .build();
```

```
no usages
@Configuration
@EnableWebSecurity
@EnableMethodSecurity
public class SecurityConfig {

    no usages
    @Bean
    UserDetailsService userDetailsService(){
        UserDetails admin = User.withUsername("CodingWallah")
                                .password("{noop}myPass")
                                .roles("ADMIN")
                                .build();

        UserDetails user = User.withUsername("shivam")
                .password("{noop}1234")
                .roles("USER")
```

```
no usages
@GetMapping("/welcome")
public String welcome(){
    return "Welcome to the Spring Security Demo App !!";
}

no usages
@GetMapping("/user")
public String userService(){
    return "Hi, User !!";
}

no usages
@GetMapping("/admin")
public String adminService(){
    return "Hi, Admin !!";
}
```

```
no usages
@GetMapping("/welcome")
public String welcome(){
    return "Welcome to the Spring Security Demo App !!";
}

no usages
@GetMapping("/user")
@PreAuthorize("hasAuthority('ROLE_USER')")
public String userService(){
    return "Hi, User !!";
}

no usages
@GetMapping("/admin")
@PreAuthorize("hasAuthority('ROLE_ADMIN')")
public String adminService(){
    return "Hi, Admin !!";
```




localhost:8080/admin
localhost:8080/admin
Hi, Admin !!
localhost:8080/user
localhost:8080/user
Whitelabel Error Page
This application has no explicit mapping for /error, so you are seeing this as a fallback.
Sat Jun 01 03:07:44 IST 2024
There was an unexpected error (type=Forbidden, status=403)

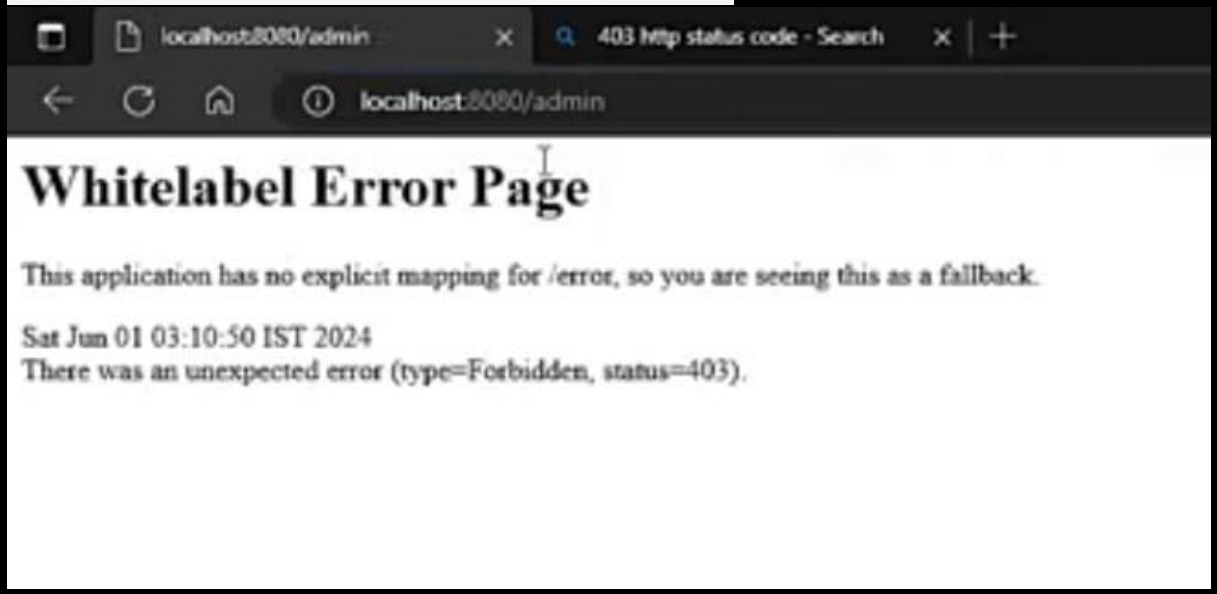
403 Forbidden

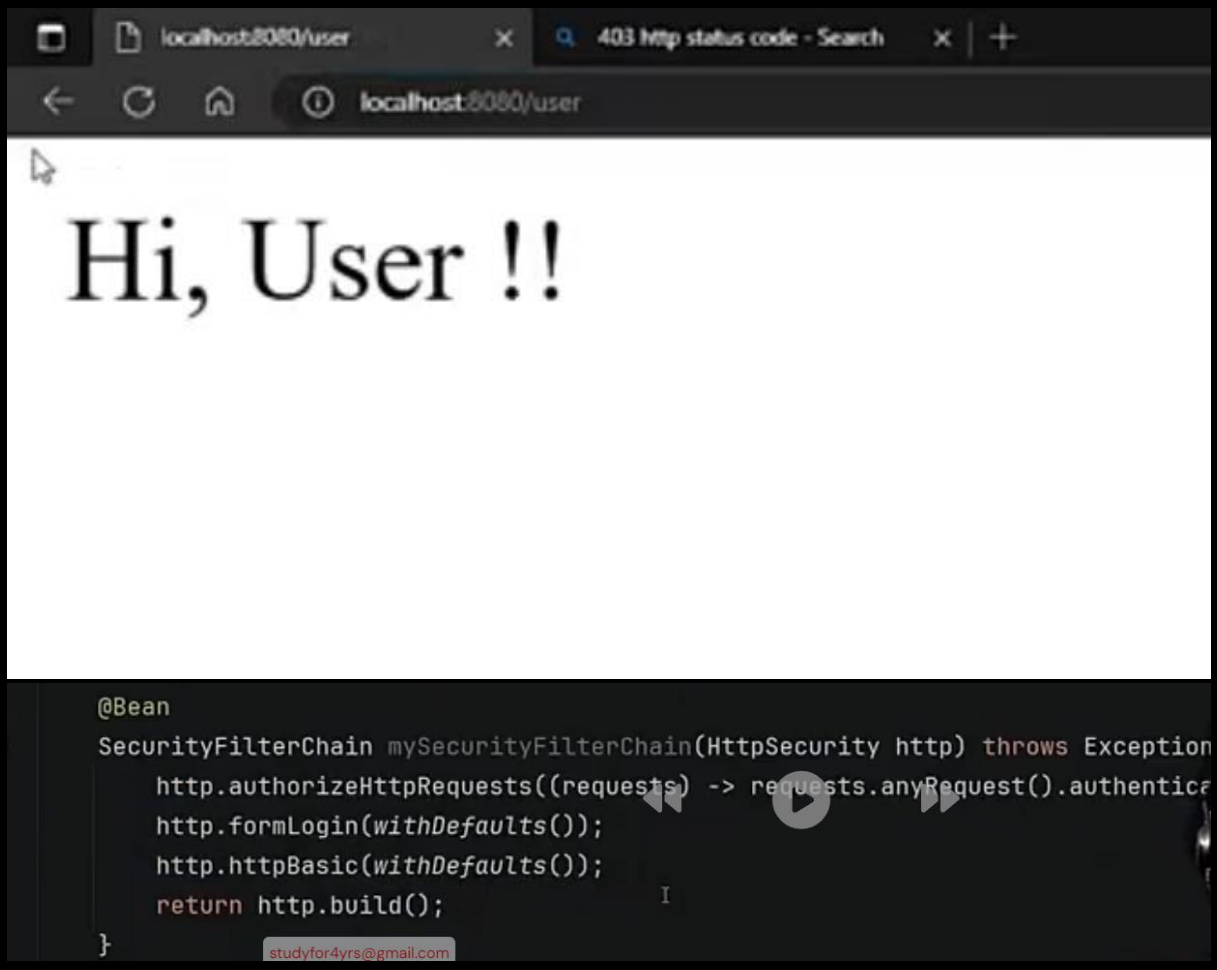
According to 2 sources

HTTP response status code 403 **Forbidden** is a client error that is returned by the server to indicate that the client does not have access to the requested resource, and does not offer an Authentication scheme by which access can be granted.

SecurityDemoApplication.java | SecurityConfig.java | SpringBootWebSecurityConfiguration.java

```
                .roles("ADMIN")
                .build();

        UserDetails user = User.withUsername("shivam")
                .password("{noop}1234")
                .roles("USER")
                .build();

        return new InMemoryUserDetailsManager(admin, user);
```


localhost:8080/user
403 http status code - Search
localhost:8080/user
Hi, User !!
@Bean
SecurityFilterChain mySecurityFilterChain(HttpSecurity http) throws Exception
http.authorizeHttpRequests((requests) -> requests.anyRequest().authentica
http.formLogin(withDefaults());
http.httpBasic(withDefaults());
return http.build();
}
studyfor4yrs@gmail.com