* **JWT HANDS ON**
* **Create authentication service that returns JWT**

**AuthenticationController.java**

package com.cognizant.spring\_learn;

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.web.bind.annotation.\*;

import org.springframework.beans.factory.annotation.Autowired;

import jakarta.servlet.http.HttpServletRequest;

import java.util.Base64;

import com.cognizant.spring\_learn.JwtUtil;

@RestController

public class AuthenticationController {

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(HttpServletRequest request) {

String authHeader = request.getHeader("Authorization");

if (authHeader == null || !authHeader.startsWith("Basic ")) {

return ResponseEntity.*status*(401).body("Missing or invalid Authorization header");

}

// Decode Basic Auth

String base64Credentials = authHeader.substring("Basic ".length());

byte[] credDecoded = Base64.*getDecoder*().decode(base64Credentials);

String credentials = new String(credDecoded);

final String[] values = credentials.split(":", 2);

String username = values[0];

String password = values[1];

Authentication authentication = authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(username, password)

);

String token = jwtUtil.generateToken(authentication);

return ResponseEntity.*ok*().body("{\"token\":\"" + token + "\"}");

}

}

**AuthenticationResponse.java**

package com.cognizant.spring\_learn;

public class AuthenticationResponse {

private String token;

public AuthenticationResponse(String token) {

this.token = token;

}

public String getToken() {

return token;

}

}

**AuthenticationRequest.java**

package com.cognizant.spring\_learn;

public class AuthenticationRequest {

private String username;

private String password;

// Getters and Setters

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**SpringlearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringlearnApplication {

public static void main(String[] args) {

SpringApplication.*run*(SpringlearnApplication.class, args);

}

}

**SecurityConfig.java**

package com.cognizant.spring\_learn;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.core.userdetails.\*;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {

return config.getAuthenticationManager();

}

@Bean

public UserDetailsService userDetailsService() {

UserDetails user = User

.*withUsername*("user")

.password("{noop}pwd") // No password encoder for simplicity

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http.csrf(csrf -> csrf.disable())

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

)

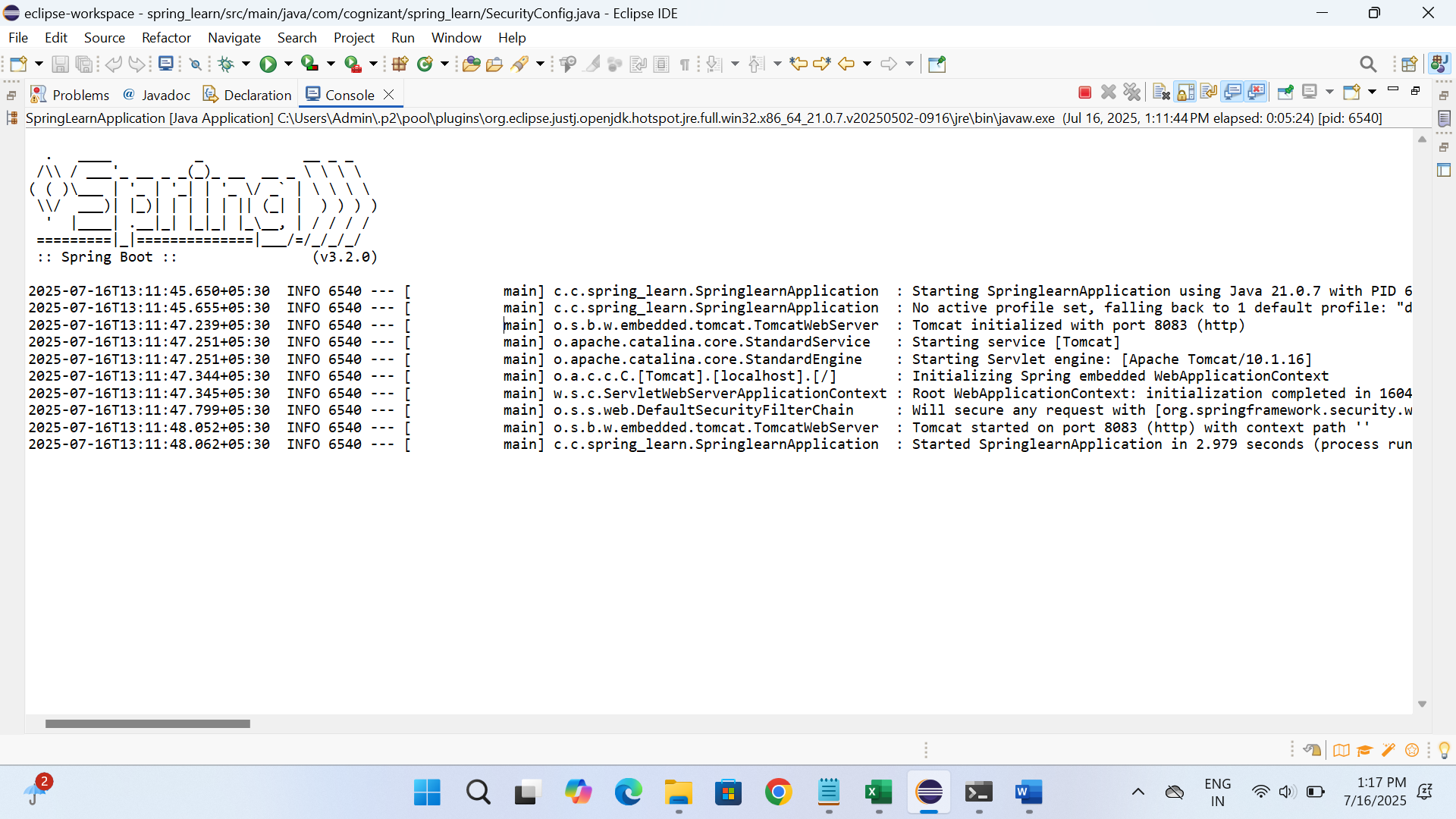
.httpBasic(); // Allows curl -u support

return http.build();

}

}

**Output**



* **Create authentication controller and configure it in SecurityConfig**

**AuthenticationController.java**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestHeader;

import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

private static final Logger *LOGGER* = LoggerFactory.*getLogger*(AuthenticationController.class);

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

*LOGGER*.info("START - /authenticate");

*LOGGER*.debug("Authorization Header: {}", authHeader);

Map<String, String> map = new HashMap<>();

map.put("token", "");

*LOGGER*.info("END - /authenticate");

return map;

}

}

**SecurityConfig.java**

package com.cognizant.spring\_learn;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.core.userdetails.UserDetailsService;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.httpBasic() // Enable basic authentication

.and()

.authorizeHttpRequests()

.requestMatchers("/countries").hasRole("USER")

.requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

.and()

.csrf().disable(); // Disable CSRF for testing with curl/Postman

return http.build();

}

@Bean

public UserDetailsService userDetailsService() {

UserDetails user = User.*withUsername*("user")

.password("{noop}pwd")

.roles("USER")

.build();

UserDetails admin = User.*withUsername*("admin")

.password("{noop}admin")

.roles("ADMIN")

.build();

return new InMemoryUserDetailsManager(user, admin);

}

}

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant.spring-learn</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-learn</name>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.0</version> <!-- Use latest stable if available -->

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<java.version>17</java.version> <!-- Use 11 or 17 -->

</properties>

<dependencies>

<!-- Spring Boot Web Starter -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Logging (optional, included by default) -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<!-- For Spring Boot App execution -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<!-- For testing (optional) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Spring Boot Maven Plugin -->

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<version>3.2.0</version>

</plugin>

</plugins>

</build>

</project>

**SpringlearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringlearnApplication {

public static void main(String[] args) {

SpringApplication.*run*(SpringlearnApplication.class, args);

}

}

**Output**

