**WEEK 4**

**Spring REST Handson**

**Exercise 1:  
Create Spring Web Project using Maven**  
Group: com.cognizant

- Artifact: spring-learn

- Dependencies: Spring Web, Spring Boot DevTools

- IDE: Eclipse

- Project built using `mvn clean package`

- Verified execution using logger in main method.  
  
Updated code in SpringApplicationLearn.java:

**package** com.cognizant.spring\_learn;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** SpringLearnApplication {

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

**public** **static** **void** main(String[] args) {

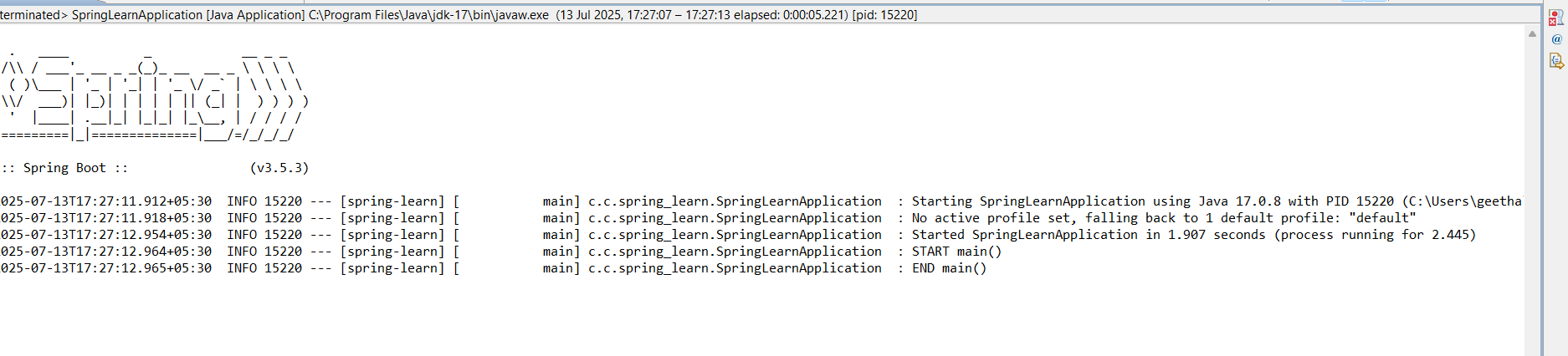
SpringApplication.*run*(SpringLearnApplication.**class**, args);

***LOGGER***.info("START main()");

***LOGGER***.info("END main()");

}

}

Output: 

**Exercise 2:  
Spring XML Configuration – Load Country Bean**

Folder Structure:  
spring-learn/

├── src/

│ ├── main/

│ │ ├── java/com/cognizant/spring\_learn/

│ │ │ ├── SpringLearnApplication.java

│ │ │ └── Country.java

│ │ └── resources/

│ │ ├── application.properties

│ │ └── country.xml

├── pom.xml  
  
country.xml:

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

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

xsi:schemaLocation="

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="country" class="com.cognizant.spring\_learn.Country">

<property name="code" value="IN" />

<property name="name" value="India" />

</bean>

</beans>  
  
Country.java:  
package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class Country {

private String code;

private String name;

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

public Country() {

LOGGER.debug("Inside Country Constructor.");

}

public String getCode() {

LOGGER.debug("Inside getCode()");

return code;

}

public void setCode(String code) {

LOGGER.debug("Inside setCode()");

this.code = code;

}

public String getName() {

LOGGER.debug("Inside getName()");

return name;

}

public void setName(String name) {

LOGGER.debug("Inside setName()");

this.name = name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}  
  
SpringLearnApplication.java:  
package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.support.ClassPathXmlApplicationContext;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("START main()");

displayCountry();

LOGGER.info("END main()");

}

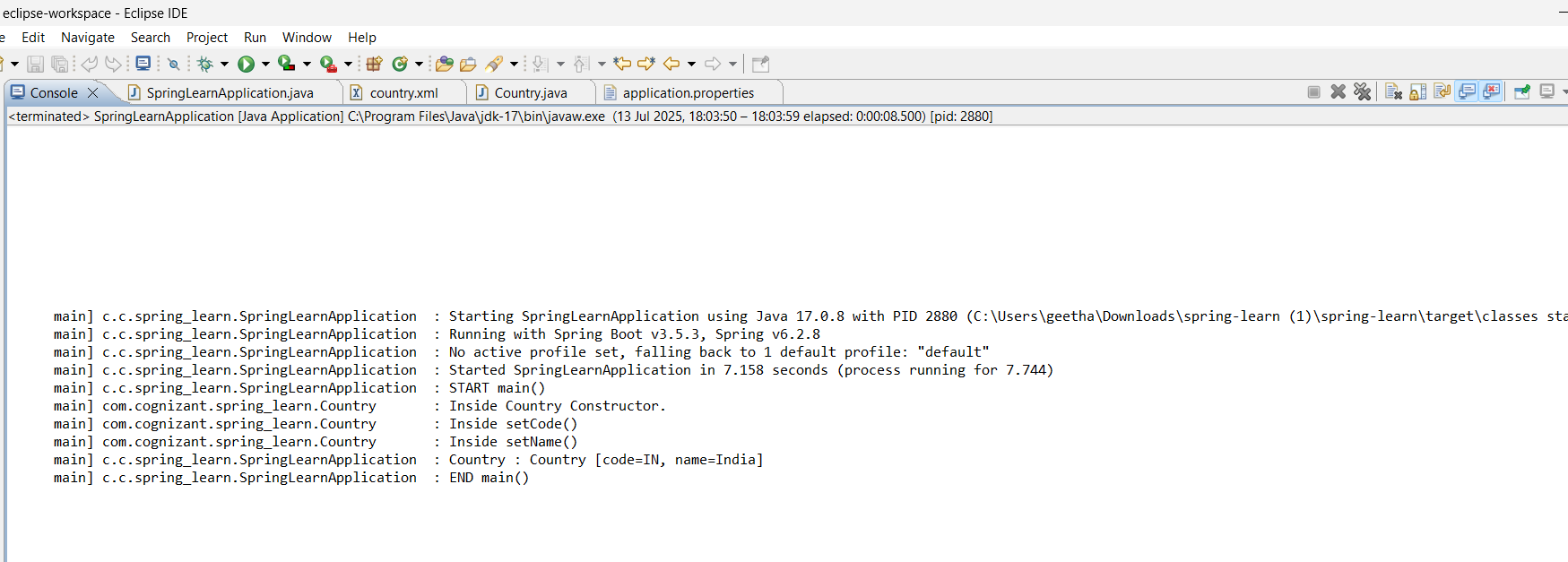
public static void displayCountry() {

ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

LOGGER.debug("Country : {}", context.getBean("country", Country.class).toString());

context.close();

}

}  
application.properties:  
logging.level.com.cognizant=DEBUG  
  
OUTPUT:  


**Exercise 3:  
Hello World RESTful Web Service**  
  
Package:com.cognizant.springlearn.controller

Class:HelloController.java

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class HelloController {

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

@GetMapping("/hello")

public String sayHello() {

LOGGER.info("Start: sayHello()");

String message = "Hello World!!";

LOGGER.info("End: sayHello()");

return message;

}

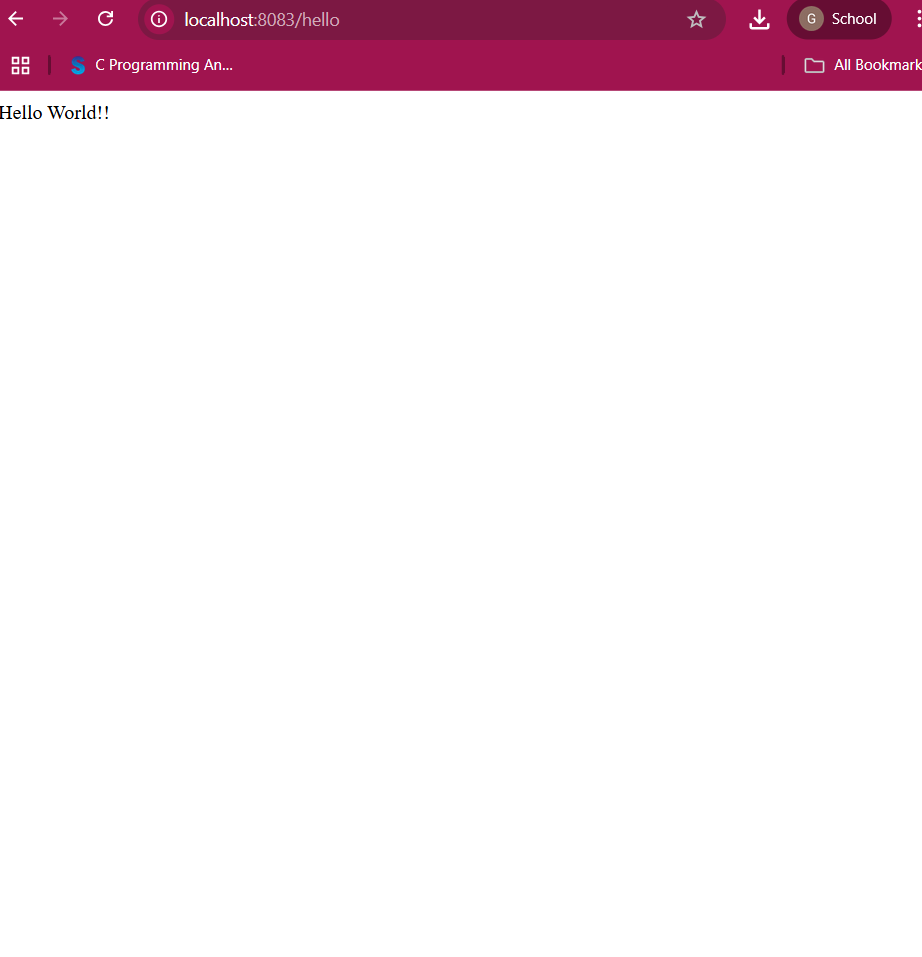
}

Application.properties:

spring.application.name=spring-learn

server.port=8083

**output:**



**Exercise 4:**

**Spring Core:XML Configuration with a Single Bean**

Model Class-Country.java:

package com.cognizant.springlearn.model;

public class Country {

private String code;

private String name;

// Getters and Setters

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

Country.xml:

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

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

xsi:schemaLocation="http://www.springframework.org/schema/beans

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="country" class="com.cognizant.springlearn.model.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

</beans>  
SpringLearnApplication.java:

package com.cognizant.springlearn;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.cognizant.springlearn.model.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class SpringLearnApplication {

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

public static void main(String[] args) {

LOGGER.info("START");

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

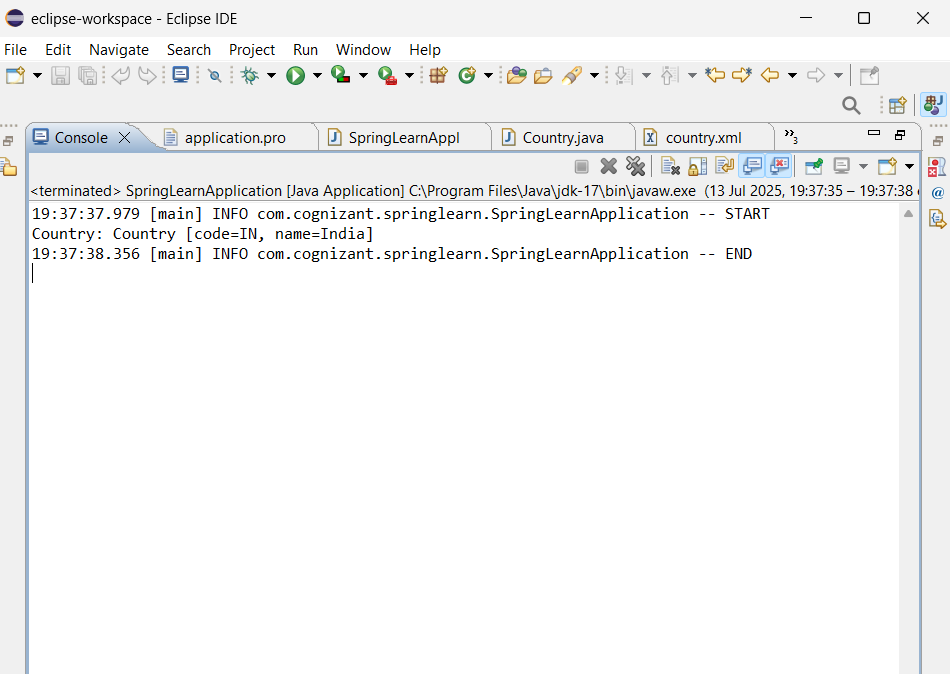
Country country = (Country) context.getBean("country");

System.out.println("Country: " + country.toString());

LOGGER.info("END");

}

}  
  
**OUTPUT:**



**Exercise 5:**

**Spring Core: XML Configuration with Collection**

Country.xml:

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

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

xsi:schemaLocation="http://www.springframework.org/schema/beans

https://www.springframework.org/schema/beans/spring-beans.xsd"

<!-- Single Country Bean -->

<bean id="country" class="com.cognizant.springlearn.model.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

<!-- List of Country Beans -->

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<bean class="com.cognizant.springlearn.model.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/>

</bean>

<bean class="com.cognizant.springlearn.model.Country">

<property name="code" value="US"/>

<property name="name" value="United States"/>

</bean>

<bean class="com.cognizant.springlearn.model.Country">

<property name="code" value="DE"/>

<property name="name" value="Germany"/>

</bean>

</list>

</constructor-arg>

</bean>

</beans>

SpringLearnApplication.java:

package com.cognizant.springlearn;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.cognizant.springlearn.model.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class SpringLearnApplication {

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

public static void main(String[] args) {

LOGGER.info("START");

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

Country country = (Country) context.getBean("country");

System.out.println("Single Country: " + country);

List<Country> countryList = (List<Country>) context.getBean("countryList");

System.out.println("Country List:");

for (Country c : countryList) {

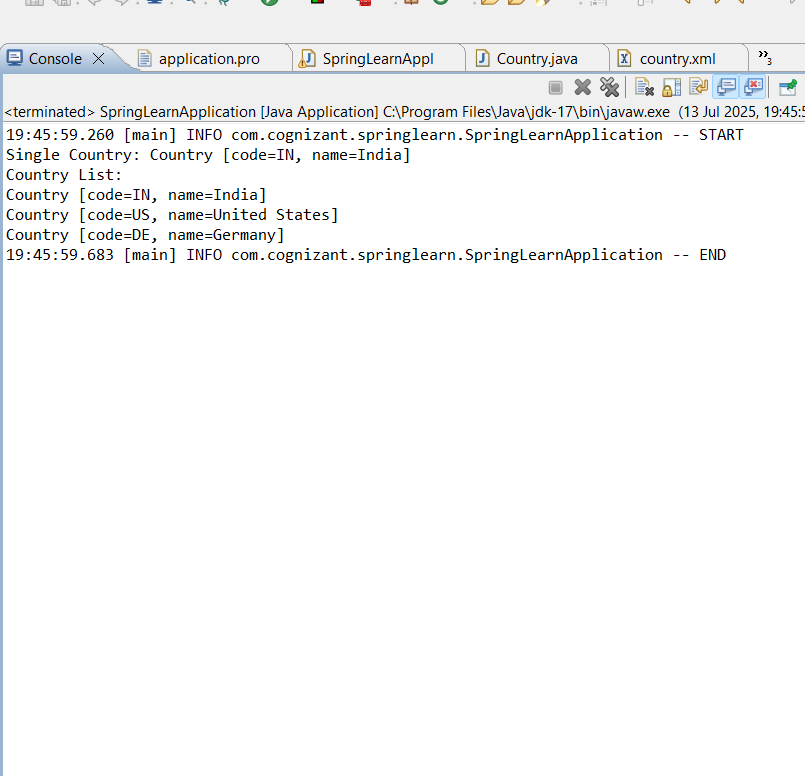
System.out.println(c);

}

LOGGER.info("END");

}

}

**OUTPUT:**  


**JWT Authentication using Spring Security**

spring-learn

│

├── src

│ └── main

│ └── java

│ └── com

│ └── cognizant

│ └── springlearn

│ ├── SpringLearnApplication.java

│ ├── controller

│ │ └── HelloController.java

│ ├── model

│ │ ├── AuthenticationRequest.java

│ │ └── AuthenticationResponse.java

│ ├── service

│ │ └── UserDetailsServiceImpl.java

│ └── security

│ ├── JwtUtil.java

│ ├── JwtRequestFilter.java

│ └── SecurityConfigurer.java

│

├── resources

│ ├── application.properties

│ └── static / templates (optional)

│

└── pom.xml

SpringLearnApplication.java:

package com.cognizant.springlearn;

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);

}

}

AuthenticationRequest.java:

package com.cognizant.springlearn.model;

public class AuthenticationRequest {

private String username;

private String password;

public AuthenticationRequest() {}

public AuthenticationRequest(String username, String password) {

this.username = username;

this.password = password;

// Getters & Setters

}

AuthenticationResponse.java:

package com.cognizant.springlearn.model;

public class AuthenticationResponse {

private String jwt;

public AuthenticationResponse(String jwt) {

this.jwt = jwt;

}

public String getJwt() {

return jwt;

}

}

UserDetailsServiceImpl.java:

package com.cognizant.springlearn.service;

import java.util.ArrayList;

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

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

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

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

import org.springframework.stereotype.Service;

@Service

public class UserDetailsServiceImpl implements UserDetailsService {

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

if ("admin".equals(username)) {

return new User("admin", "{noop}password", new ArrayList<>());

} else {

throw new UsernameNotFoundException("User not found with username: " + username);

}

}

}

JwtUtil.java:

package com.cognizant.springlearn.security;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.util.function.Function;

import io.jsonwebtoken.\*;

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

import org.springframework.stereotype.Service;

@Service

public class JwtUtil {

private String SECRET\_KEY = "secret";

public String extractUsername(String token) {

return extractClaim(token, Claims::getSubject);

}

public Date extractExpiration(String token) {

return extractClaim(token, Claims::getExpiration);

}

public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {

final Claims claims = extractAllClaims(token);

return claimsResolver.apply(claims);

}

private Claims extractAllClaims(String token) {

return Jwts.parser().setSigningKey(SECRET\_KEY).parseClaimsJws(token).getBody();

}

private Boolean isTokenExpired(String token) {

return extractExpiration(token).before(new Date());

}

public String generateToken(UserDetails userDetails) {

Map<String, Object> claims = new HashMap<>();

return createToken(claims, userDetails.getUsername());

}

private String createToken(Map<String, Object> claims, String subject) {

return Jwts.builder().setClaims(claims).setSubject(subject)

.setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 60 \* 10))

.signWith(SignatureAlgorithm.HS256, SECRET\_KEY).compact();

}

public Boolean validateToken(String token, UserDetails userDetails) {

final String username = extractUsername(token);

return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));

}

}

JwtRequestFilter.java:

package com.cognizant.springlearn.security;

import java.io.IOException;

import com.cognizant.springlearn.service.UserDetailsServiceImpl;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

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

import org.springframework.stereotype.Component;

import org.springframework.web.filter.OncePerRequestFilter;

@Component

public class JwtRequestFilter extends OncePerRequestFilter {

@Autowired

private UserDetailsServiceImpl userDetailsService;

@Autowired

private JwtUtil jwtUtil;

@Override

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain chain)

throws ServletException, IOException {

final String authorizationHeader = request.getHeader("Authorization");

String username = null;

String jwt = null;

if (authorizationHeader != null && authorizationHeader.startsWith("Bearer ")) {

jwt = authorizationHeader.substring(7);

username = jwtUtil.extractUsername(jwt);

}

if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {

UserDetails userDetails = this.userDetailsService.loadUserByUsername(username);

if (jwtUtil.validateToken(jwt, userDetails)) {

UsernamePasswordAuthenticationToken authToken =

new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());

authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(authToken);

}

}

chain.doFilter(request, response);

}

}

SecurityConfigurer.java:

package com.cognizant.springlearn.security;

import com.cognizant.springlearn.service.UserDetailsServiceImpl;

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

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.web.SecurityFilterChain;

import org.springframework.security.config.http.SessionCreationPolicy;

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

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration

public class SecurityConfigurer {

@Autowired

private JwtRequestFilter jwtRequestFilter;

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeHttpRequests()

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

.anyRequest().authenticated()

.and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);

http.addFilterBefore(jwtRequestFilter, UsernamePasswordAuthenticationFilter.class);

return http.build();

}

@Bean

public AuthenticationManager authenticationManager(AuthenticationConfiguration configuration) throws Exception {

return configuration.getAuthenticationManager();

}

}

HelloController.java:

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.AuthenticationRequest;

import com.cognizant.springlearn.model.AuthenticationResponse;

import com.cognizant.springlearn.security.JwtUtil;

import com.cognizant.springlearn.service.UserDetailsServiceImpl;

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

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

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

@RestController

public class HelloController {

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private UserDetailsServiceImpl userDetailsService;

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/hello")

public String hello() {

return "Hello, JWT-secured world!";

}

@PostMapping("/authenticate")

public AuthenticationResponse authenticate(@RequestBody AuthenticationRequest request) throws Exception {

authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(request.getUsername(), request.getPassword()));

final UserDetails userDetails = userDetailsService.loadUserByUsername(request.getUsername());

final String jwt = jwtUtil.generateToken(userDetails);

return new AuthenticationResponse(jwt);

}

}