**Week 4 - Spring Rest using Spring Boot SupersetId-6431499**

**Name : Balaji V**

**Spring Rest HandsOn**

**HandsOn 1-Create a Spring Web Project using Maven**

**Code:**

**pom.xml:**

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

<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringWebProject</name>

<description>SpringWebProject</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<excludes>

<exclude>

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

<artifactId>spring-boot-devtools</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

**SpringLearnApplication.java:**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@SpringBootApplication

public class SpringLearnApplication{

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

public static void main(String[] args){

logger.info("Starting Spring Boot Application...");

logger.info("Application Name: Spring Learn");

logger.info("Group: com.cognizant");

logger.info("Artifact: spring-learn");

SpringApplication.run(SpringLearnApplication.class, args);

logger.info("CTS-Spring Boot Application Started Successfully!!!");

logger.info("Application is running on: http://localhost:8080");

logger.info("Available endpoints:");

logger.info("- GET /hello - Simple greeting endpoint");

logger.info("- GET /actuator/health - Health check endpoint");

}

}

**HelloController.java:**

package com.cognizant.springlearn.controller;

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

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

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

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@RestController

public class HelloController{

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

@GetMapping("/hello")

public String hello(){

logger.info("Hello endpoint called");

return "Hello from Spring Boot Learning Project!";

}

@GetMapping("/hello/{name}")

public String helloWithName(@PathVariable String name){

logger.info("Hello endpoint called with name: {}", name);

return "Hello, " + name + "! Welcome to Spring Boot Learning Project!";

}

@GetMapping("/greet")

public String greet(@RequestParam(defaultValue = "World") String name){

logger.info("Greet endpoint called with name: {}", name);

return "Greetings, " + name + "! This is a Spring Boot application.";

}

}

**Application.properties:**

spring.application.name=spring-learn

server.port=8080

server.servlet.context-path=/api

logging.level.com.cognizant.springlearn=INFO

logging.level.org.springframework.web=DEBUG

logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} - %msg%n

spring.devtools.restart.enabled=true

spring.devtools.livereload.enabled=true

spring.banner.location=classpath:banner.txt

**SpringLearnApplicationTests.java:**

package com.cognizant.springlearn;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.junit.jupiter.SpringJUnitConfig;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@SpringBootTest

@SpringJUnitConfig

class SpringLearnApplicationTests{

private static final Logger logger=LoggerFactory.getLogger(SpringLearnApplicationTests.class);

@Test

void contextLoads(){

logger.info("Testing application context loading...");

logger.info("Application context loaded successfully!!!");

}

@Test

void applicationStartsSuccessfully(){

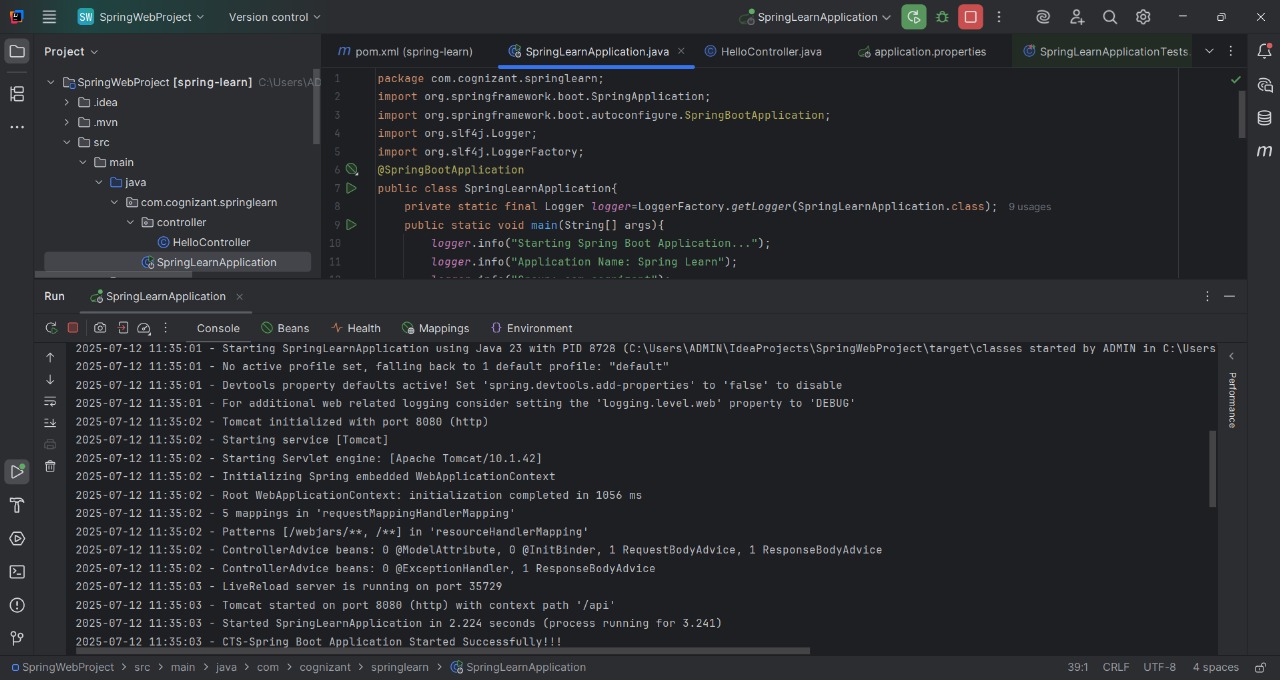
logger.info("Testing application startup...");

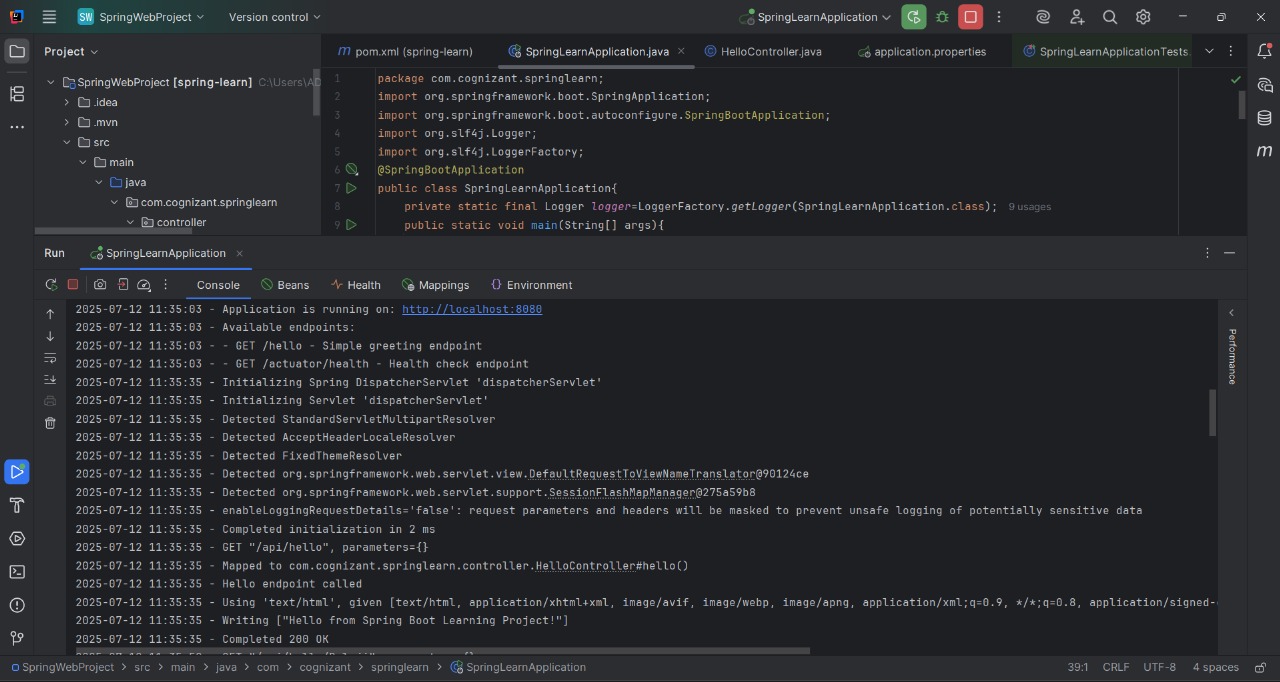
logger.info("Application started successfully!!!");

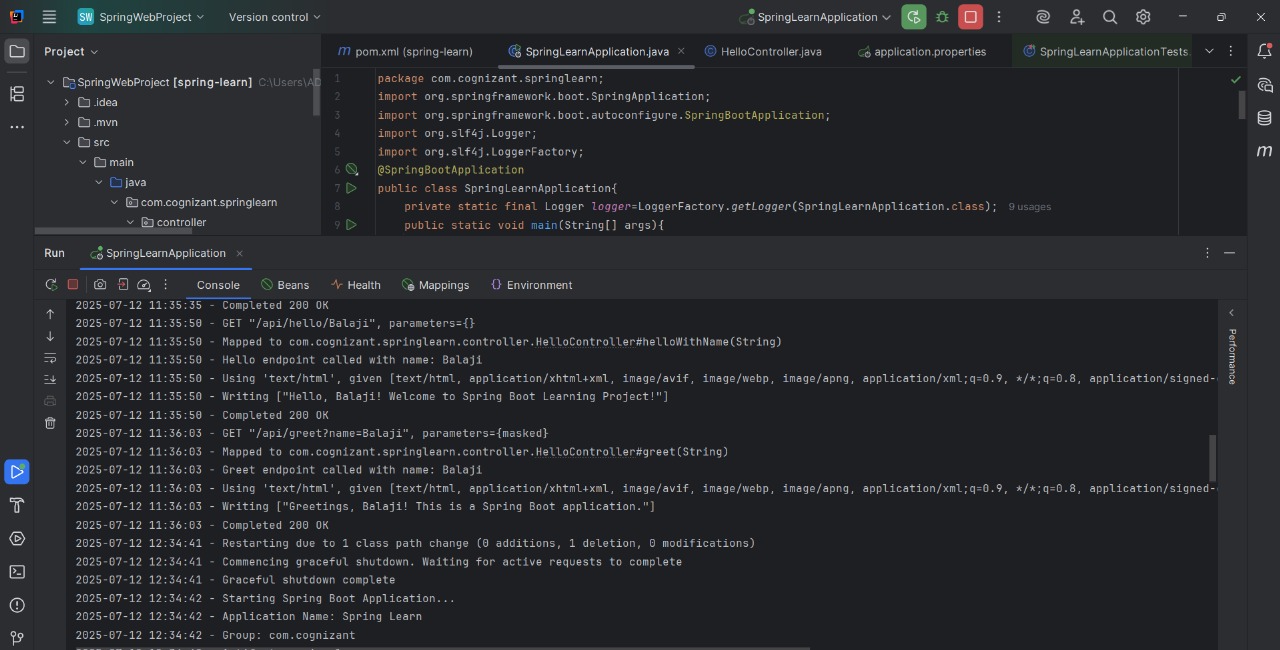
}

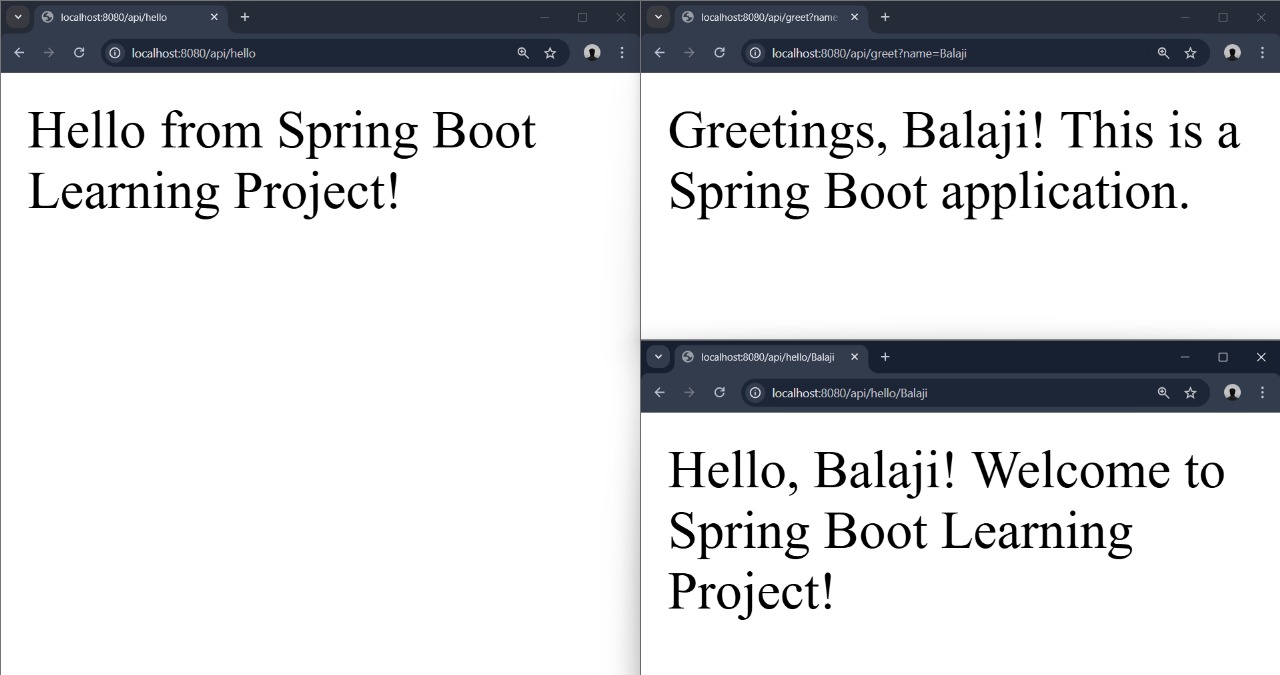
}

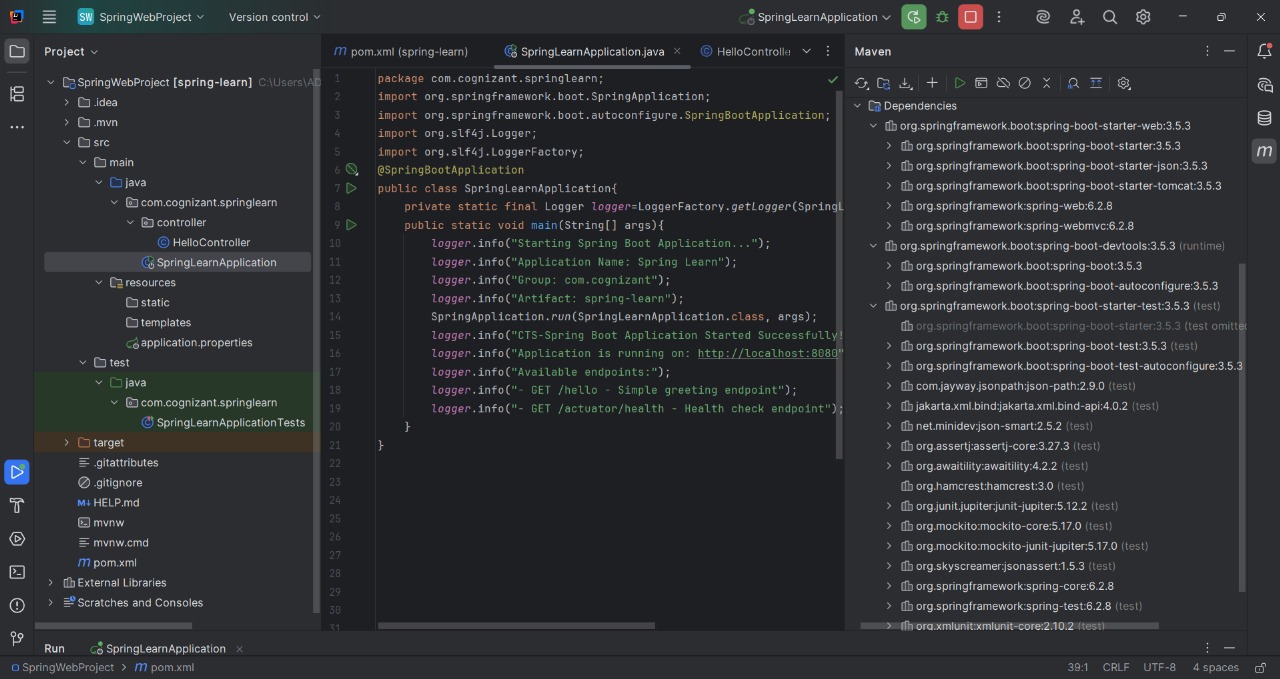
**Output:**











**HandsOn 4**

**Spring Core – Load Country from Spring Configuration XML**

**Code:**

**pom.xml:**

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

<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>SpringCore-Project</name>

<description>SpringCore-Project</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-boot-configuration-processor</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jcl</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**Country.java:**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class Country{

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

private String code;

private String name;

public Country(){

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

}

public String getCode(){

LOGGER.debug("Inside getCode() method. Returning code: {}", code);

return code;

}

public void setCode(String code){

LOGGER.debug("Inside setCode() method. Setting code to: {}", code);

this.code = code;

}

public String getName(){

LOGGER.debug("Inside getName() method. Returning name: {}", name);

return name;

}

public void setName(String name){

LOGGER.debug("Inside setName() method. Setting name to: {}", name);

this.name = name;

}

@Override

public String toString(){

LOGGER.debug("Inside toString() method.");

return "Country{" + "code='" + code + '\'' + ", name='" + name + '\'' + '}';

}

}

**SpringLearnApplication.java:**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication{

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

public static void main(String[] args){

SpringLearnApplication app=new SpringLearnApplication();

app.displayCountry();

}

@SuppressWarnings({"RedundantCast"})

public void displayCountry(){

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

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

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

}

}

**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.springlearn.Country">

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

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

</bean>

</beans>

**application.properties:**

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

<configuration>

<appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<appender name="FILE" class="ch.qos.logback.core.FileAppender">

<file>application.log</file>

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<root level="DEBUG">

<appender-ref ref="CONSOLE"/>

<appender-ref ref="FILE"/>

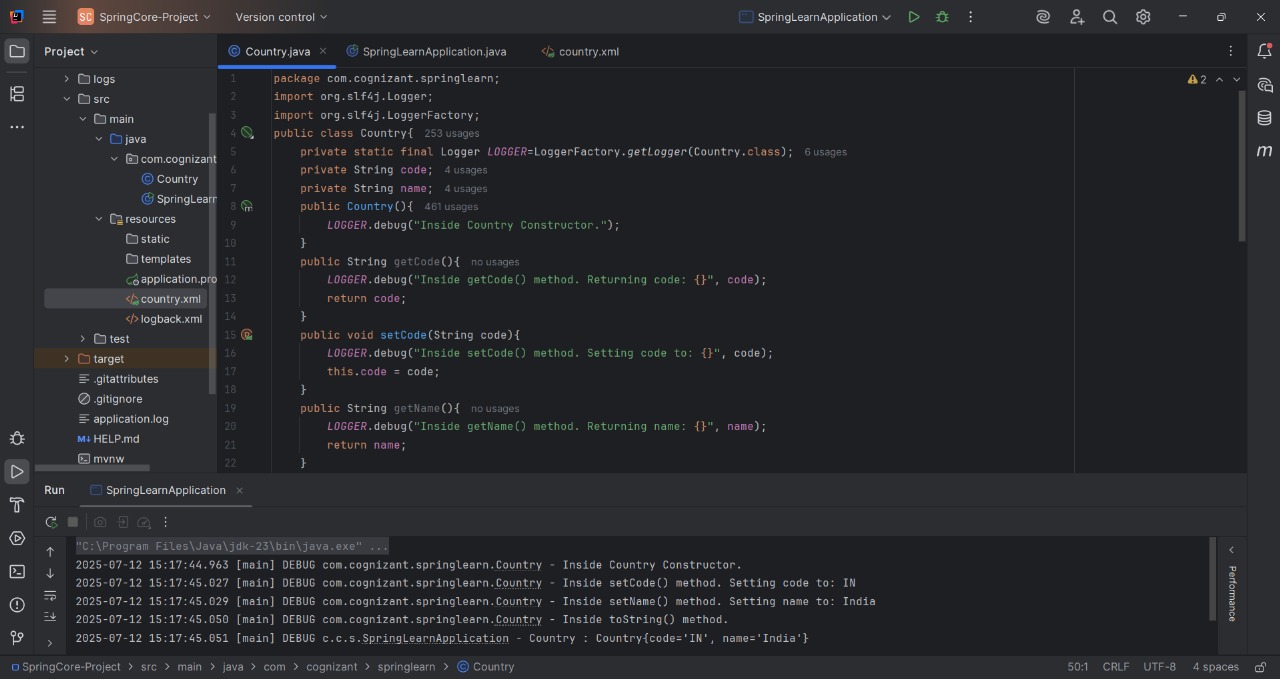
</root>

<logger name="org.springframework" level="INFO"/>

<logger name="com.cognizant.springlearn" level="DEBUG"/>

</configuration>

**Output:**



**Hello World RESTful Web Service**

**Code:**

**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() method called");

String message="Hello World!!!-Balaji-CTS-6431499";

logger.info("End - sayHello() method execution completed");

return message;

}

}

**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);

}

}

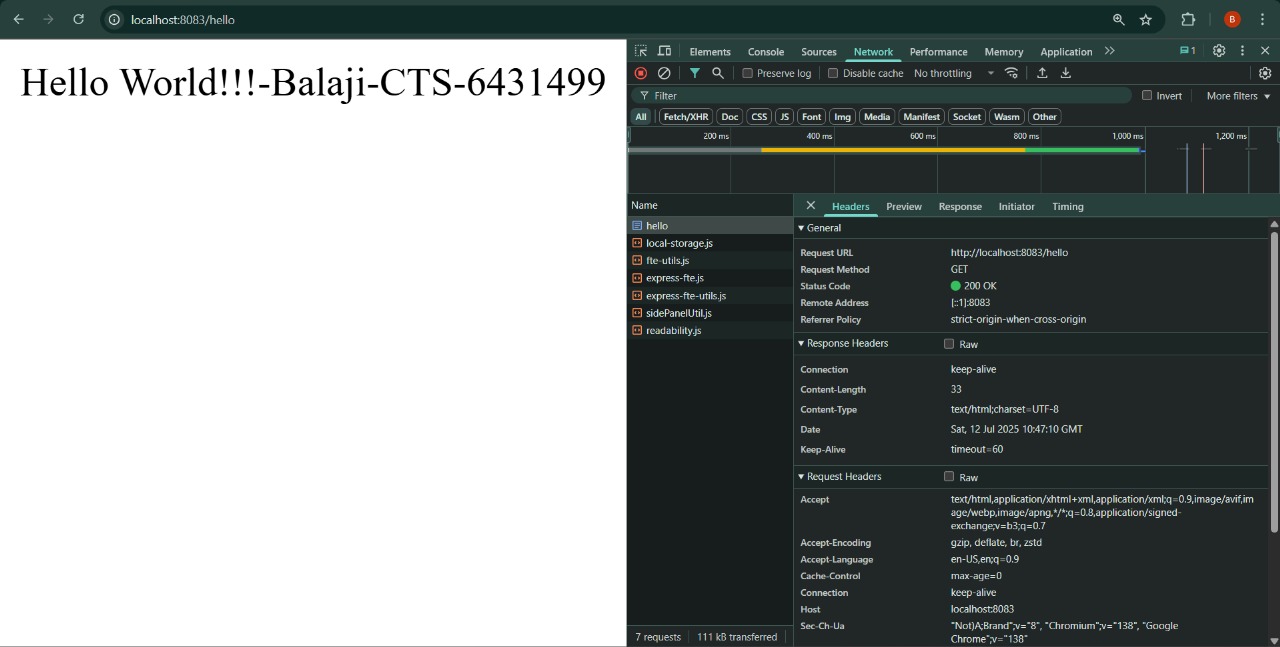
**application.properties:**

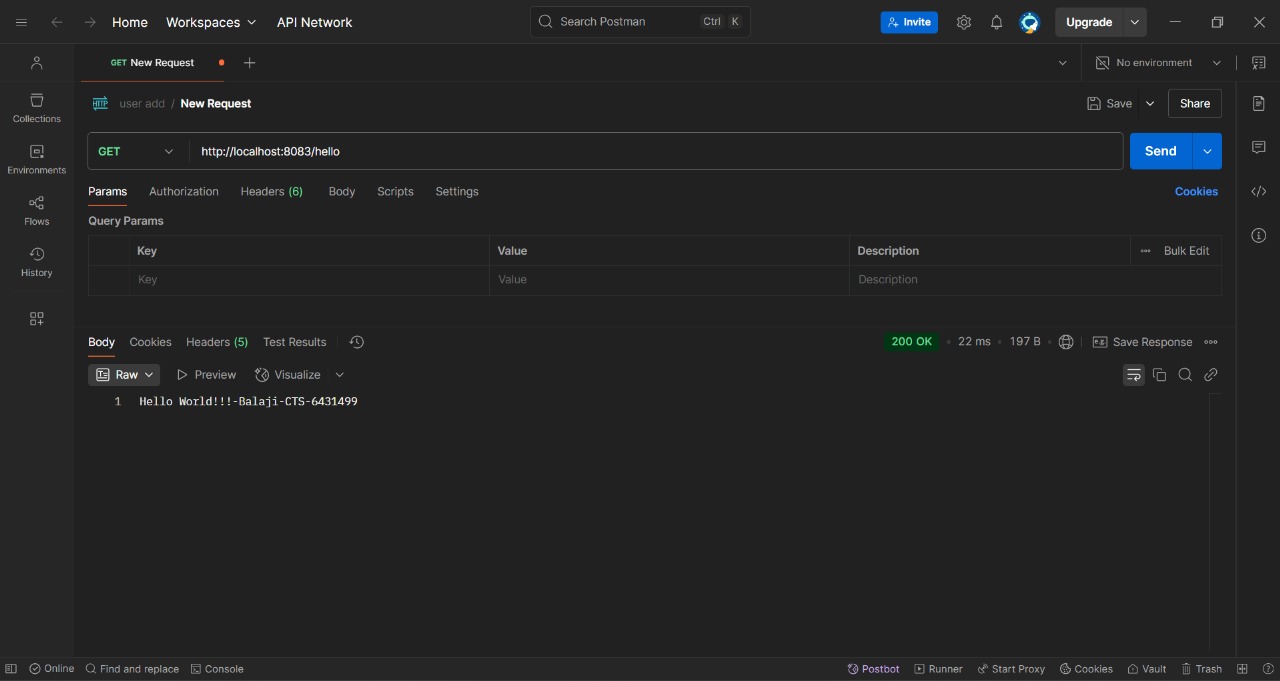
server.port=8083

logging.level.com.cognizant.springlearn=INFO

logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} - %msg%n

**Output:**







**REST - Country Web Service**

**Code:**

**CountryController.java:**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

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

@RestController

public class CountryController{

private ApplicationContext context;

public CountryController(){

this.context = new ClassPathXmlApplicationContext("spring-config.xml");

}

@RequestMapping("/country")

public Country getCountryIndia() {

// Load India bean from spring xml configuration

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

return indiaCountry;

}

}

**Country.java:**

package com.cognizant.springlearn.model;

public class Country{

private String code;

private String name;

public Country(){}

public Country(String code,String name){

this.code=code;

this.name=name;

}

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 + '\'' + '}';

}

}

**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);

}

}

**application.properties:**

server.port=8083

server.servlet.context-path=/

spring.application.name=spring-learn

logging.level.com.cognizant.springlearn=DEBUG

logging.level.root=INFO

**spring-config.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="indiaCountry" class="com.cognizant.springlearn.model.Country">

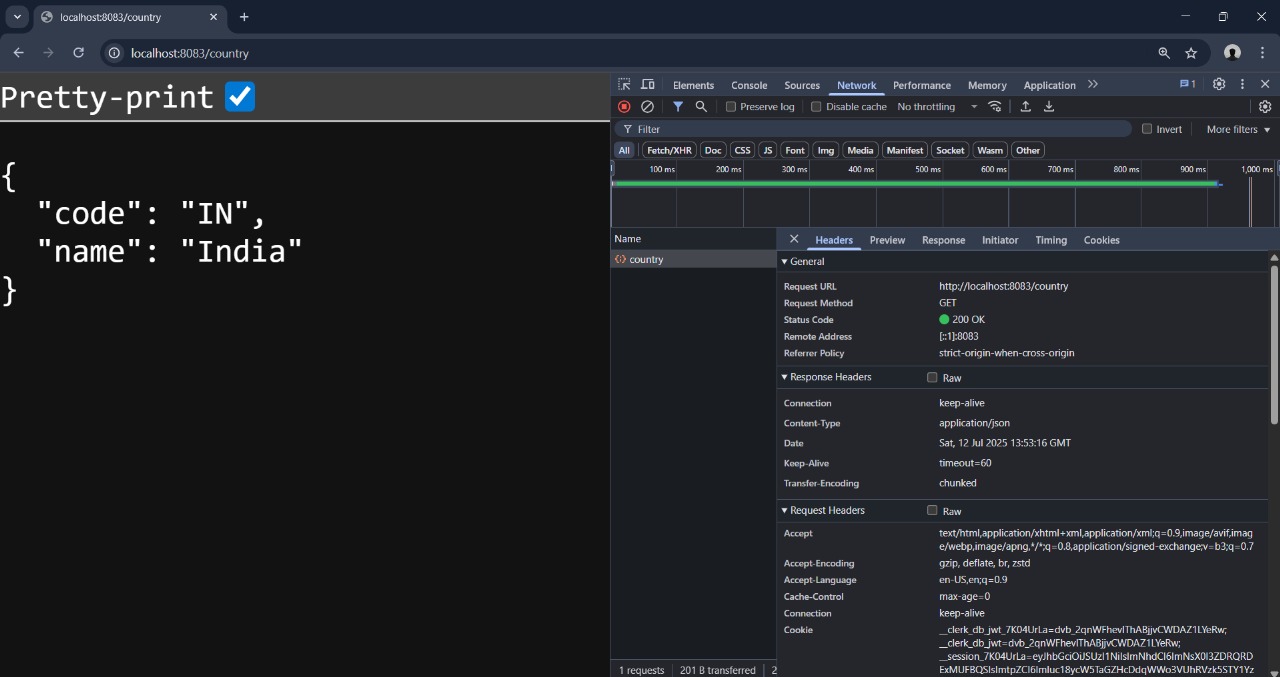
<property name="code" value="IN"/>

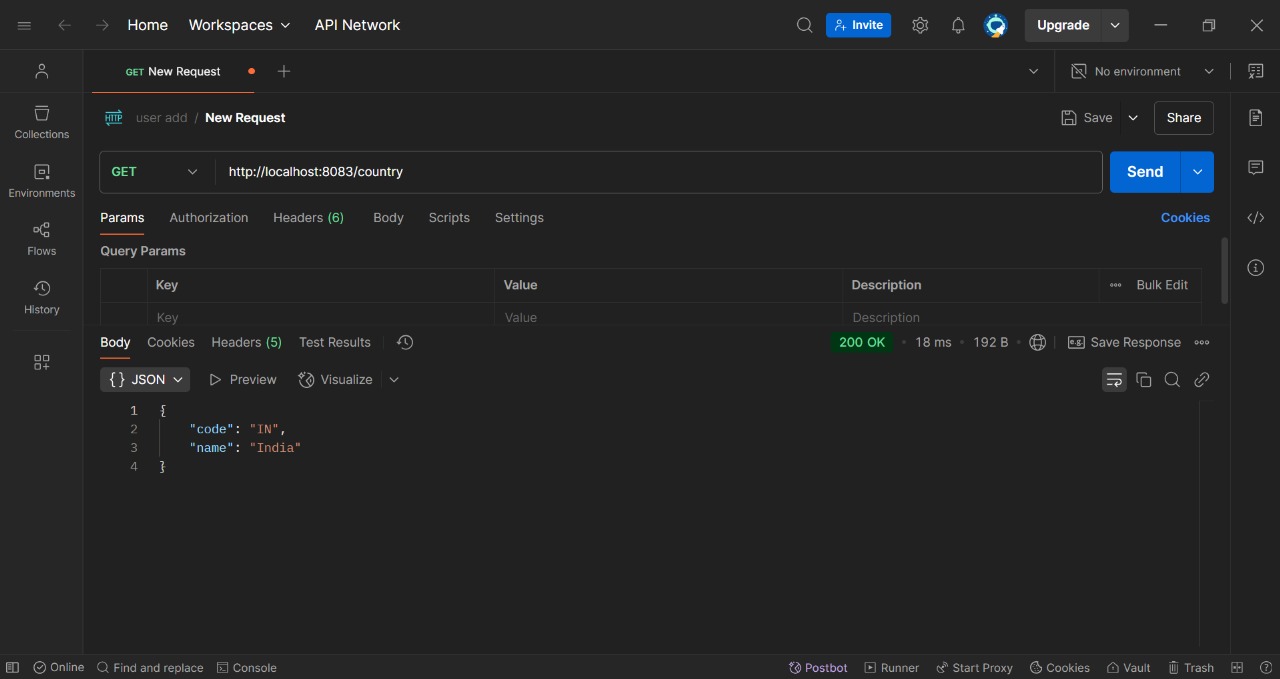
<property name="name" value="India"/>

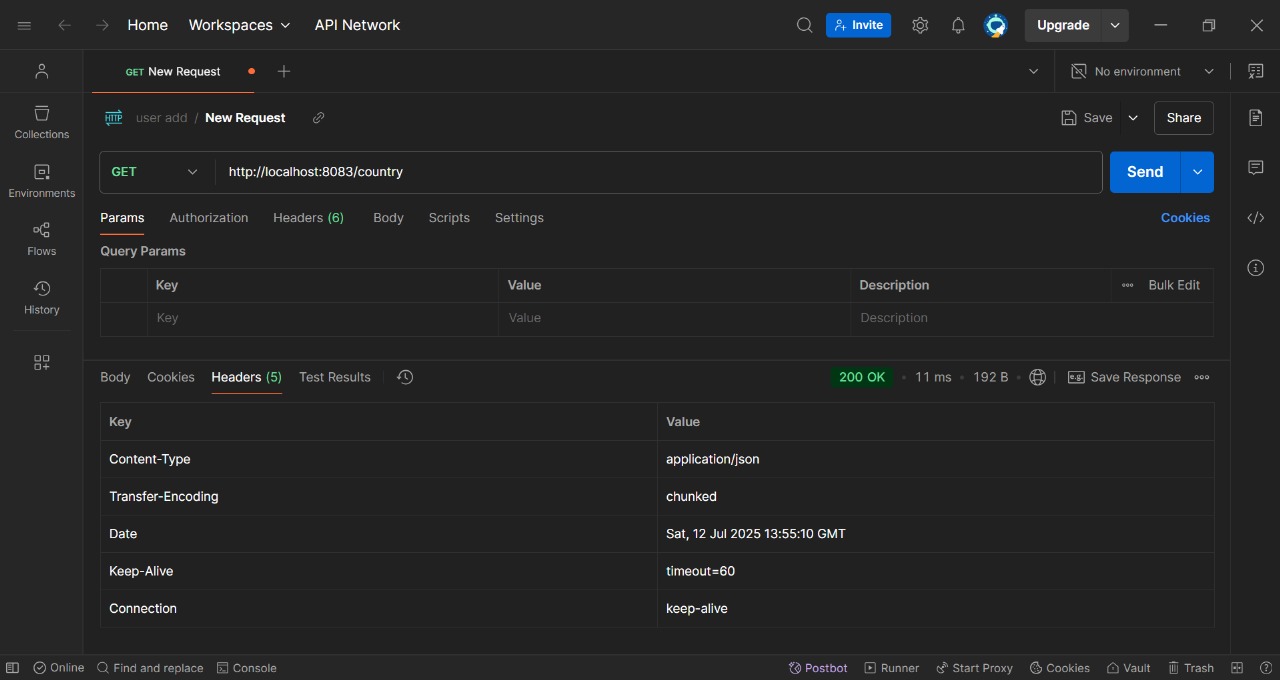
</bean>

</beans>

**Output:**







**REST - Get country based on country code**

**Code:**

**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);

}

}

**CountryController.java:**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

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

import org.springframework.http.ResponseEntity;

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

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

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

@RestController

public class CountryController{

@Autowired

private CountryService countryService;

@GetMapping("/countries/{code}")

public ResponseEntity<Country> getCountry(@PathVariable String code){

Country country=countryService.getCountry(code);

if(country!=null){

return ResponseEntity.ok(country);

}else{

return ResponseEntity.notFound().build();

}

}

}

**Country.java:**

package com.cognizant.springlearn.model;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name="country")

public class Country{

private String code;

private String name;

public Country(){}

public Country(String code,String name){

this.code=code;

this.name=name;

}

@XmlElement

public String getCode(){

return code;

}

public void setCode(String code){

this.code=code;

}

@XmlElement

public String getName(){

return name;

}

public void setName(String name){

this.name=name;

}

}

**CountryList.java:**

package com.cognizant.springlearn.model;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlRootElement;

import java.util.List;

@XmlRootElement(name="countries")

public class CountryList{

private List<Country> countries;

@XmlElement(name="country")

public List<Country> getCountries(){

return countries;

}

public void setCountries(List<Country> countries){

this.countries=countries;

}

}

**CountryService.java:**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.model.CountryList;

import org.springframework.core.io.ClassPathResource;

import org.springframework.stereotype.Service;

import javax.xml.bind.JAXBContext;

import javax.xml.bind.JAXBException;

import javax.xml.bind.Unmarshaller;

import java.io.IOException;

import java.util.List;

import java.util.Optional;

@Service

public class CountryService{

private List<Country> countries;

public CountryService(){

loadCountries();

}

private void loadCountries(){

try{

JAXBContext jaxbContext=JAXBContext.newInstance(CountryList.class);

Unmarshaller unmarshaller=jaxbContext.createUnmarshaller();

ClassPathResource resource=new ClassPathResource("country.xml");

CountryList countryList=(CountryList) unmarshaller.unmarshal(resource.getInputStream());

this.countries=countryList.getCountries();

}catch(JAXBException | IOException e){

throw new RuntimeException("Failed to load countries from XML", e);

}

}

public Country getCountry(String code){

Optional<Country> country=countries.stream()

.filter(c -> c.getCode().equalsIgnoreCase(code))

.findFirst();

return country.orElse(null);

}

public Country getCountryWithIteration(String code){

for(Country country : countries){

if (country.getCode().equalsIgnoreCase(code)){

return country;

}

}

return null;

}

}

**application.properties:**

server.port=8083

server.servlet.context-path=/

spring.application.name=spring-learn

logging.level.com.cognizant.springlearn=DEBUG

logging.level.root=INFO

**country.xml:**

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

<countries>

<country>

<code>IN</code>

<name>India</name>

</country>

<country>

<code>US</code>

<name>United States</name>

</country>

<country>

<code>UK</code>

<name>United Kingdom</name>

</country>

<country>

<code>CA</code>

<name>Canada</name>

</country>

<country>

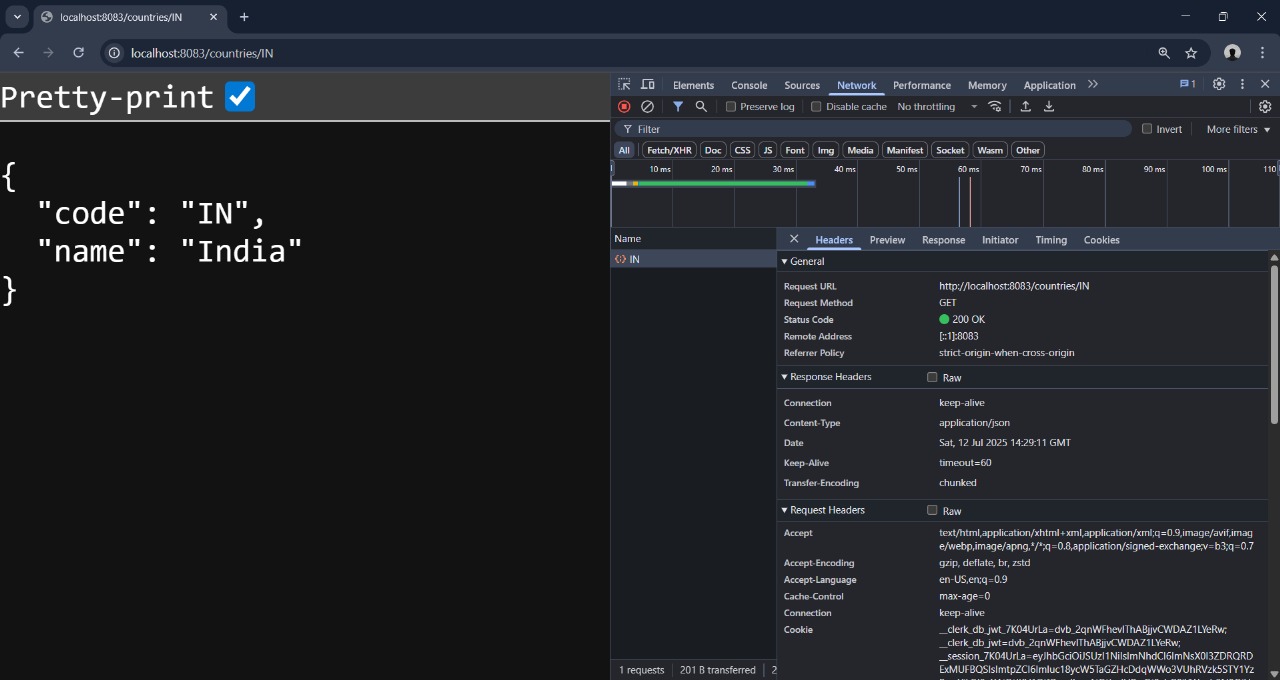
<code>AU</code>

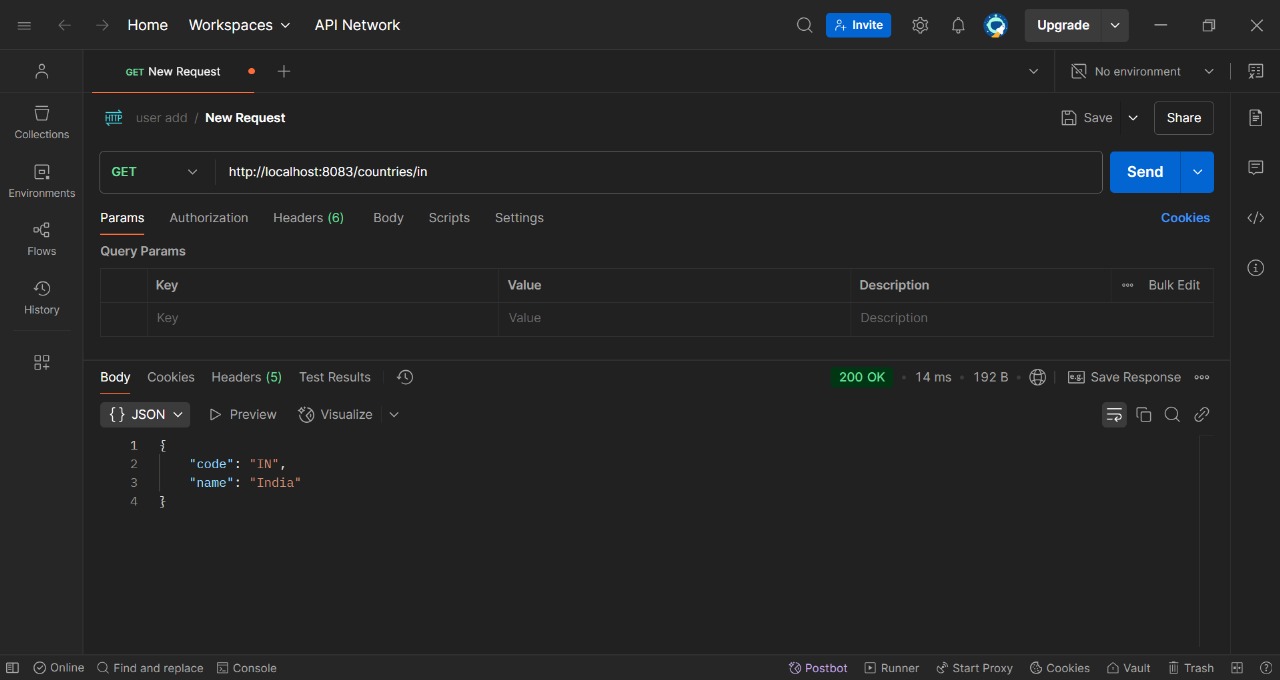
<name>Australia</name>

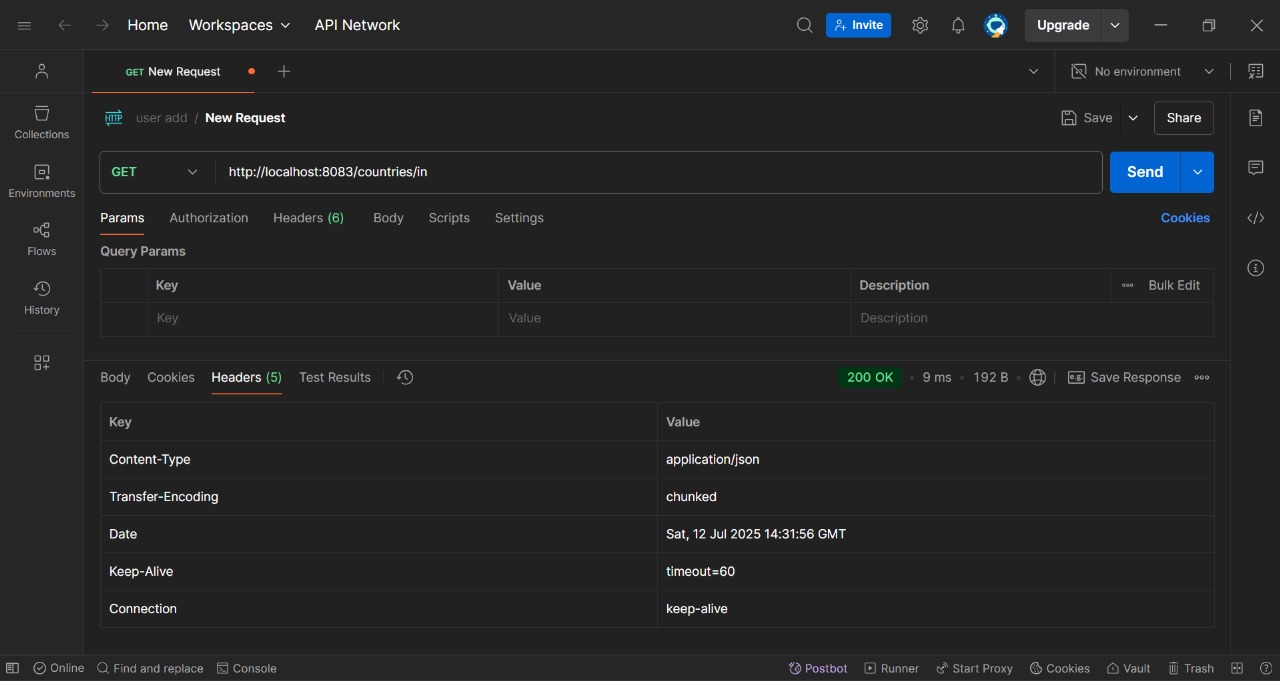
</country>

</countries>

**Output:**







**JWT HandsOn  
Create authentication service that returns JWT**

**Code:**

**SecurityConfig.java:**

package com.example.jwtauth.config;

import org.springframework.context.annotation.Bean;

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.config.http.SessionCreationPolicy;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

public class SecurityConfig{

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception{

http

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

.sessionManagement(session ->

session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

);

return http.build();

}

}

**AuthController.java:**

package com.example.jwtauth.controller;

import com.example.jwtauth.dto.AuthResponse;

import com.example.jwtauth.service.AuthService;

import com.example.jwtauth.service.JwtService;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

@RestController

@RequestMapping("/")

public class AuthController{

@Autowired

private AuthService authService;

@Autowired

private JwtService jwtService;

@PostMapping("/authenticate")

public ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authHeader){

try{

String[] credentials=authService.extractCredentials(authHeader);

String username=credentials[0];

String password=credentials[1];

if(!authService.validateCredentials(username,password)){

return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Invalid credentials");

}

String token=jwtService.generateToken(username);

return ResponseEntity.ok(new AuthResponse(token));

}catch(Exception e){

return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Authentication failed: " + e.getMessage());

}

}

}

**AuthResponse.java:**

package com.example.jwtauth.dto;

import com.fasterxml.jackson.annotation.JsonProperty;

public class AuthResponse{

@JsonProperty("token")

private String token;

public AuthResponse(){}

public AuthResponse(String token){

this.token=token;

}

public String getToken(){

return token;

}

public void setToken(String token){

this.token=token;

}

}

**AuthService.java:**

package com.example.jwtauth.service;

import org.springframework.stereotype.Service;

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

@Service

public class AuthService{

private final Map<String, String> users=new HashMap<>();

public AuthService(){

users.put("user", "pwd");

users.put("balaji", "cts6431499");

users.put("admin", "admin123");

}

public String[] extractCredentials(String authHeader){

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

throw new IllegalArgumentException("Invalid authorization header");

}

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

String credentials=new String(Base64.getDecoder().decode(base64Credentials));

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

if(parts.length!=2){

throw new IllegalArgumentException("Invalid credentials format");

}

return parts;

}

public boolean validateCredentials(String username,String password){

return username!=null && !username.trim().isEmpty() && password!=null && !password.trim().isEmpty();

}

}

**JwtService.java:**

package com.example.jwtauth.service;

import com.example.jwtauth.util.JwtUtil;

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

import org.springframework.stereotype.Service;

@Service

public class JwtService{

@Autowired

private JwtUtil jwtUtil;

public String generateToken(String username){

return jwtUtil.generateToken(username);

}

public boolean validateToken(String token,String username){

return jwtUtil.validateToken(token, username);

}

}

**JwtUtil.java:**

package com.example.jwtauth.util;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

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

import org.springframework.stereotype.Component;

import javax.crypto.SecretKey;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import java.util.function.Function;

@Component

public class JwtUtil{

@Value("${jwt.secret}")

private String secret;

@Value("${jwt.expiration}")

private Long expiration;

private SecretKey getSigningKey(){

return Keys.hmacShaKeyFor(secret.getBytes());

}

public String generateToken(String username){

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

return createToken(claims,username);

}

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

Date now=new Date();

Date expiryDate=new Date(now.getTime() + expiration);

return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(now).setExpiration(expiryDate).signWith(getSigningKey(), SignatureAlgorithm.HS256).compact();

}

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.parserBuilder().setSigningKey(getSigningKey()).build().parseClaimsJws(token).getBody();

}

public Boolean isTokenExpired(String token){

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

}

public Boolean validateToken(String token,String username){

final String extractedUsername=extractUsername(token);

return (extractedUsername.equals(username) && !isTokenExpired(token));

}

}

**JwtAuthApplication.java:**

package com.example.jwtauth;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JwtAuthApplication{

public static void main(String[] args){

SpringApplication.run(JwtAuthApplication.class, args);

}

}

**application.properties:**

server.port=8090

jwt.secret=mySecretKey123456789012345678901234567890

jwt.expiration=1200000

logging.level.org.springframework.security=DEBUG

**pom.xml:**

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

<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.5.3</version>

<relativePath/>

</parent>

<groupId>com.example</groupId>

<artifactId>jwt-auth</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>AuthenticationService</name>

<description>AuthenticationService</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**Output:**

