|  |  |
| --- | --- |
| **WEEK 4-Spring Rest HandsOn** | **Superset Id : 6429486**  **Name : Akshaya V** |

**Spring Rest HandsOn**

**Hands on 1-Create a Spring Web Project using Maven**

**Code:**

**SpringProjectApplication.java:**

package com.cognizant.springlearn;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
@SpringBootApplication  
public class SpringProjectApplication{  
 private static final Logger *LOGGER*=LoggerFactory.*getLogger*(SpringProjectApplication.class);  
 public static void main(String[] args){  
 *LOGGER*.info("Starting Spring Boot Application");  
 *LOGGER*.info("Application Name: SpringProject");  
 *LOGGER*.info("Group: com.cognizant");  
 *LOGGER*.info("Artifact: spring-learn");  
 SpringApplication.*run*(SpringProjectApplication.class,args);  
 }  
}

**application.properties:**

spring.application.name=SpringProject  
server.port=0  
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

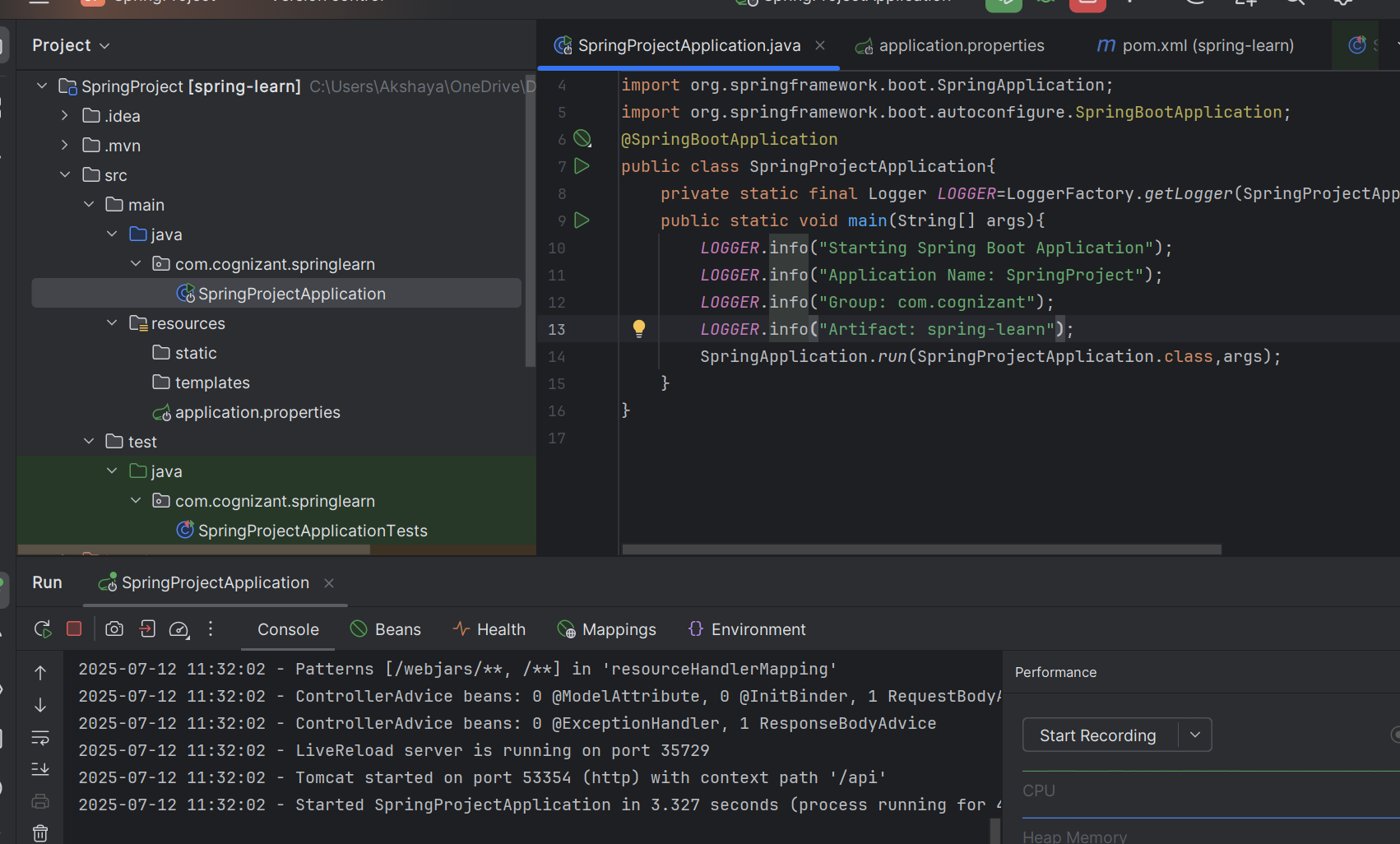
**SpringProjectApplicationTests.java:**

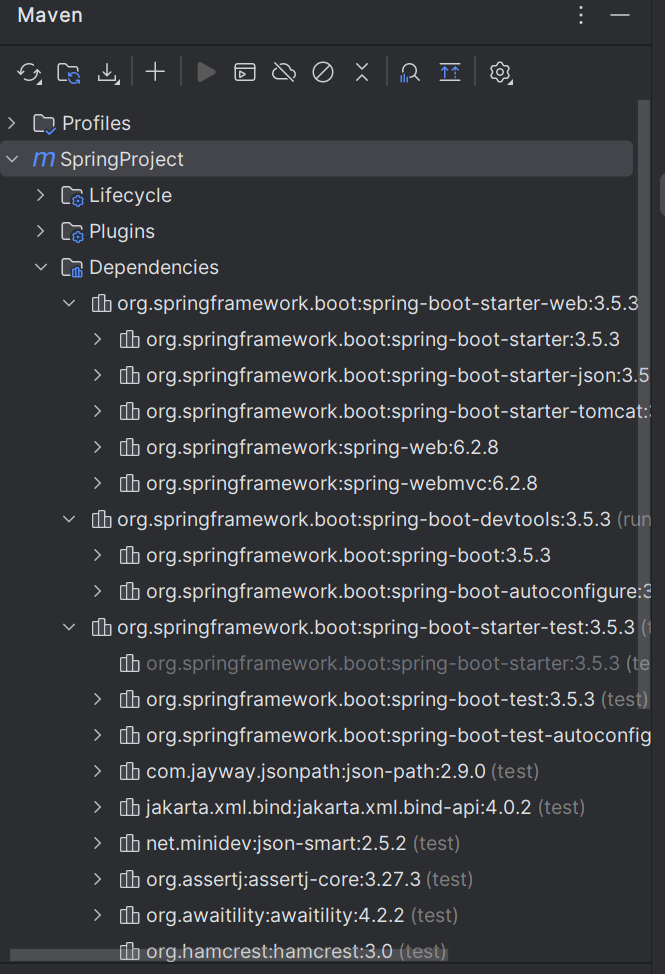
package com.cognizant.springlearn;  
import org.junit.jupiter.api.Test;  
import org.springframework.boot.test.context.SpringBootTest;  
import org.springframework.test.context.junit.jupiter.SpringJUnitConfig;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
@SpringBootTest  
@SpringJUnitConfig  
class SpringProjectApplicationTests{  
 private static final Logger *logger*=LoggerFactory.*getLogger*(SpringProjectApplicationTests.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!");  
 }  
}

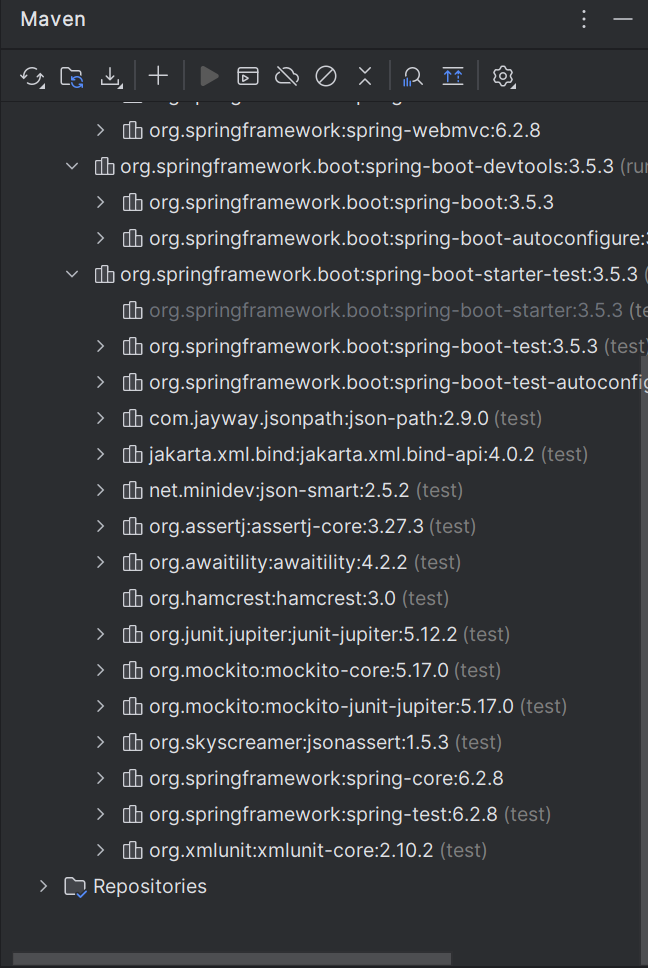
**pom.xml:**

<?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/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.cognizant</groupId>  
 <artifactId>spring-learn</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>SpringProject</name>  
 <description>SpringProject</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>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**Output:**







**Hands on 4**

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

**Code:**

**Country.java:**

package com.cognizant.springcountry;  
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()");  
 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 + '\'' + '}';  
 }  
}

**SpringCountryApplication.java:**

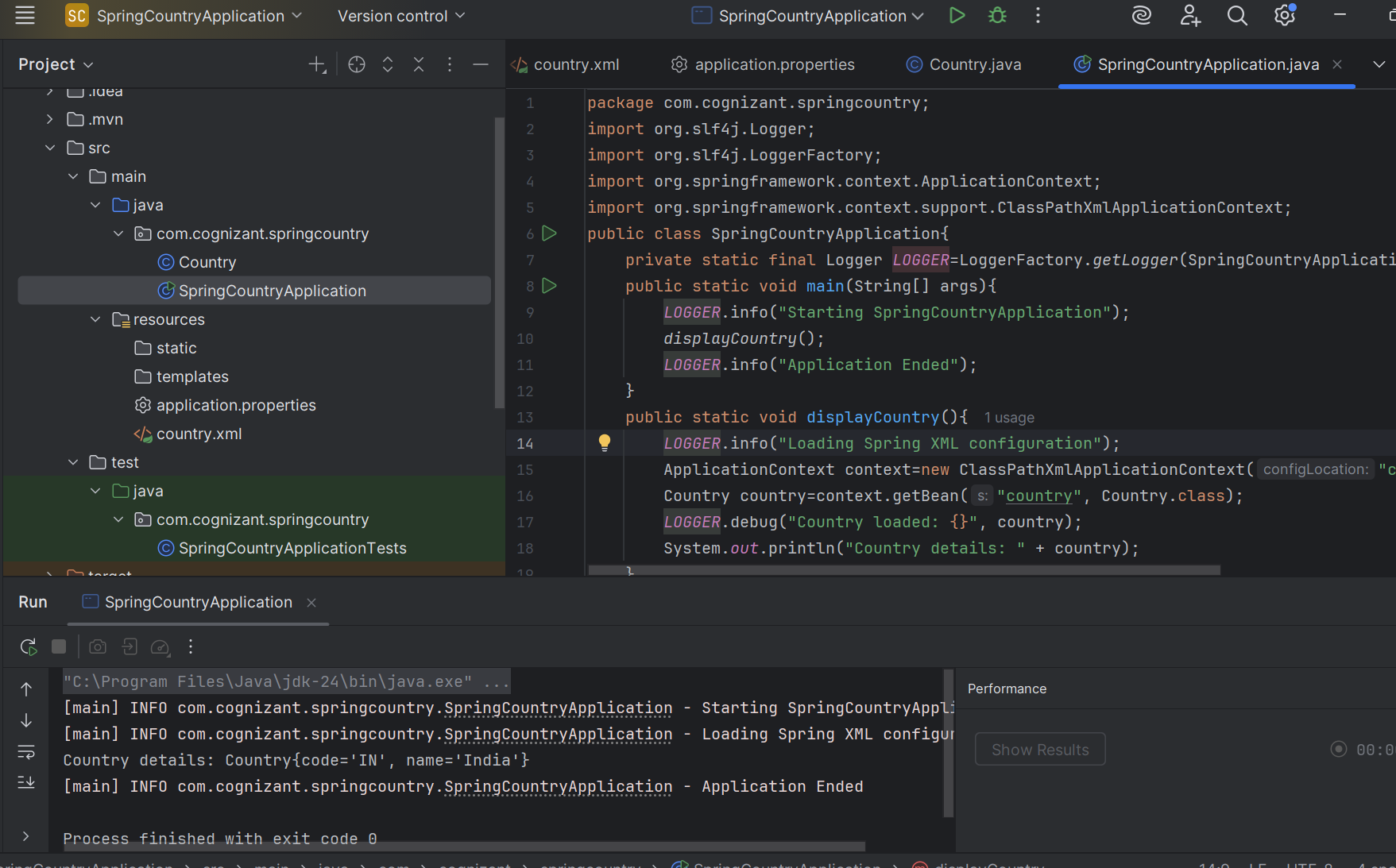
package com.cognizant.springcountry;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
public class SpringCountryApplication{  
 private static final Logger *LOGGER*=LoggerFactory.*getLogger*(SpringCountryApplication.class);  
 public static void main(String[] args){  
 *LOGGER*.info("Starting SpringCountryApplication");  
 *displayCountry*();  
 *LOGGER*.info("Application Ended");  
 }  
 public static void displayCountry(){  
 *LOGGER*.info("Loading Spring XML configuration");  
 ApplicationContext context=new ClassPathXmlApplicationContext("country.xml");  
 Country country=context.getBean("country", Country.class);  
 *LOGGER*.debug("Country loaded: {}", country);  
 System.*out*.println("Country details: " + country);  
 }  
}

**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.springcountry.Country">  
 <property name="code" value="IN" />  
 <property name="name" value="India" />  
 </bean>  
</beans>

**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>  
 <groupId>com.cognizant</groupId>  
 <artifactId>spring-country</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>SpringCountryApplication</name>  
 <description>SpringCountryApplication</description>  
 <properties>  
 <java.version>17</java.version>  
 <spring.version>5.3.34</spring.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-beans</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-simple</artifactId>  
 <version>2.0.13</version>  
 </dependency>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.10.1</version>  
 <configuration>  
 <source>${java.version}</source>  
 <target>${java.version}</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**Output:**



**Hello World RESTful Web Service**

**Code:**

**SpringHelloApplication.java:**

package com.cognizant.springhello;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
@SpringBootApplication  
public class SpringHelloApplication{  
 private static final Logger *LOGGER*=LoggerFactory.*getLogger*(SpringHelloApplication.class);  
 public static void main(String[] args){  
 *LOGGER*.info("Starting SpringHelloApplication");  
 SpringApplication.*run*(SpringHelloApplication.class, args);  
 *LOGGER*.info("SpringHelloApplication Started Successfully");  
 }  
}

**HelloController.java:**

package com.cognizant.springhello.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 This is Akshaya!!";  
 *LOGGER*.info("END - sayHello()");  
 return message;  
 }  
}

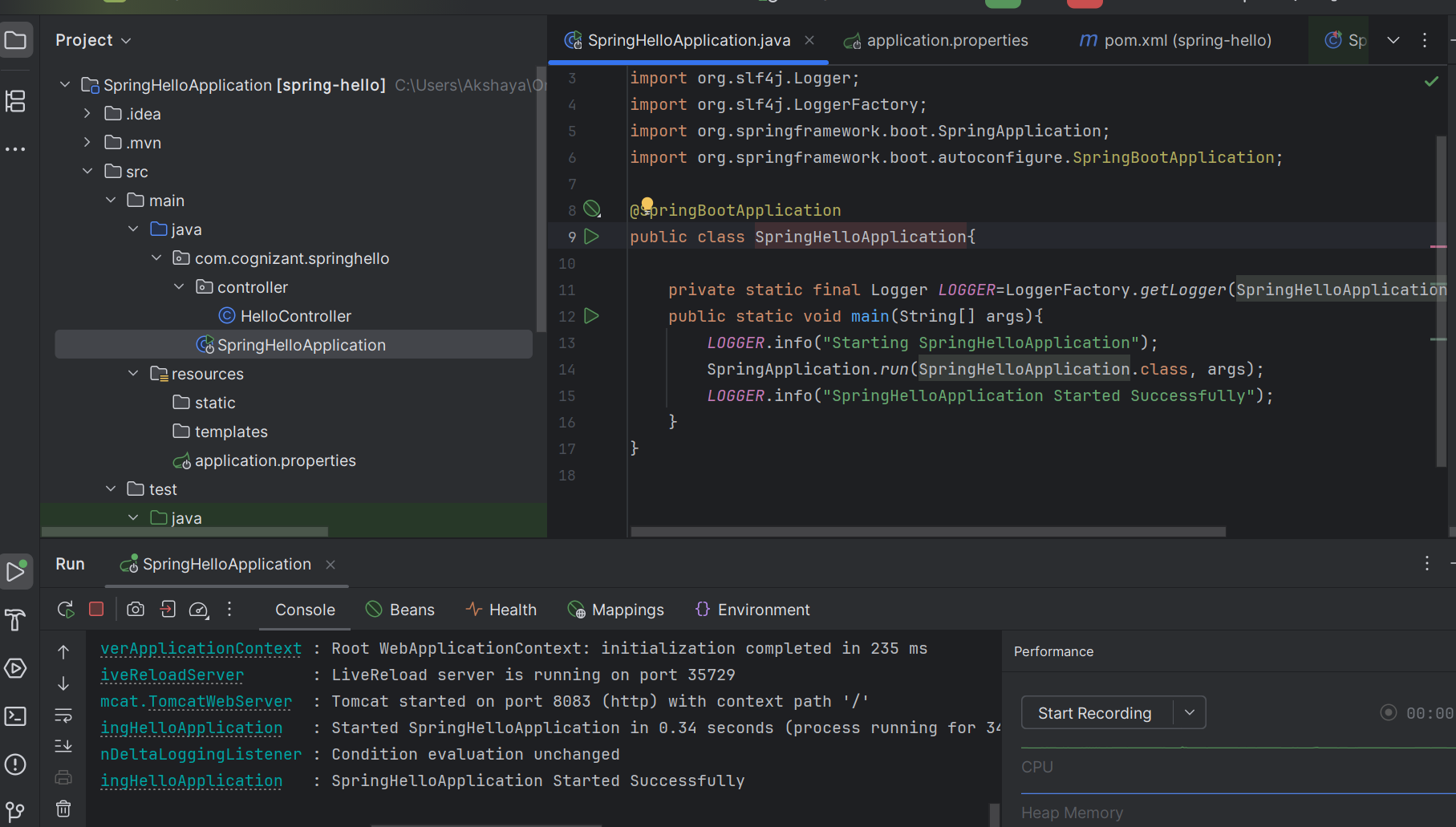
**application.properties:**

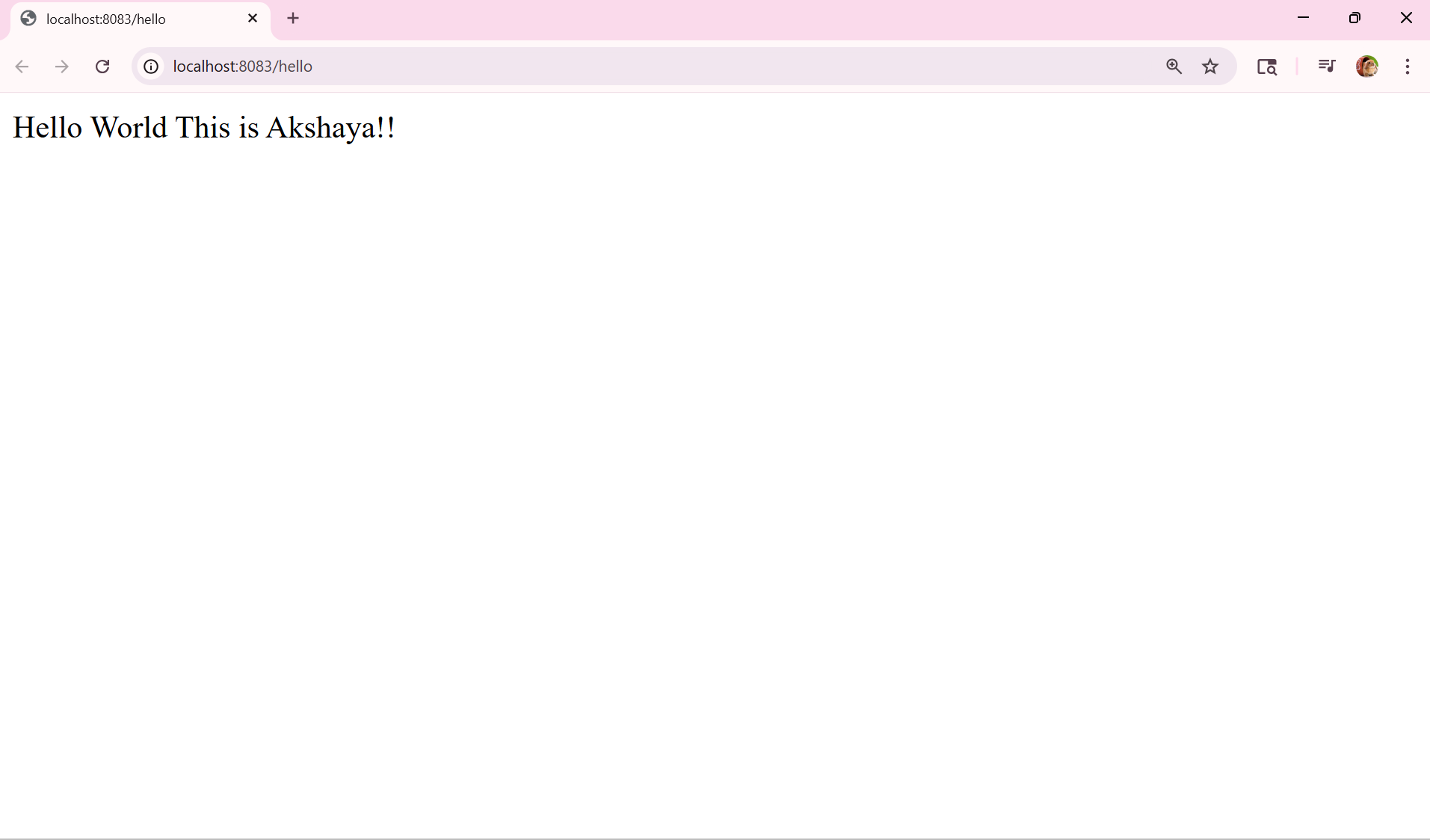
spring.application.name=SpringHelloApplication  
server.port=8083

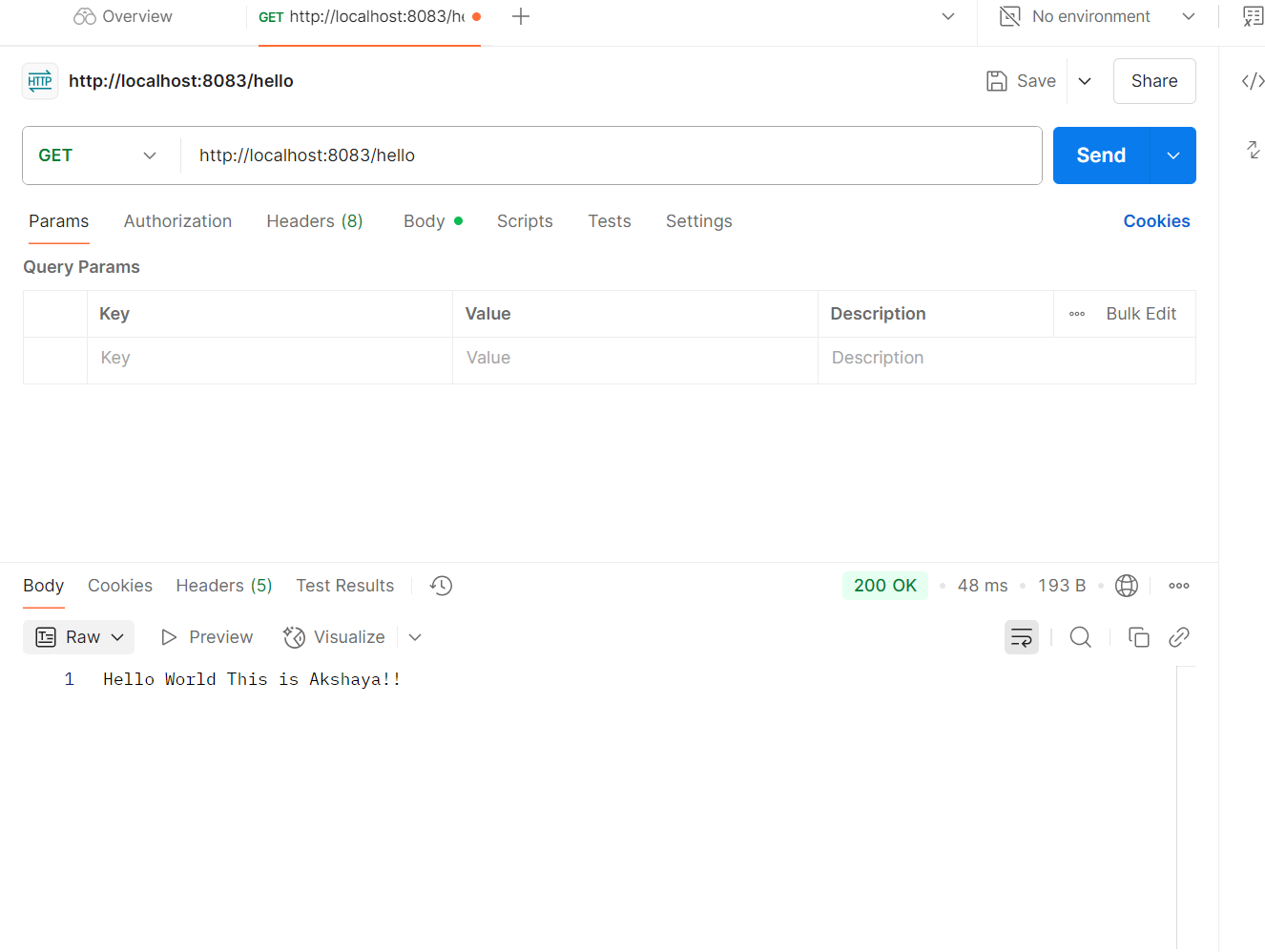
**pom.xml:**

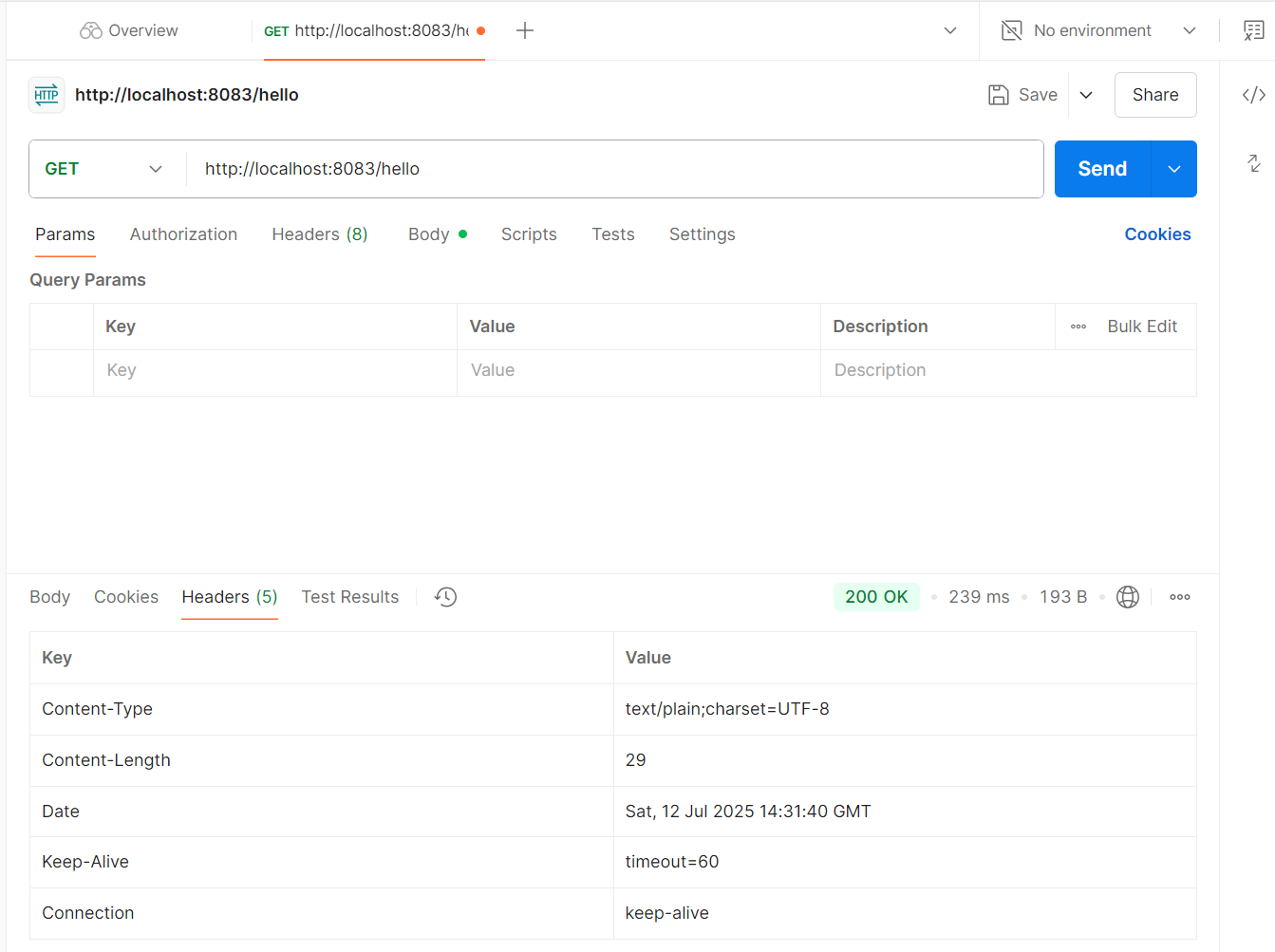
<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.cognizant</groupId>  
 <artifactId>spring-hello</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>SpringHelloApplication</name>  
 <description>SpringHelloApplication</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.slf4j</groupId>  
 <artifactId>slf4j-simple</artifactId>  
 <version>2.0.13</version>  
 </dependency>  
 </dependencies>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**Output:**

****

****

****

****

**REST - Country Web Service**

**Code:**

**CountryController.java:**

package com.cognizant.springrestcountry.controller;  
import com.cognizant.springrestcountry.model.Country;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
@RestController  
public class CountryController{  
 private static final Logger *LOGGER*=LoggerFactory.*getLogger*(CountryController.class);  
 @GetMapping("/country")  
 public Country getCountryIndia(){  
 *LOGGER*.info("Start - getCountryIndia()");  
 ApplicationContext context=new ClassPathXmlApplicationContext("country.xml");  
 Country country=context.getBean("country",Country.class);  
 *LOGGER*.info("End - getCountryIndia()");  
 return country;  
 }  
}

**Country.java:**

package com.cognizant.springrestcountry.model;  
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("Getting country code");  
 return code;  
 }  
 public void setCode(String code){  
 *LOGGER*.debug("Setting country code: {}", code);  
 this.code = code;  
 }  
 public String getName(){  
 *LOGGER*.debug("Getting country name");  
 return name;  
 }  
 public void setName(String name){  
 *LOGGER*.debug("Setting country name: {}", name);  
 this.name = name;  
 }  
 @Override  
 public String toString(){  
 return "Country{" + "code='" + code + '\'' + ", name='" + name + '\'' + '}';  
 }  
}

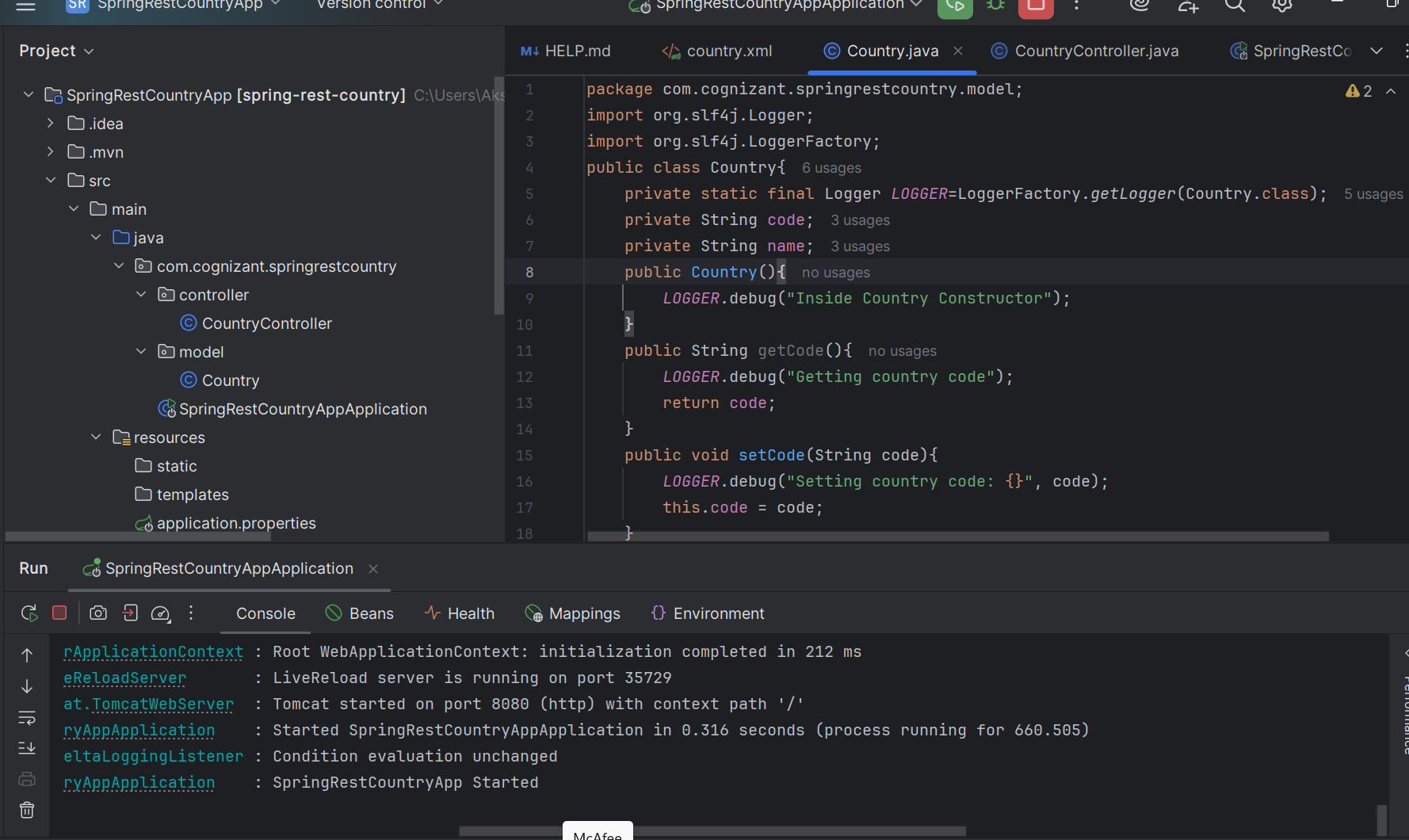
**SpringRestCountryAppApplication.java:**

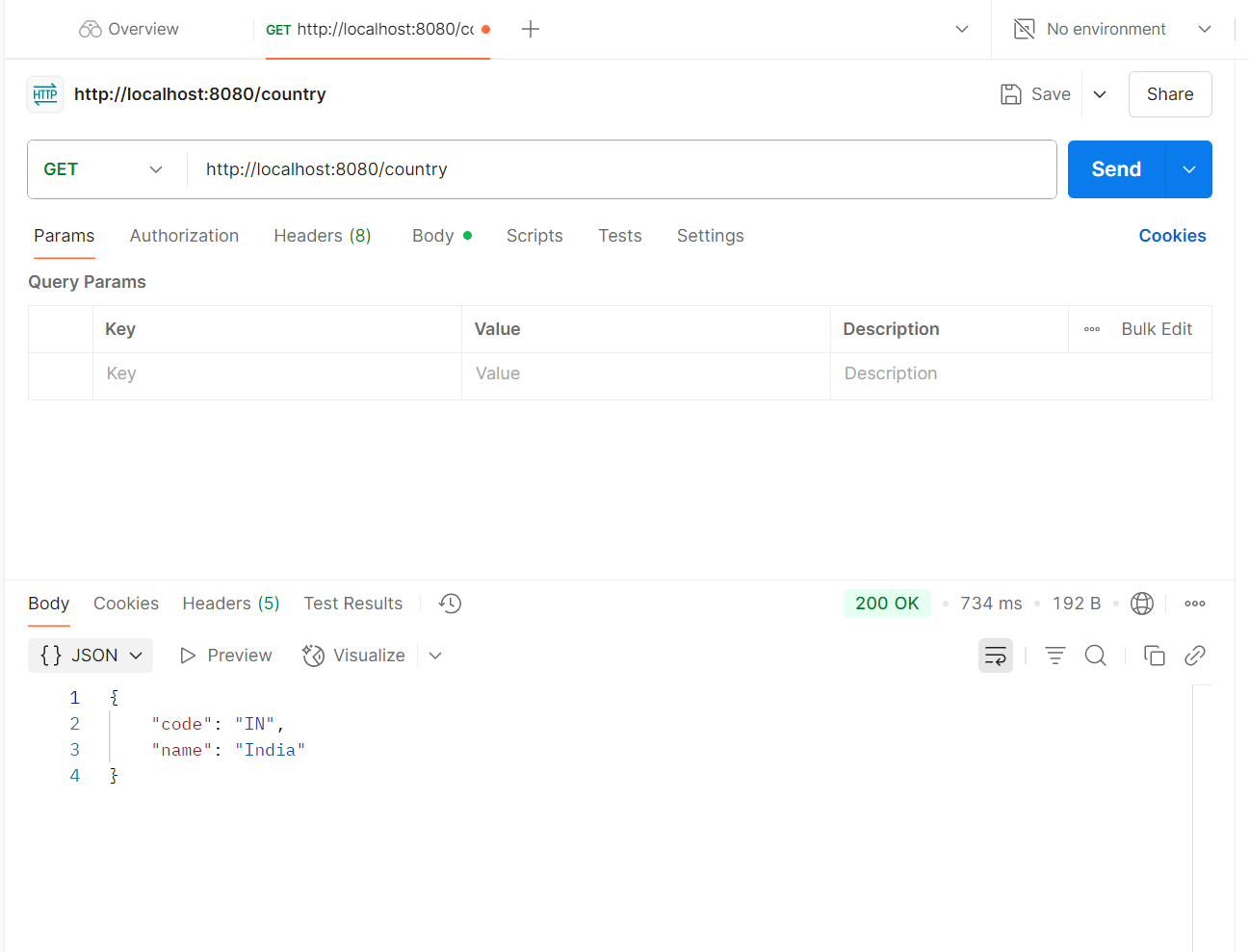
package com.cognizant.springrestcountry;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
@SpringBootApplication  
public class SpringRestCountryAppApplication{  
 private static final Logger *LOGGER*=LoggerFactory.*getLogger*(SpringRestCountryAppApplication.class);  
 public static void main(String[] args){  
 *LOGGER*.info("Starting SpringRestCountryApp");  
 SpringApplication.*run*(SpringRestCountryAppApplication.class, args);  
 *LOGGER*.info("SpringRestCountryApp Started");  
 }  
}

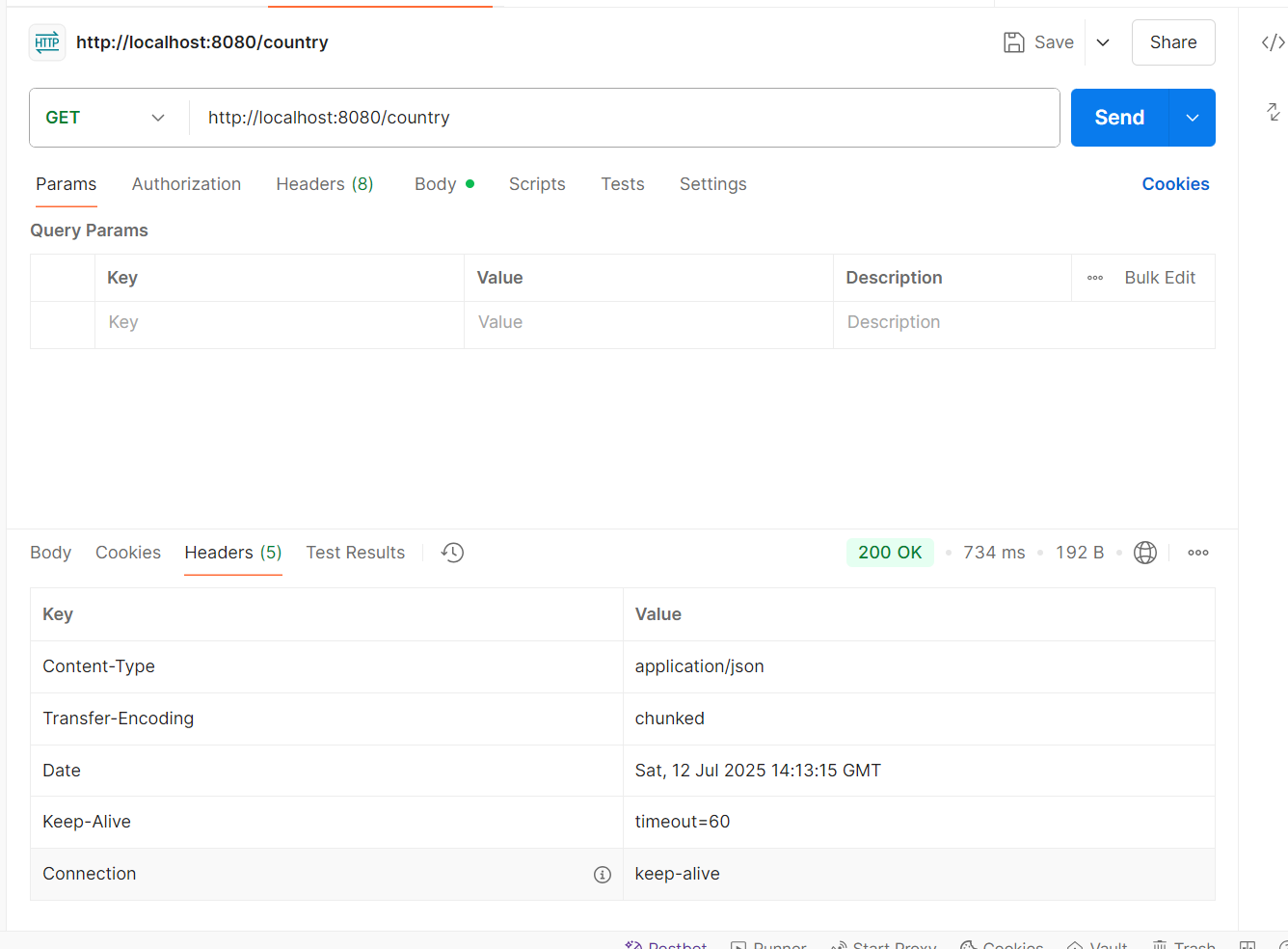
**country.xml:**

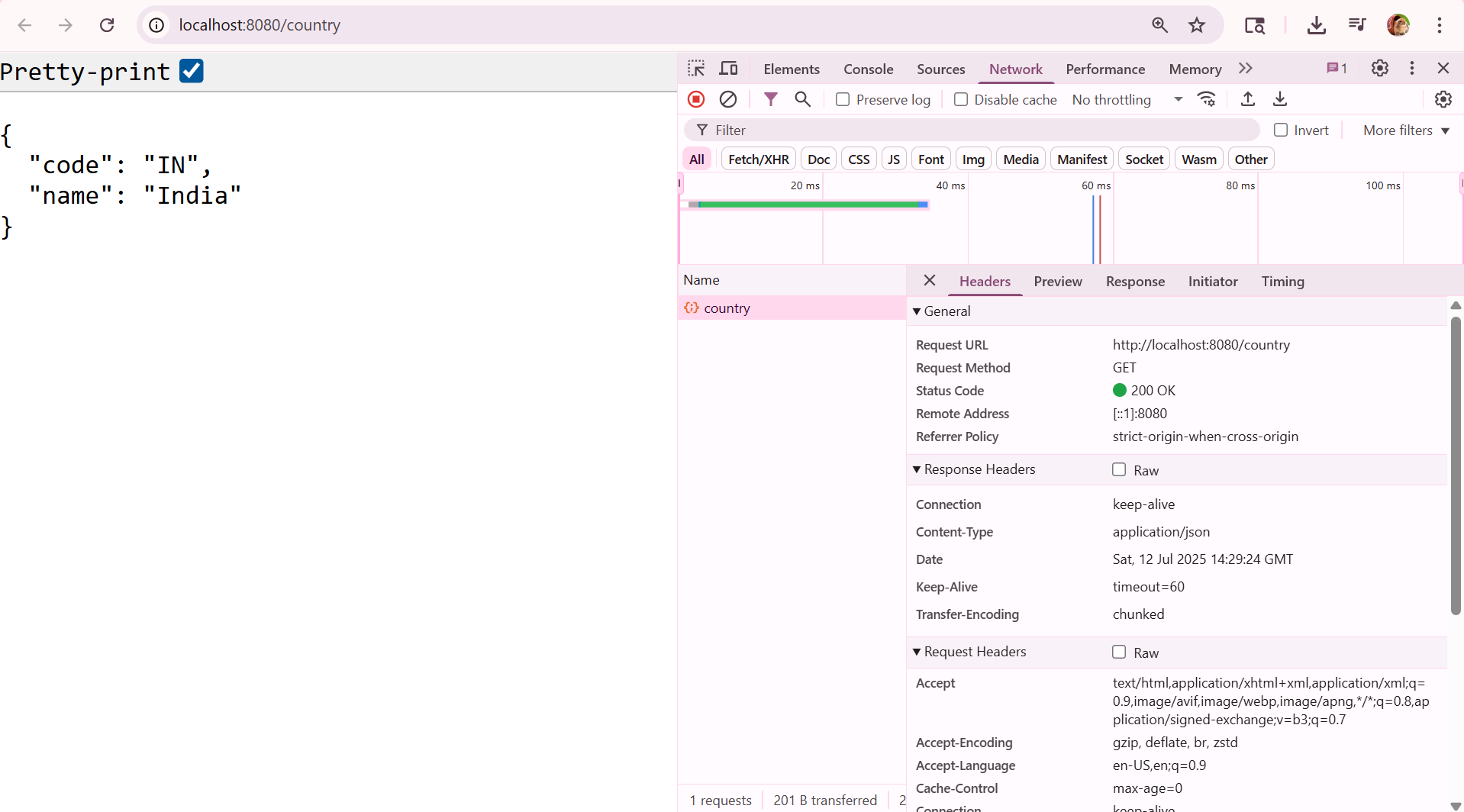
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
 <bean id="country" class="com.cognizant.springrestcountry.model.Country">  
 <property name="code" value="IN" />  
 <property name="name" value="India" />  
 </bean>  
</beans>

**Output:**

****

****

****

****

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

**Code:**

**Country.java:**

package com.cognizant.springcountryservice.model;  
public class Country{  
 private String code;  
 private String name;  
 public Country(){  
 System.*out*.println("Inside Country Constructor");  
 }  
 public String getCode(){  
 System.*out*.println("Getter: getCode()");  
 return code;  
 }  
 public void setCode(String code){  
 System.*out*.println("Setter: setCode()");  
 this.code=code;  
 }  
 public String getName(){  
 System.*out*.println("Getter: getName()");  
 return name;  
 }  
 public void setName(String name){  
 System.*out*.println("Setter: setName()");  
 this.name=name;  
 }  
 @Override  
 public String toString(){  
 return "Country{" + "code='" + code + '\'' + ", name='" + name + '\'' + '}';  
 }  
}

**CountryController.java:**

package com.cognizant.springcountryservice.controller;  
import com.cognizant.springcountryservice.model.Country;  
import com.cognizant.springcountryservice.service.CountryService;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
@RestController  
public class CountryController{  
 private static final Logger *LOGGER*=LoggerFactory.*getLogger*(CountryController.class);  
 @Autowired  
 private CountryService countryService;  
 @GetMapping("/countries/{code}")  
 public Country getCountry(@PathVariable String code){  
 *LOGGER*.info("START: getCountry with code: {}", code);  
 Country country=countryService.getCountry(code);  
 *LOGGER*.info("END: getCountry");  
 return country;  
 }  
}

**CountryService.java:**

package com.cognizant.springcountryservice.service;  
import com.cognizant.springcountryservice.model.Country;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import org.springframework.stereotype.Service;  
import java.util.List;  
@Service  
public class CountryService{  
 public Country getCountry(String code){  
 ApplicationContext context=new ClassPathXmlApplicationContext("country.xml");  
 List<Country>countryList=context.getBean("countryList", List.class);  
 String trimmedCode=code.trim();  
 return countryList.stream()  
 .filter(country->country.getCode().equalsIgnoreCase(trimmedCode))  
 .findFirst()  
 .orElseThrow(()->new RuntimeException("Country not found: " + trimmedCode));  
 }  
}

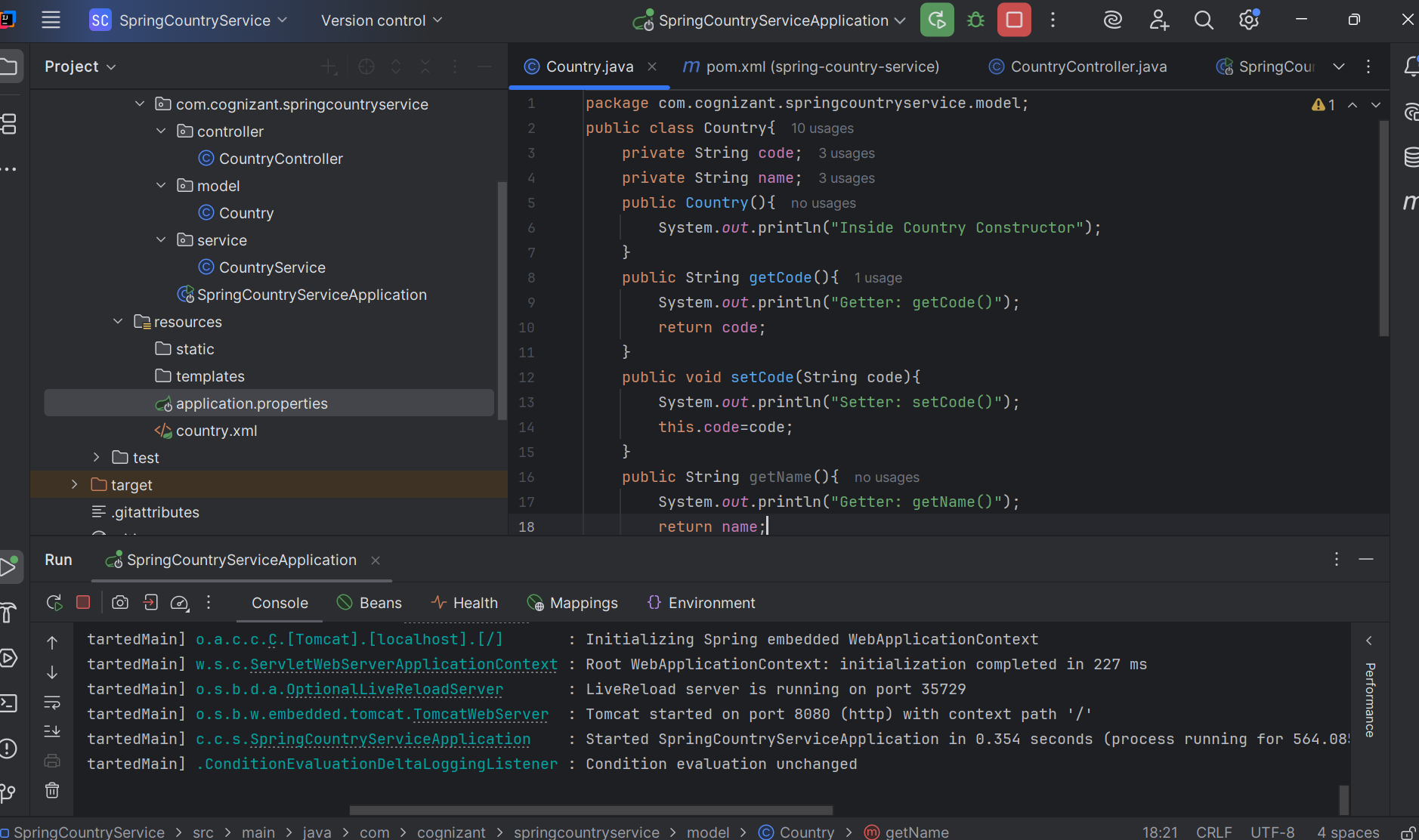
**SpringCountryServiceApplication.java:**

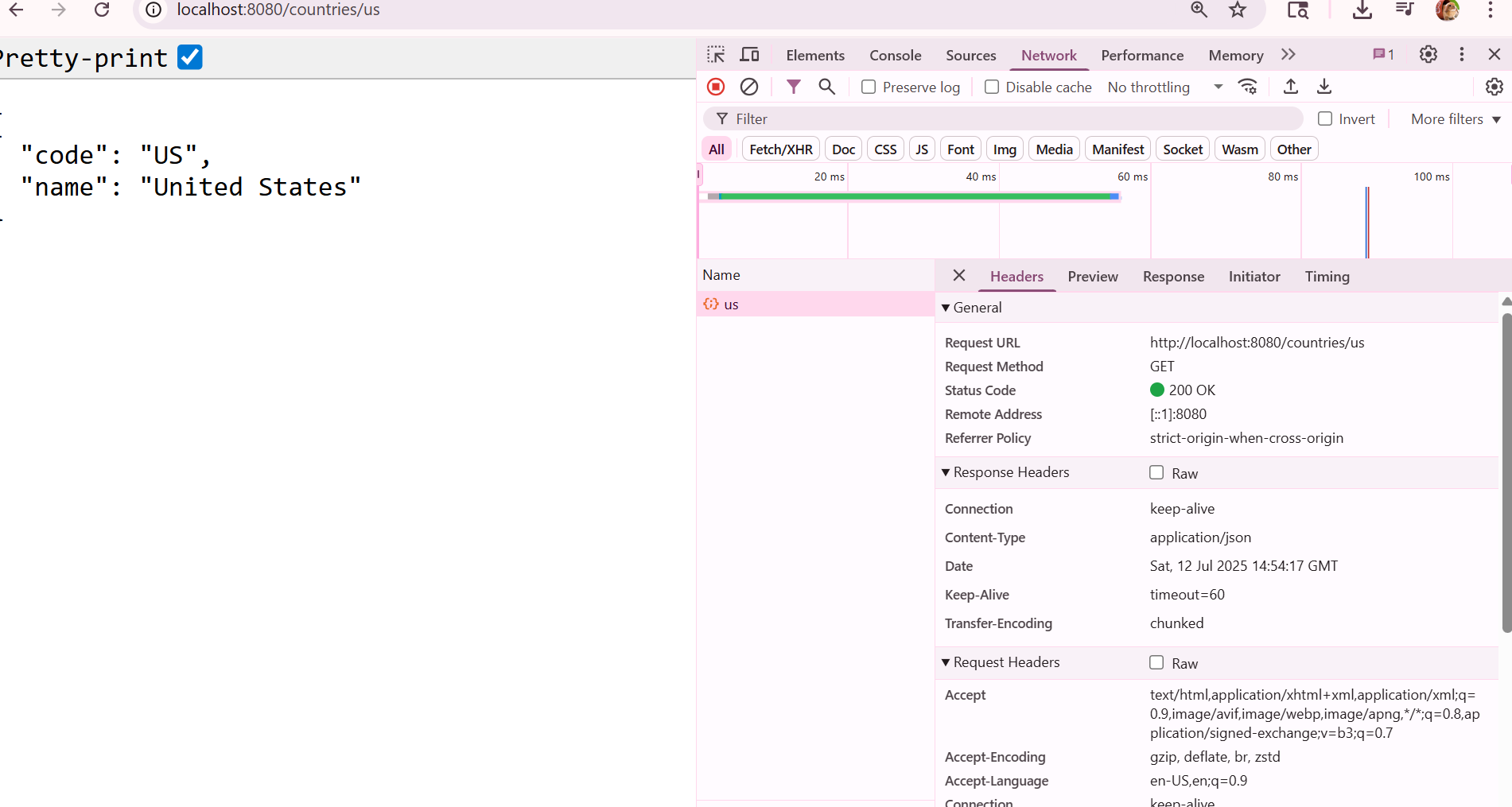
package com.cognizant.springcountryservice;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
@SpringBootApplication  
public class SpringCountryServiceApplication{  
 public static void main(String[] args){  
 SpringApplication.*run*(SpringCountryServiceApplication.class, args);  
 }  
}

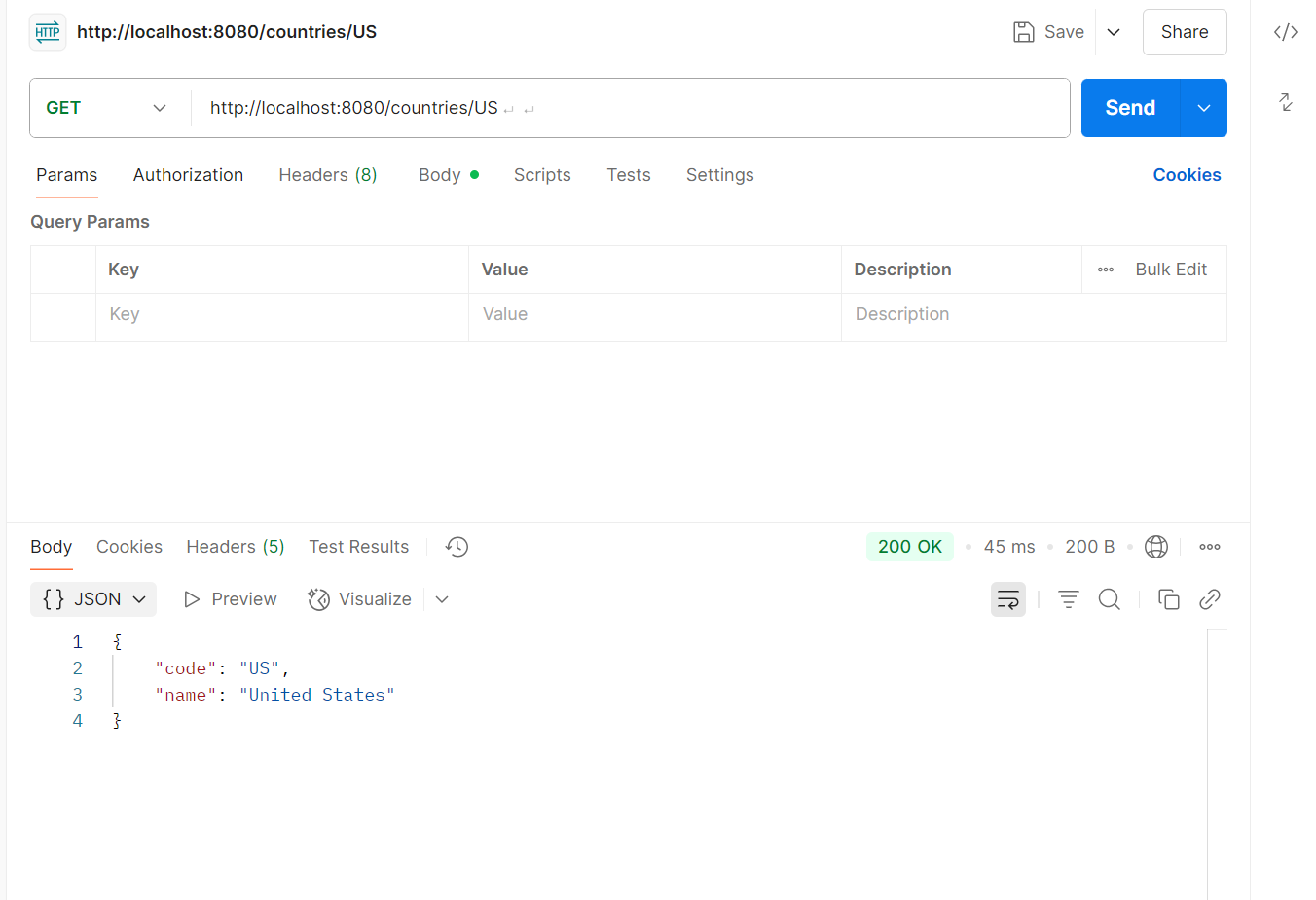
**country.xml:**

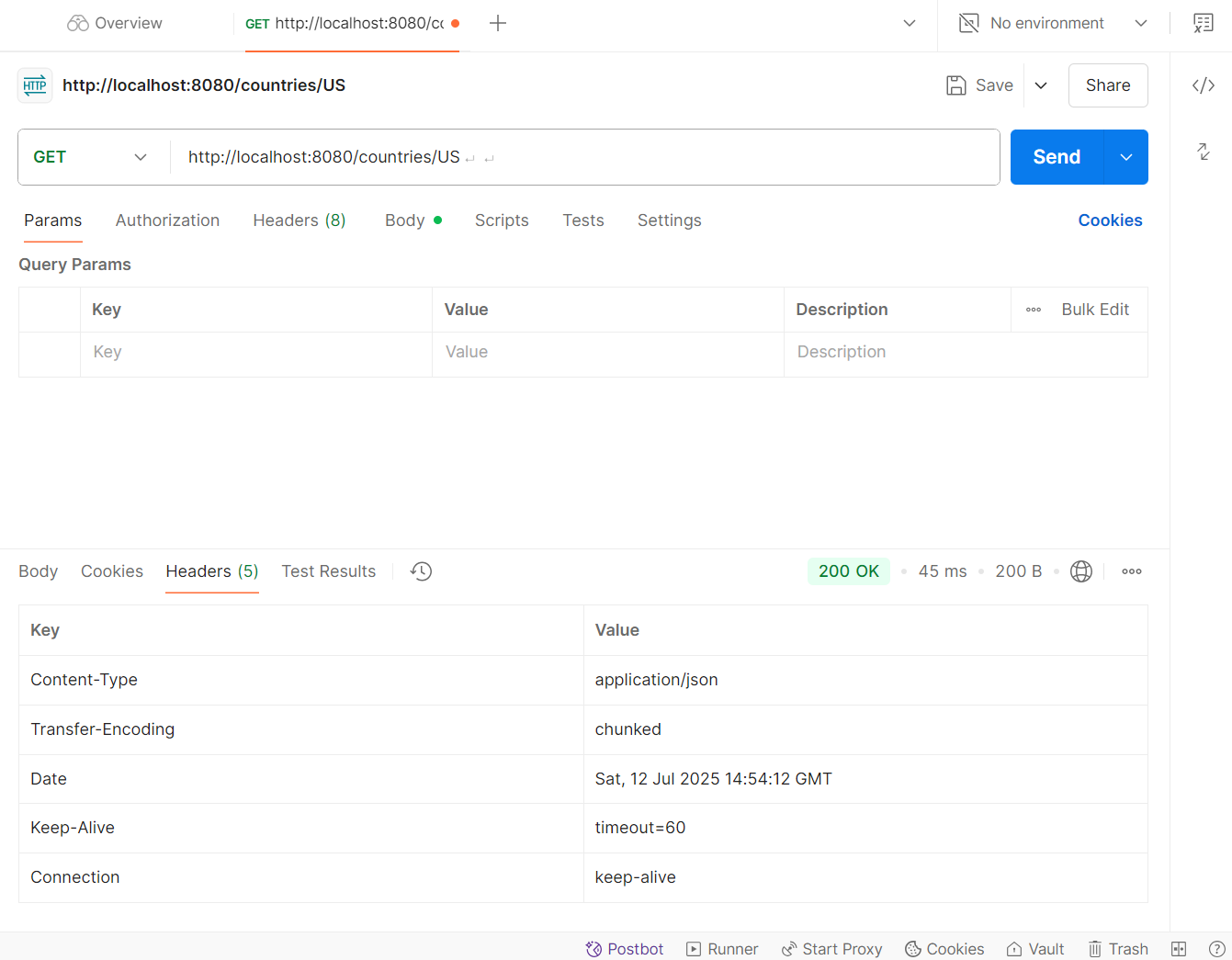
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 https://www.springframework.org/schema/beans/spring-beans.xsd">  
 <bean id="countryList" class="java.util.ArrayList">  
 <constructor-arg>  
 <list>  
 <bean class="com.cognizant.springcountryservice.model.Country">  
 <property name="code" value="IN"/>  
 <property name="name" value="India"/>  
 </bean>  
 <bean class="com.cognizant.springcountryservice.model.Country">  
 <property name="code" value="US"/>  
 <property name="name" value="United States"/>  
 </bean>  
 <bean class="com.cognizant.springcountryservice.model.Country">  
 <property name="code" value="DE"/>  
 <property name="name" value="Germany"/>  
 </bean>  
 <bean class="com.cognizant.springcountryservice.model.Country">  
 <property name="code" value="JP"/>  
 <property name="name" value="Japan"/>  
 </bean>  
 </list>  
 </constructor-arg>  
 </bean>  
</beans>

**Output:**

****

****

****

****

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

**Code:**

**SecurityConfig.java:**

package com.cognizant.jwtauth.config;  
import com.cognizant.jwtauth.service.CustomUserDetailsService;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;  
import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;  
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.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.web.SecurityFilterChain;  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig{  
 @Bean  
 public PasswordEncoder passwordEncoder(){  
 return new BCryptPasswordEncoder();  
 }  
 @Bean  
 public DaoAuthenticationProvider daoAuthenticationProvider(CustomUserDetailsService userDetailsService){  
 DaoAuthenticationProvider authProvider=new DaoAuthenticationProvider();  
 authProvider.setUserDetailsService(userDetailsService);  
 authProvider.setPasswordEncoder(passwordEncoder());  
 return authProvider;  
 }  
 @Bean  
 public AuthenticationManager authenticationManager(AuthenticationConfiguration config)throws Exception{  
 return config.getAuthenticationManager();  
 }  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http,DaoAuthenticationProvider authProvider)throws Exception {  
 http.csrf(csrf->csrf.disable())  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers("/authenticate").permitAll()  
 .anyRequest().authenticated()  
 )  
 .sessionManagement(session -> session  
 .sessionCreationPolicy(SessionCreationPolicy.*STATELESS*)  
 )  
 .authenticationProvider(authProvider);  
  
 return http.build();  
 }  
}

**AuthController.java:**

package com.cognizant.jwtauth.controller;  
import com.cognizant.jwtauth.dto.AuthResponse;  
import com.cognizant.jwtauth.util.JwtUtil;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
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.security.core.AuthenticationException;  
import org.springframework.web.bind.annotation.\*;  
import jakarta.servlet.http.HttpServletRequest;  
import java.nio.charset.StandardCharsets;  
import java.util.Base64;  
@RestController  
public class AuthController{  
 @Autowired  
 private AuthenticationManager authenticationManager;  
 @Autowired  
 private JwtUtil jwtUtil;  
 @PostMapping("/authenticate")  
 public ResponseEntity<?> authenticate(HttpServletRequest request){  
 try{  
 String authHeader=request.getHeader("Authorization");  
 if(authHeader==null || !authHeader.startsWith("Basic ")){  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*)  
 .body("Authorization header missing or invalid");  
 }  
 String base64Credentials=authHeader.substring("Basic ".length());  
 String credentials=new String(Base64.*getDecoder*().decode(base64Credentials), StandardCharsets.*UTF\_8*);  
 String[] parts=credentials.split(":", 2);  
 if(parts.length!=2){  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*)  
 .body("Invalid credentials format");  
 }  
 String username=parts[0];  
 String password=parts[1];  
 Authentication authentication=authenticationManager.authenticate(  
 new UsernamePasswordAuthenticationToken(username, password)  
 );  
 String token=jwtUtil.generateToken(username);  
 return ResponseEntity.*ok*(new AuthResponse(token));  
 }   
 catch(AuthenticationException e){  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*)  
 .body("Invalid credentials");  
 }   
 catch(Exception e){  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*)  
 .body("Authentication failed");  
 }  
 }  
}

**AuthResponse.java:**

package com.cognizant.jwtauth.dto;  
public class AuthResponse{  
 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;  
 }  
}

**CustomAuthenticationFilter.java:**

package com.cognizant.jwtauth.filter;  
import com.cognizant.jwtauth.dto.AuthResponse;  
import com.cognizant.jwtauth.util.JwtUtil;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.AuthenticationException;  
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;  
import jakarta.servlet.FilterChain;  
import jakarta.servlet.ServletException;  
import jakarta.servlet.http.HttpServletRequest;  
import jakarta.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.nio.charset.StandardCharsets;  
import java.util.ArrayList;  
import java.util.Base64;  
public class CustomAuthenticationFilter extends UsernamePasswordAuthenticationFilter{  
 private final AuthenticationManager authenticationManager;  
 private final JwtUtil jwtUtil;  
 public CustomAuthenticationFilter(AuthenticationManager authenticationManager, JwtUtil jwtUtil){  
 this.authenticationManager=authenticationManager;  
 this.jwtUtil=jwtUtil;  
 setFilterProcessesUrl("/authenticate");  
 }  
 @Override  
 public Authentication attemptAuthentication(HttpServletRequest request,  
 HttpServletResponse response) throws AuthenticationException{  
  
 String authHeader=request.getHeader("Authorization");  
 if(authHeader==null||!authHeader.startsWith("Basic ")){  
 throw new RuntimeException("Authorization header missing or invalid");  
 }  
 String base64Credentials=authHeader.substring("Basic ".length());  
 String credentials=new String(Base64.*getDecoder*().decode(base64Credentials), StandardCharsets.*UTF\_8*);  
 String[] parts=credentials.split(":", 2);  
 if(parts.length != 2){  
 throw new RuntimeException("Invalid credentials format");  
 }  
 String username=parts[0];  
 String password=parts[1];  
 UsernamePasswordAuthenticationToken authToken =  
 new UsernamePasswordAuthenticationToken(username, password, new ArrayList<>());  
 return authenticationManager.authenticate(authToken);  
 }  
 @Override  
 protected void successfulAuthentication(HttpServletRequest request,  
 HttpServletResponse response,  
 FilterChain chain,  
 Authentication authResult) throws IOException, ServletException {  
  
 String username=authResult.getName();  
 String token=jwtUtil.generateToken(username);  
 AuthResponse authResponse=new AuthResponse(token);  
 response.setContentType("application/json");  
 response.setCharacterEncoding("UTF-8");  
 ObjectMapper mapper=new ObjectMapper();  
 response.getWriter().write(mapper.writeValueAsString(authResponse));  
 }  
}

**CustomUserDetailsService.java:**

package com.cognizant.jwtauth.service;  
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.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Service;  
import java.util.ArrayList;  
@Service  
public class CustomUserDetailsService implements UserDetailsService{  
 private final PasswordEncoder passwordEncoder;  
  
 public CustomUserDetailsService(PasswordEncoder passwordEncoder){  
 this.passwordEncoder=passwordEncoder;  
 }  
 @Override  
 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException{  
 if ("akshaya".equals(username)){  
 return new User("akshaya", passwordEncoder.encode("aks@123"), new ArrayList<>());  
 }  
 throw new UsernameNotFoundException("User not found: " + username);  
 }  
}

**JwtUtil.java:**

package com.cognizant.jwtauth.util;  
import io.jsonwebtoken.Claims;  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import io.jsonwebtoken.security.Keys;  
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{  
 private final String SECRET\_KEY="mySecretKey12345mySecretKey12345mySecretKey12345";  
 private final SecretKey key=Keys.*hmacShaKeyFor*(SECRET\_KEY.getBytes());  
 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(key)  
 .build()  
 .parseClaimsJws(token)  
 .getBody();  
 }  
 private Boolean isTokenExpired(String token){  
 return extractExpiration(token).before(new Date());  
 }  
 public String generateToken(String username){  
 Map<String, Object> claims=new HashMap<>();  
 return createToken(claims, username);  
 }  
 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 \* 20)) // 20 minutes  
 .signWith(key, SignatureAlgorithm.*HS256*)  
 .compact();  
 }  
 public Boolean validateToken(String token,String username){  
 final String extractedUsername=extractUsername(token);  
 return (extractedUsername.equals(username) && !isTokenExpired(token));  
 }  
}

**JwtAuthApplication.java:**

package com.cognizant.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  
spring.application.name=jwt-auth-service

**Output:**

