**Week 4 (2) - 6428652**

**Ex 3:**

**Hello World RESTful Web Service** 3d  
Write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework. Refer details below

**Solution:**

**SpringLearn Application.java:**

**package** com.cognizant.spring\_learn.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 response = "Hello World!!";

***LOGGER***.info("END - sayHello()");

**return** response;

}

}

**Hellocontroller.java:**

package com.cognizant.spring\_learn.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 response = "Hello World!!";

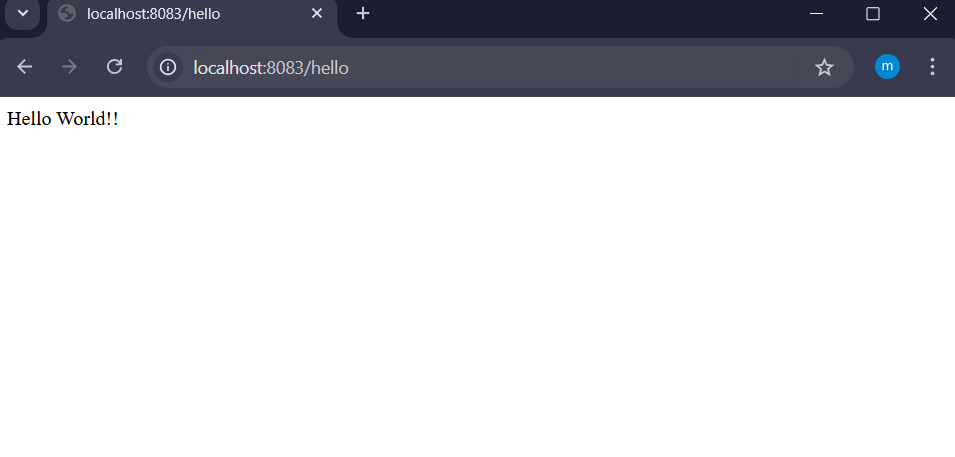
LOGGER.info("END - sayHello()");

return response;

}

}

**OUTPUT:**

  
**Ex 4:**

**REST - Country Web Service**   
  
Write a REST service that returns India country details in the earlier created spring learn application.  
**URL**: /country  
**Controller**: com.cognizant.spring-learn.controller.CountryController  
**Method Annotation**: @RequestMapping  
**Method Name**: getCountryIndia()  
**Method Implementation**: Load India bean from spring xml configuration and return  
**Sample Request**: http://localhost:8083/country  
**Sample Response**:

{

  "code": "IN",

  "name": "India"

}  
SME to explain the following aspects:

* What happens in the controller method?
* How the bean is converted into JSON reponse?
* In network tab of developer tools show the HTTP header details received
* In postman click on "Headers" tab to view the HTTP header details received

**SpringLearnApplication.java:**

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) {

LOGGER.info("START");

SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("END");

}

}

**Country.java:**

package com.cognizant.spring\_learn.model;

public class Country {

private String code;

private String name;

public Country() {

super();

}

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*

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

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

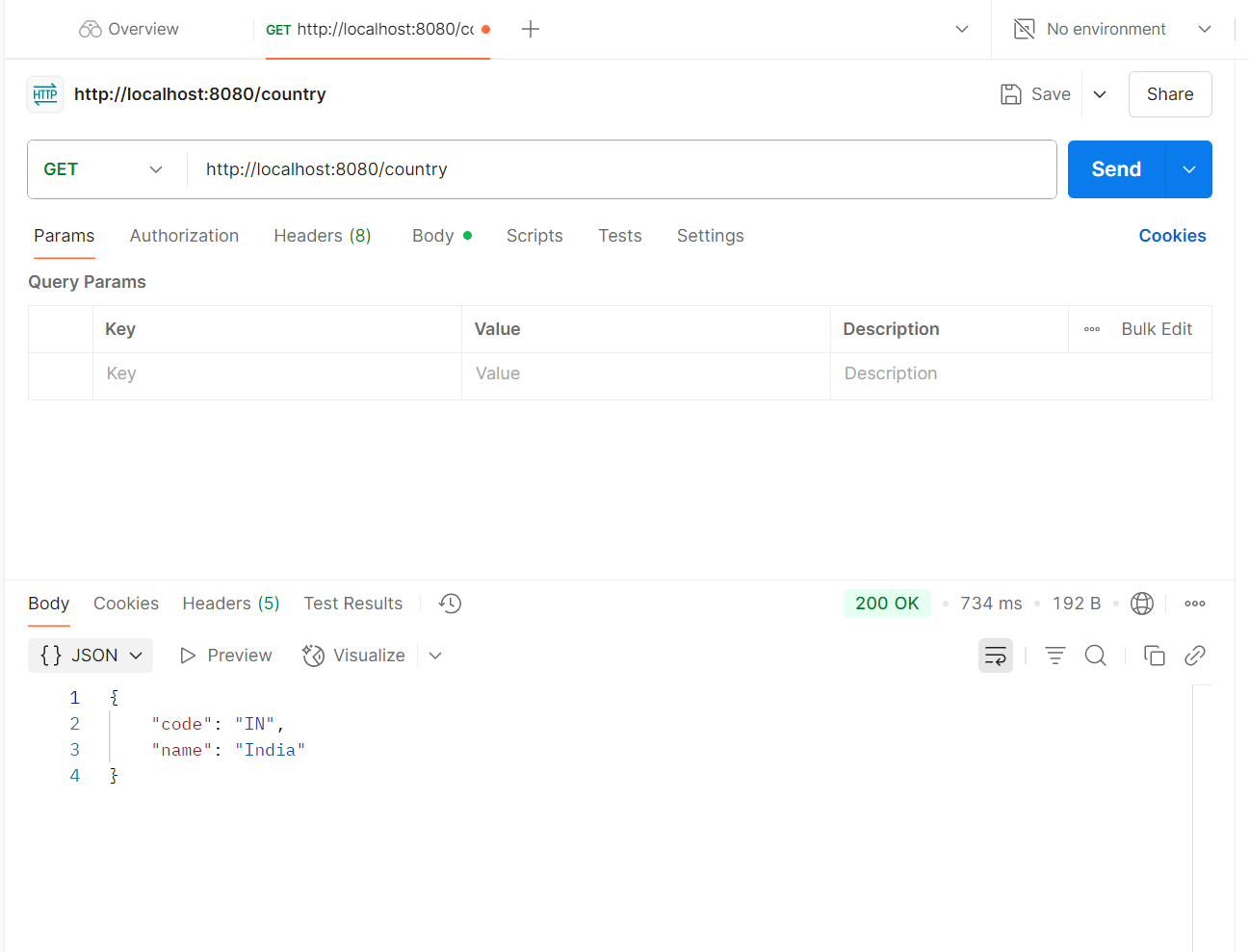
<property name=*"code"* value=*"IN"*/>

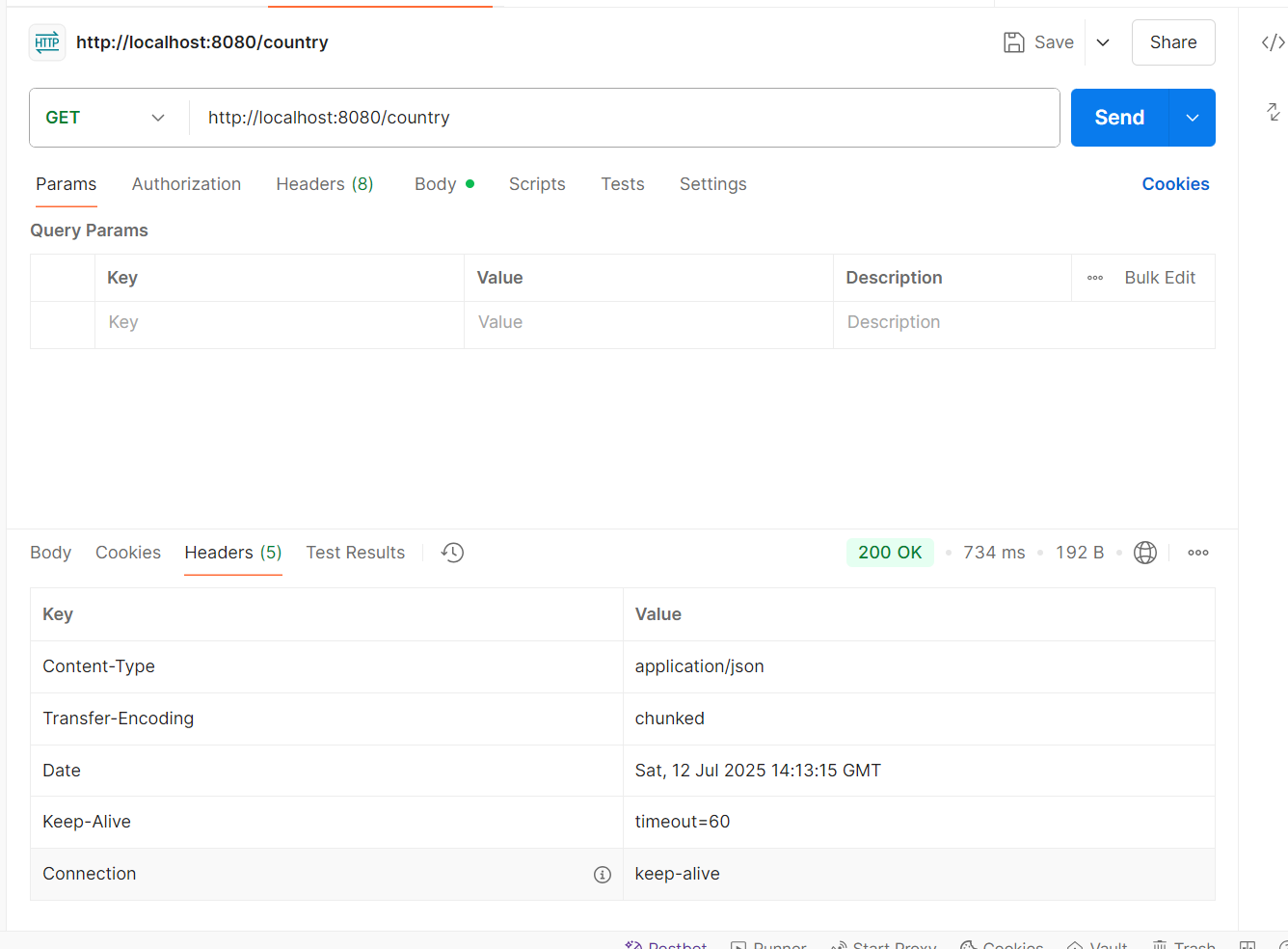
<property name=*"name"* value=*"India"*/>

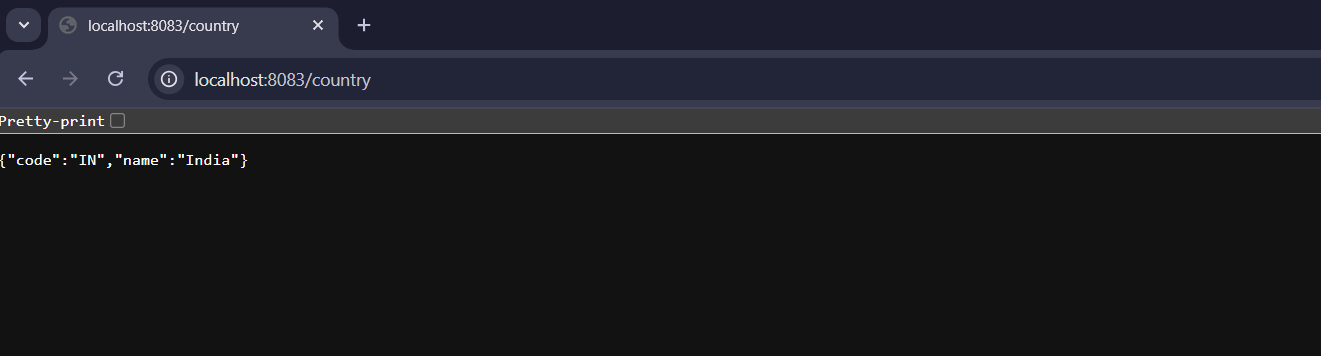
</bean>

</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));  
 }  
}

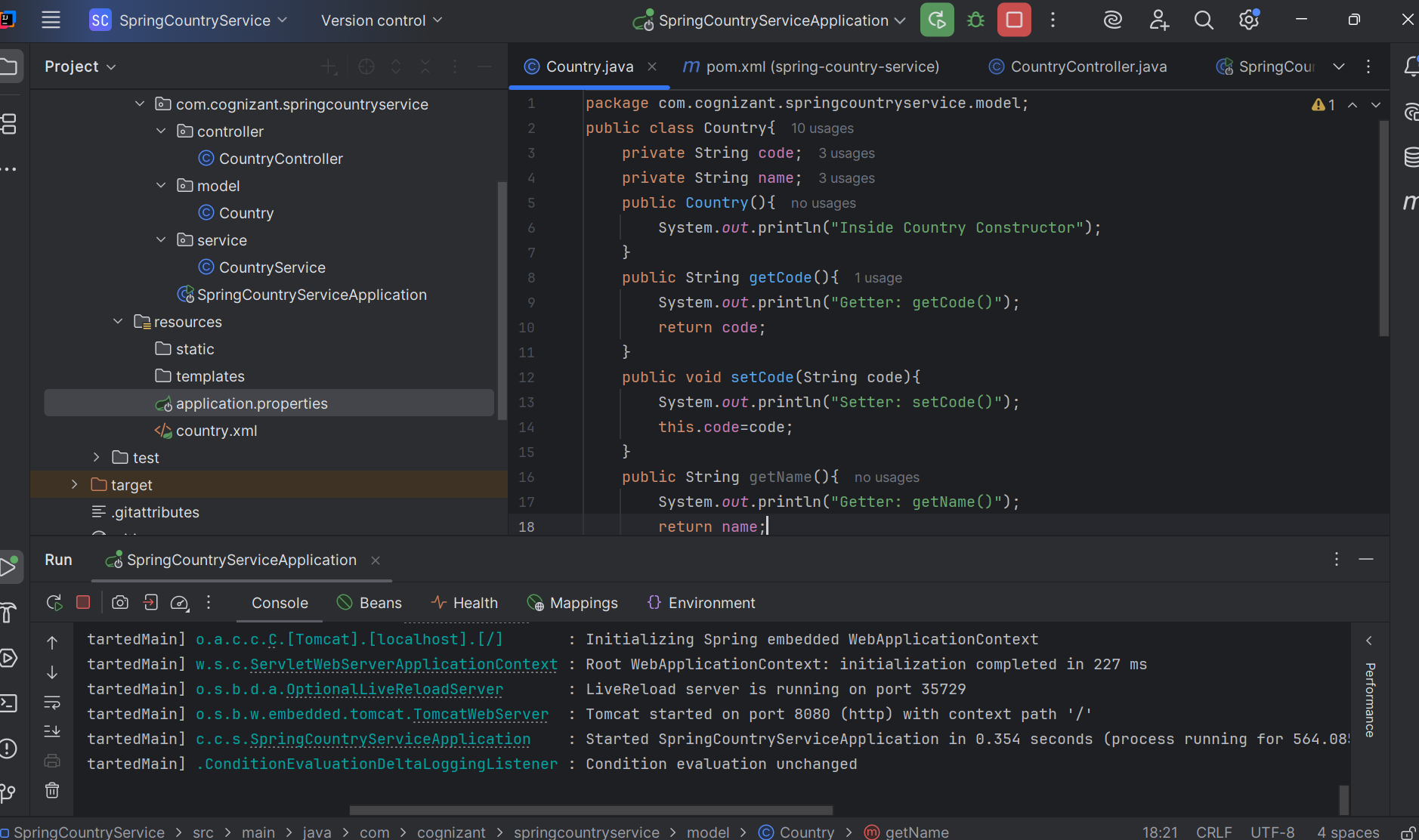
**SpringCountryServiceApplication.java:**

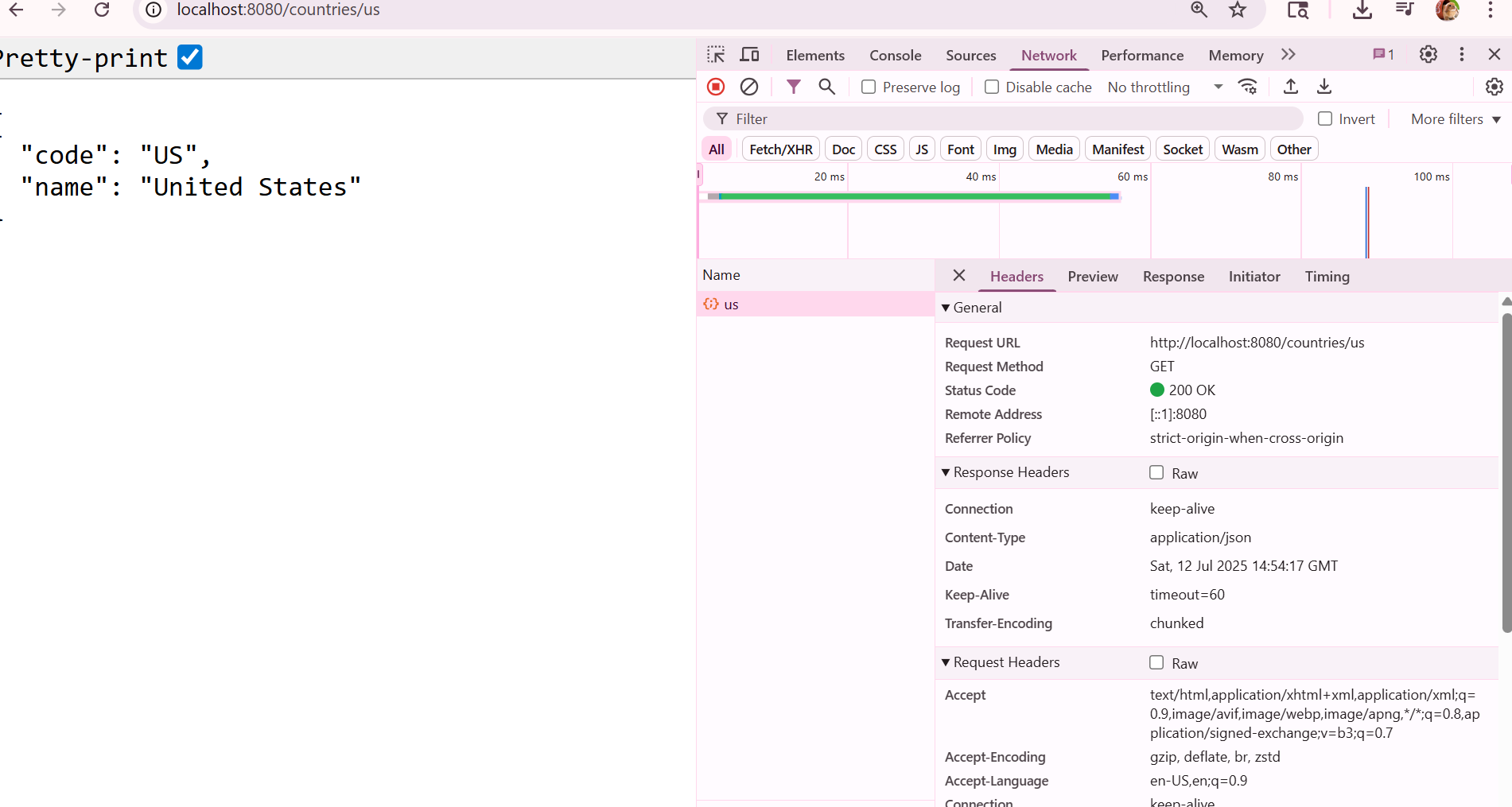
package com.cognizant.springcountryservice;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
@SpringBootApplication  
public class SpringCountryServiceApplication{  
 public static void main(String[] args){  
 SpringApplication.*run*(SpringCountryServiceApplication.class,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

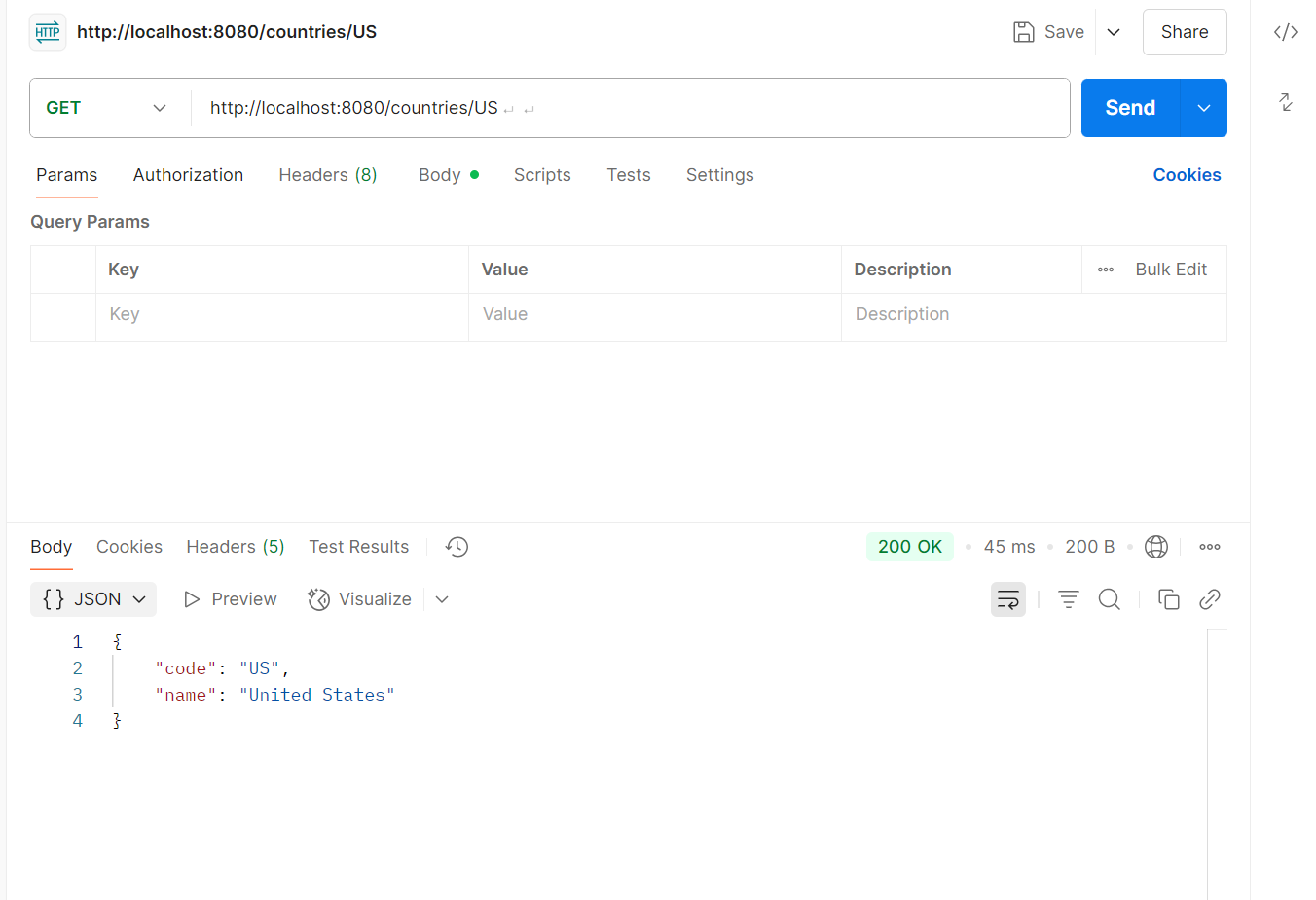
**country.xml:**

<?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

