**Exercise 1:Create a Spring Web Project using Maven:**

**SpringLearnApplication.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 SpringLearnApplication {

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

public static void main(String[] args) {

LOGGER.info("START");

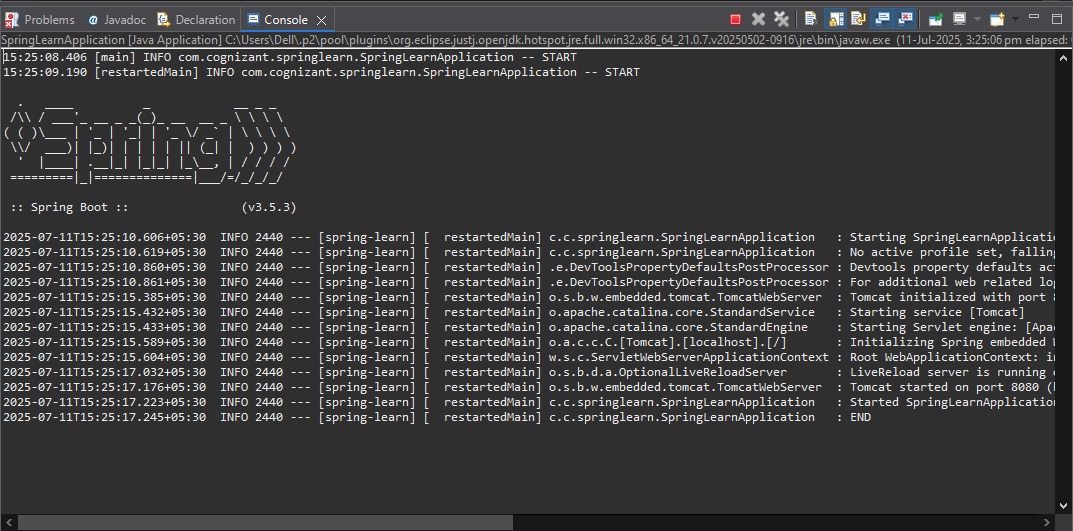
SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("END");

}

}

**Output:**



**Exercise 2 Spring Core – Load SimpleDateFormat from Spring Configuration XML:**

**date-format.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="dateFormat" class="java.text.SimpleDateFormat">

<constructor-arg value="dd/MM/yyyy" />

</bean>

</beans>

**SpringLearnApplication.java:**

package com.cognizant.spring\_learn;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

public static void main(String[] args) {

displayDate();

}

public static void displayDate() {

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);

try {

Date date = format.parse("31/12/2018");

System.out.println("Parsed Date: " + date);

} catch (ParseException e) {

System.out.println("Error parsing date: " + e.getMessage());

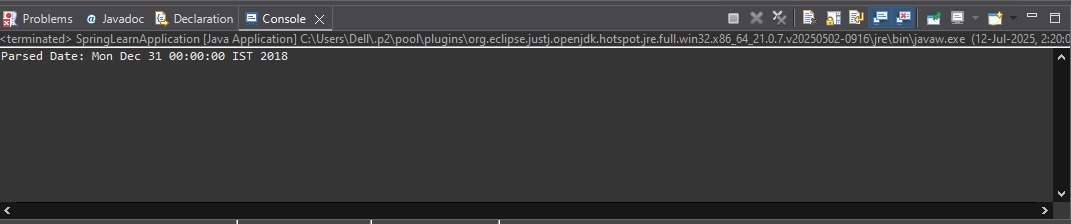
}

((ClassPathXmlApplicationContext) context).close();

}

}

**Output:**

****

**Exercise 3: Spring Core – Load Country from Spring Configuration XML:**

**SpringLearnApplication.java:**

package com.cognizant.springlearn;

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) {

LOGGER.debug("START");

SpringApplication.run(SpringLearnApplication.class, args);

displayCountry();

LOGGER.debug("END");

}

public static void displayCountry() {

try (ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("country.xml")) {

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

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

}

}

}

**Country.java:**

package com.cognizant.springlearn;

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 + "]";

}

}

**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>

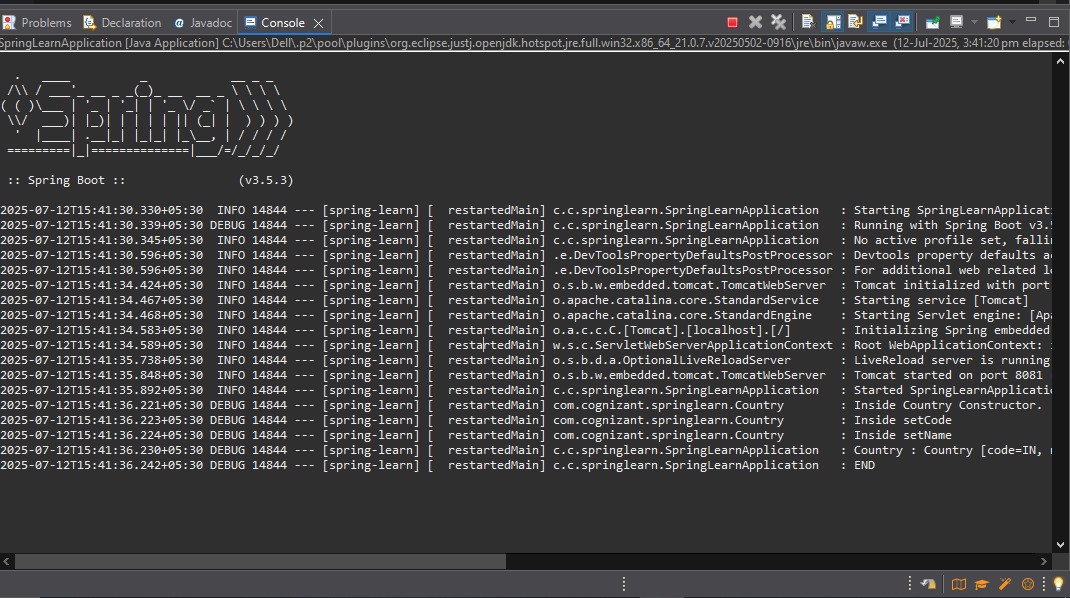
**application.properties:**

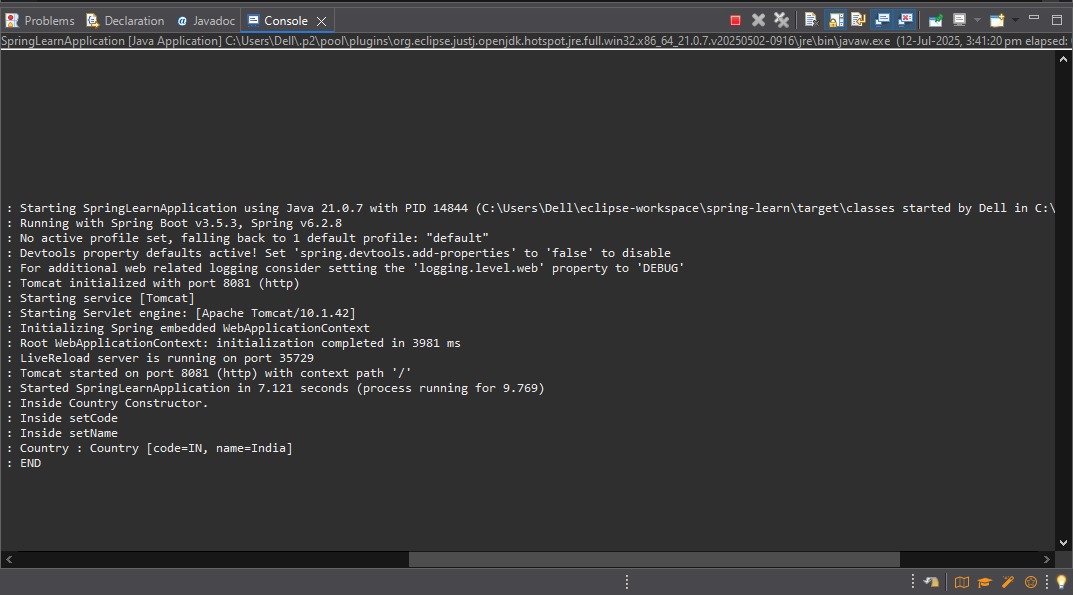
spring.application.name=spring-learn

server.port=8081

logging.level.com.cognizant.springlearn=DEBUG

**Output:**





**Exercise 4:Hello World RESTful Web Service:**

**HelloController class:**

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");

String message = "Hello World!!";

LOGGER.info("END");

return message;

}

}

**SpringLearnApplication:**

package com.cognizant.springlearn;

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) {

LOGGER.debug("START");

SpringApplication.run(SpringLearnApplication.class, args);

displayCountry();

LOGGER.debug("END");

}

public static void displayCountry() {

try (ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("country.xml")) {

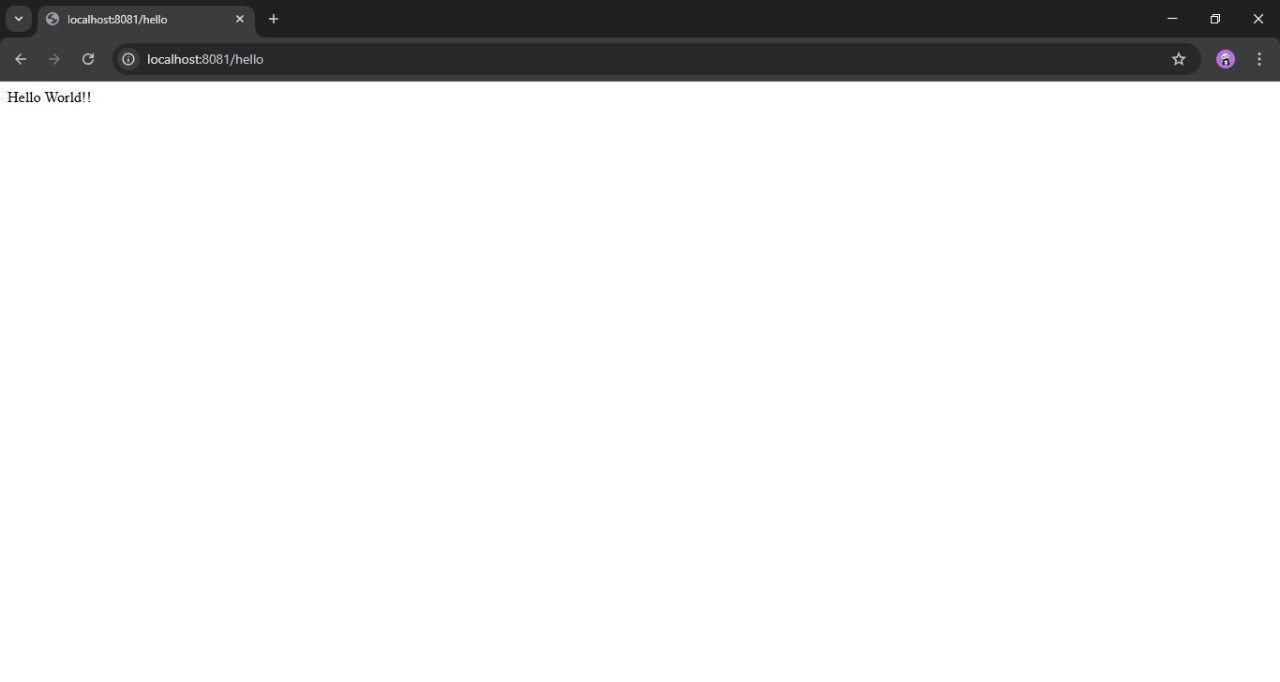
Country country = context.getBean("country", Country.class);

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

}

}

**Output:**



**Exercise 5:** **REST - Country Web Service**

**Country.java**

package com.cognizant.spring\_learn.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;

}

}

**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="in" class="com.cognizant.springlearn.model.Country">

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

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

</bean>

</beans>

**CountryController.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);

}

}

**SpringLearnApplication.java**

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

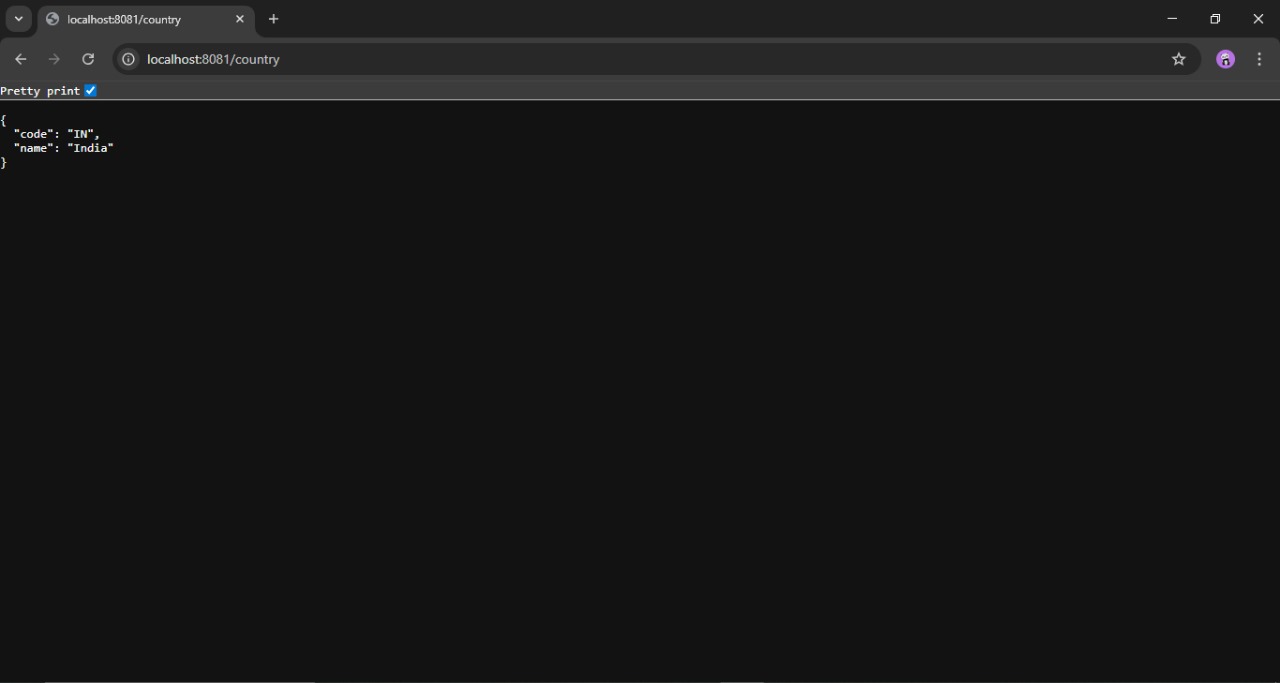
public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Output:**

**Exercise 6:** **REST - Get country based on country code**

**CountryService.java**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

import org.springframework.stereotype.Service;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.List;

@Service

public class CountryService {

public Country getCountry(String code) throws Exception {

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

@SuppressWarnings("unchecked")

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

for (Country country : countries) {

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

return country;

}

}

throw new Exception("Country not found");

}

}

**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.web.bind.annotation.\*;

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) throws Exception {

return countryService.getCountry(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);

}

}

**Country.java**

package com.cognizant.springlearn.model;

public class Country {

private String code;

private String name;

public Country() {

super();

}

public Country(String code, String name) {

super();

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 + "]";

}

}

**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.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="JP"/>

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

</bean>

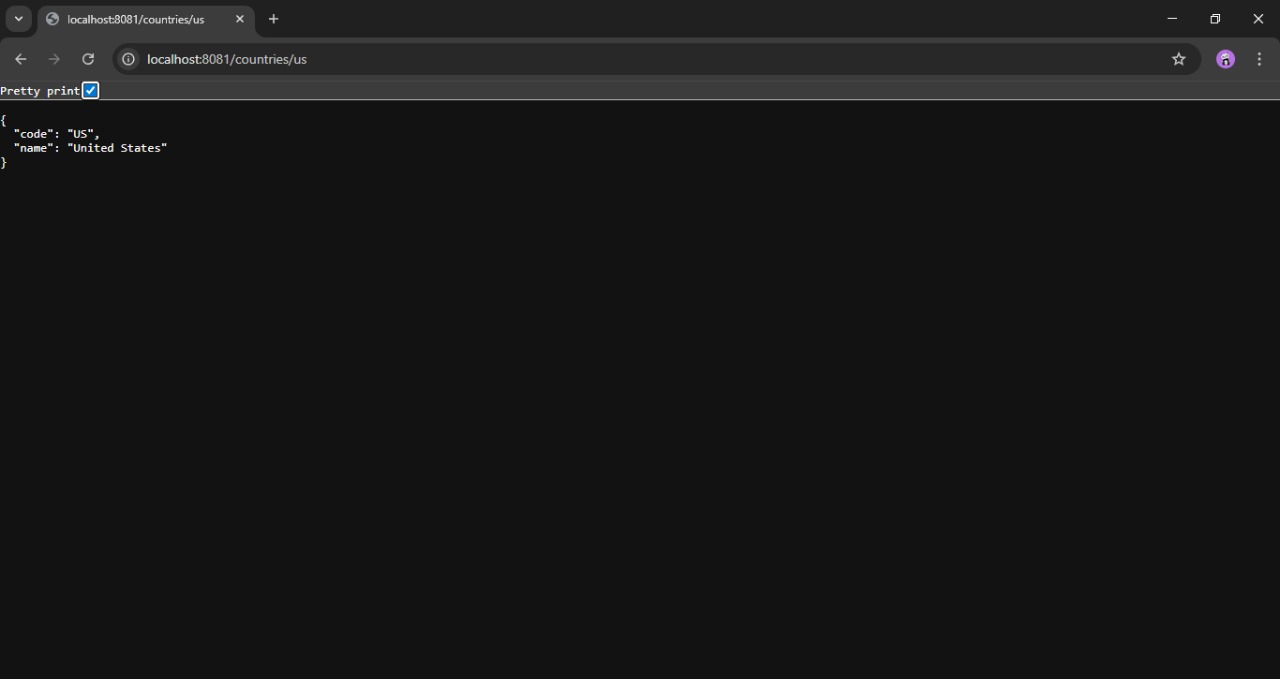
</list>

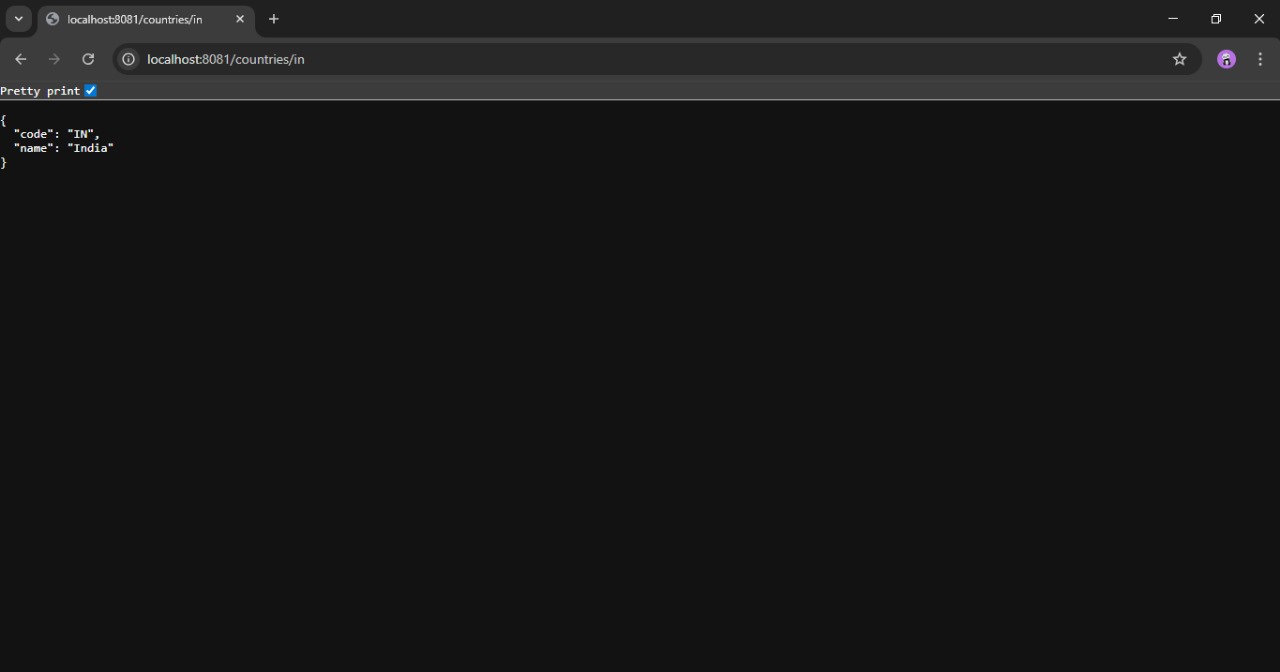
</constructor-arg>

</bean>

</beans>

**Output:**





**Exercise 7: Create authentication service that returns JWT**

**AuthenticationController.java**

package com.friend.spring\_auth.controller;

import com.friend.spring\_auth.util.JwtUtil;

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.\*;

import org.springframework.security.core.Authentication;

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

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

import javax.servlet.http.HttpServletRequest;

import java.util.Base64;

@RestController

public class AuthenticationController {

private final JwtUtil jwtUtil;

private final AuthenticationManager authenticationManager;

public AuthenticationController(JwtUtil jwtUtil, AuthenticationManager authenticationManager) {

this.jwtUtil = jwtUtil;

this.authenticationManager = authenticationManager;

}

@GetMapping("/authenticate")

public ResponseEntity<?> generateJwt(HttpServletRequest request) {

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

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

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

}

String[] creds = extractCredentials(authHeader);

if (creds == null) {

return ResponseEntity.badRequest().body("Malformed base64 credentials");

}

String user = creds[0];

String pwd = creds[1];

try {

Authentication auth = authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(user, pwd)

);

SecurityContextHolder.getContext().setAuthentication(auth);

String jwt = jwtUtil.generateToken(user);

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

} catch (AuthenticationException ex) {

return ResponseEntity.status(401).body("Unauthorized access");

}

}

private String[] extractCredentials(String authHeader) {

try {

String base64 = authHeader.substring(6);

byte[] decoded = Base64.getDecoder().decode(base64);

return new String(decoded).split(":", 2);

} catch (Exception e) {

return null;

}

}

}

**SecurityConfig.java**

package com.friend.spring\_auth.config;

import org.springframework.context.annotation.\*;

import org.springframework.security.authentication.\*;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.\*;

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

import org.springframework.security.crypto.password.\*;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public AuthenticationManager authManager(HttpSecurity http, PasswordEncoder encoder) throws Exception {

return http.getSharedObject(AuthenticationManagerBuilder.class)

.inMemoryAuthentication()

.withUser("friend").password(encoder.encode("pass")).roles("USER")

.and()

.and()

.build();

}

@Bean

public PasswordEncoder encoder() {

return new BCryptPasswordEncoder();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeHttpRequests()

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

.anyRequest().authenticated()

.and()

.httpBasic()

.and()

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

return http.build();

}

}

**JwtUtil.java**

package com.friend.spring\_auth.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private final String SECRET = "my\_secret\_key";

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

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

.setExpiration(new Date(System.currentTimeMillis() + 1000 \* 60 \* 5))

.signWith(SignatureAlgorithm.HS256, SECRET.getBytes())

.compact();

}

}

**Output :**

{

"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJmcmllbmQiLCJpYXQiOjE3Mjg4NzA1MDAsImV4cCI6MTcyODg3MDgwMH0.zcJHKBoFROdxgk7I9\_QKlA\_cRNVZygNDXQoWj2YUJek"

}