**Hands on 1**

**Create a Spring Web Project using Maven**

**SpringLearnApplication.java**

package com.cognizant;

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);

System.out.println("Main method executed - SpringLearnApplication");

}

}

**WelcomeController.java**

package com.cognizant;

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

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

@RestController

public class WelcomeController {

@GetMapping("/")

public String home() {

return "Spring Boot Application is Running Successfully!";

}

}

**SpringLearnApplicationTests.java**

package com.cognizant;

import org.junit.jupiter.api.Test;

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

@SpringBootTest

class SpringLearnApplicationTests {

@Test

void contextLoads() {

}

}

**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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Spring Boot Hands-on 1</description>

<packaging>jar</packaging>

<parent>

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

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

<version>3.2.4</version>

<relativePath/>

</parent>

<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>

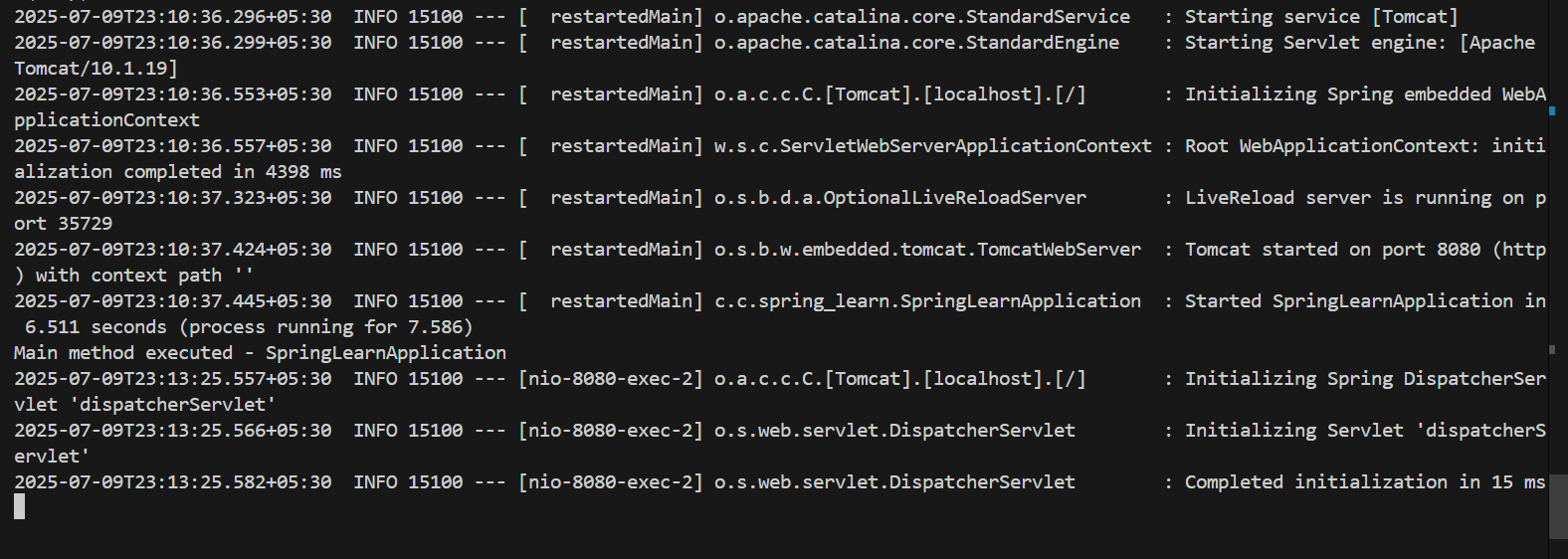
</plugin>

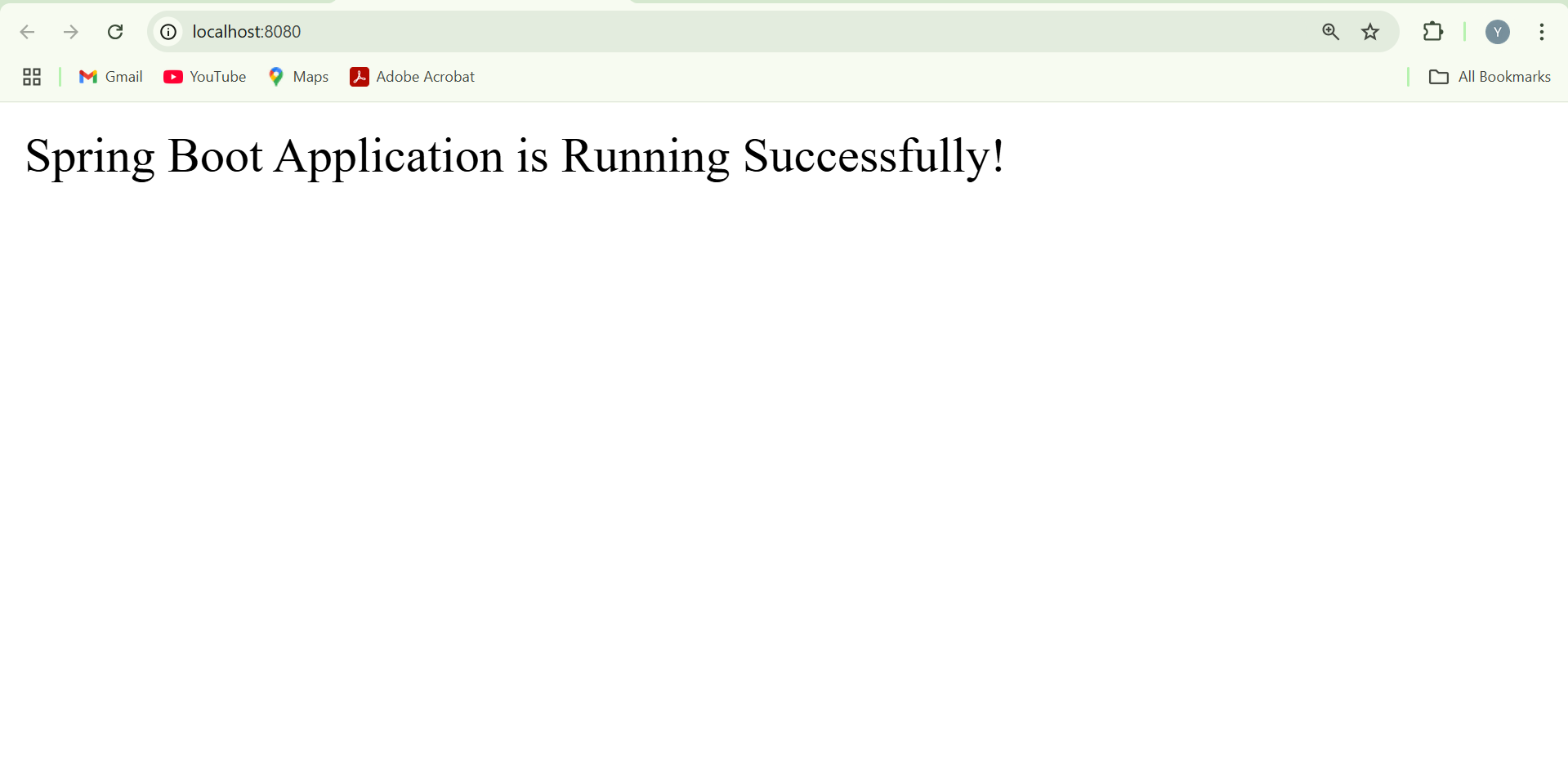
</plugins>

</build>

</project>

**Output:**





**Hands on 2**

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

**SpringLearnApplication.java**

package com.cognizant;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

System.out.println("Main method executed - SpringLearnApplication");

displayDate(); // Call the method

}

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

}

} }

**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>

**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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Spring Boot Hands-on 2</description>

<packaging>jar</packaging>

<parent>

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

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

<version>3.2.4</version>

<relativePath/>

</parent>

<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</groupId>

<artifactId>spring-context</artifactId>

</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>

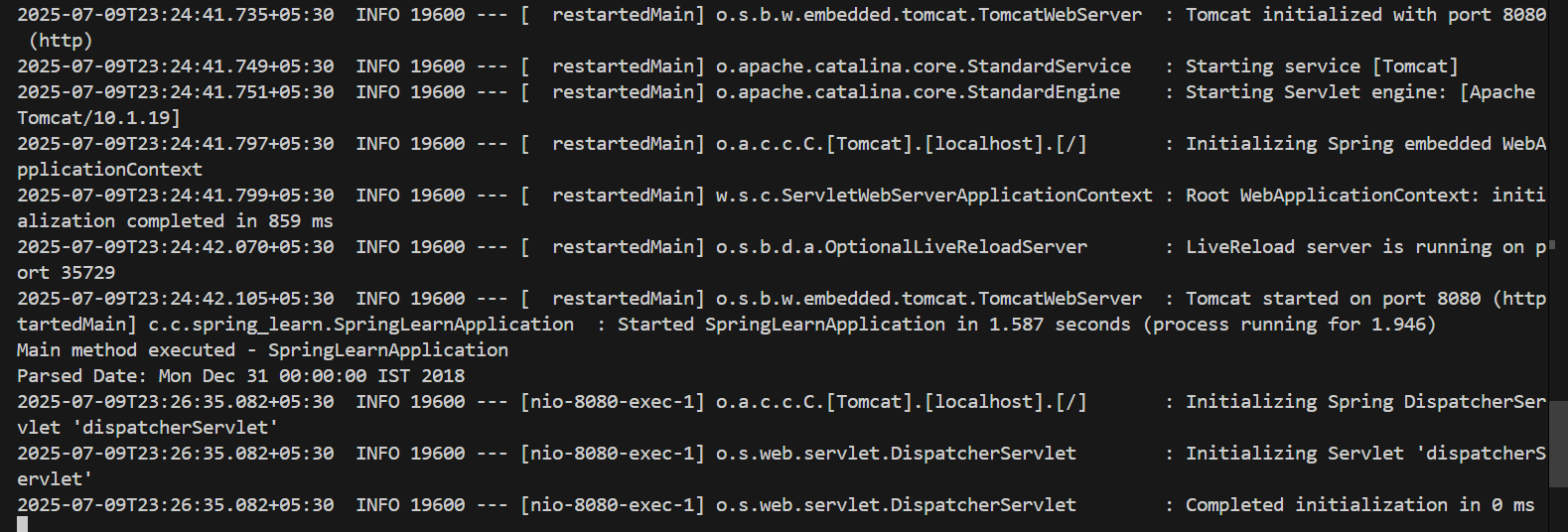
</plugin>

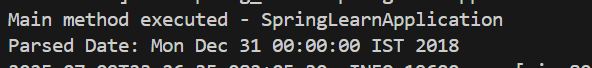
</plugins>

</build>

</project>

**Output:**





**Hands on 3**

**Spring Core - Incorporate Logging**

SpringLearnApplication.java

package com.cognizant;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

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) {

SpringApplication.run(SpringLearnApplication.class, args);

LOGGER.info("START main");

displayDate();

LOGGER.info("END main");

}

public static void displayDate() {

LOGGER.info("START displayDate");

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

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

try {

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

LOGGER.debug("Parsed Date: {}", date.toString());

} catch (ParseException e) {

LOGGER.error("Error parsing date: {}", e.getMessage());

}

LOGGER.info("END displayDate");

}

}

**application.properties**

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.pattern.console=%d{yyMMdd}|%d{HH:mm:ss.SSS}|%-20.20thread|%5p|%-25.25logger{25}|%25M|%m%n

**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>

**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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-learn</name>

<description>Spring Boot Hands-on 1 to 3</description>

<parent>

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

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

<version>3.2.4</version>

<relativePath/>

</parent>

<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</groupId>

<artifactId>spring-context</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>

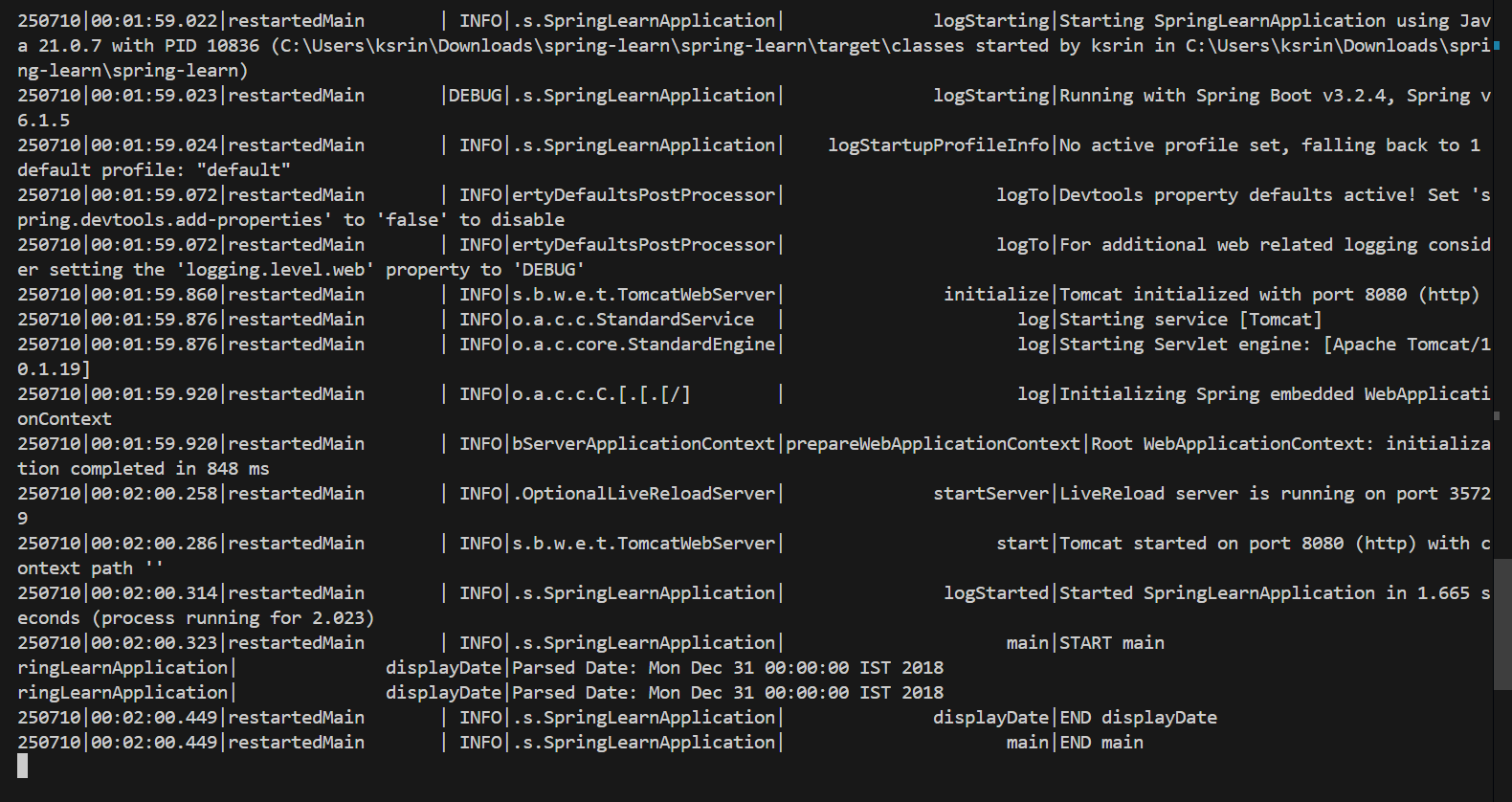
</plugin>

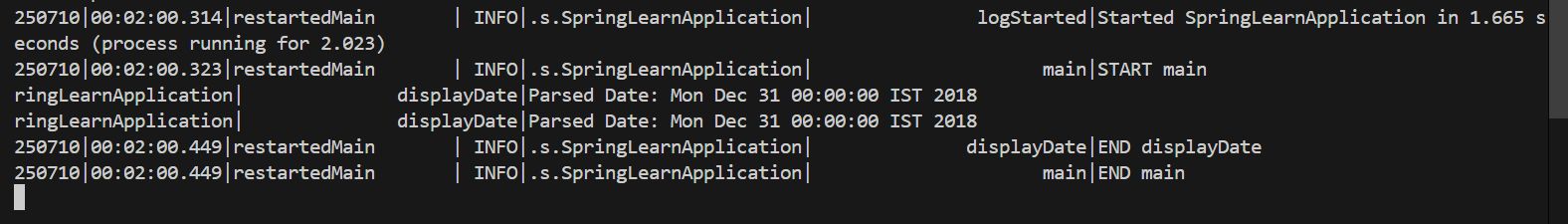
</plugins>

</build>

</project>

**Output:**





**Hands on 4**

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

**Country.java**

package com.cognizant.spring\_learn;

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 + "]";

}

}

**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.spring\_learn.Country">

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

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

</bean>

</beans>

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

public static void main(String[] args) {

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

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

System.out.println(country); // Output: Country [code=IN, name=India]

}

}

**pom.xml**

<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-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-learn</name>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.5</version>

</dependency>

<dependency>

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

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

<version>3.2.4</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>21</source>

<target>21</target>

</configuration>

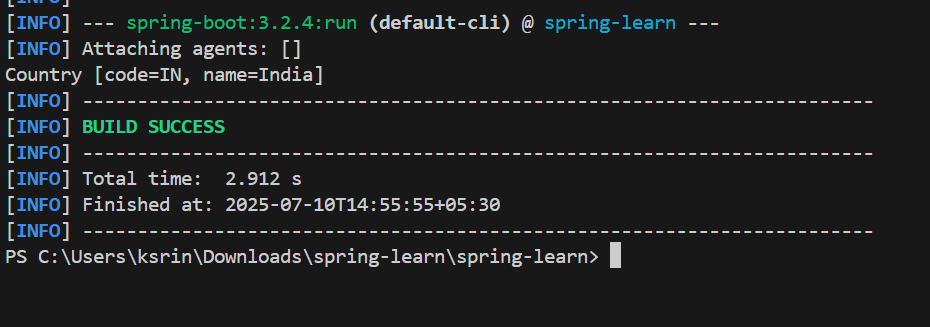
</plugin>

</plugins>

</build>

</project>

**Output:**



**Hands on 5**

**Spring Core – Demonstration of Singleton Scope and Prototype Scope**

**Country.java**

package com.cognizant.springlearn;

public class Country {

private String code;

private String name;

public Country() {

System.out.println("Country Constructor Called");

}

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;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

displayCountry();

}

public static void displayCountry() {

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

Country country1 = context.getBean("country", Country.class);

Country country2 = context.getBean("country", Country.class);

System.out.println("Country 1: " + country1);

System.out.println("Country 2: " + country2);

System.out.println("Are both countries same object? " + (country1 == country2));

}

}

**country.xml**

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

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

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

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

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

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

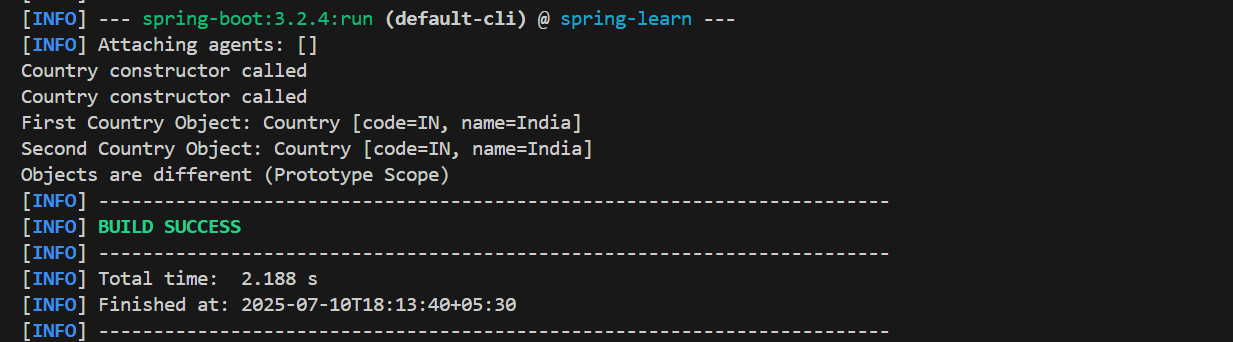
<property name="code" value="IN"/>

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

</bean>

</beans>

**Output:**



**Hands on 6**

**Spring Core – Load list of countries from Spring Configuration XML**

**Country.java**

package com.cognizant.spring\_learn;

public class Country {

private String code;

private String name;

public Country() {

System.out.println("Country Constructor Called");

}

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.spring\_learn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

displayCountries();

}

public static void displayCountries() {

LOGGER.info("START");

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

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

LOGGER.debug("---- Country List ----");

for (Country country : countryList) {

LOGGER.debug("Code: {} | Name: {}", country.getCode(), country.getName());

}

LOGGER.debug("----------------------");

LOGGER.info("END");

}

}

**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.spring\_learn.Country">

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

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

</bean>

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

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

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

</bean>

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

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

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

</bean>

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

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

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

</bean>

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

<constructor-arg>

<list>

<ref bean="in"/>

<ref bean="us"/>

<ref bean="de"/>

<ref bean="jp"/>

</list>

</constructor-arg>

</bean>

</beans>

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<parent>

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

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

<version>3.2.4</version>

</parent>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

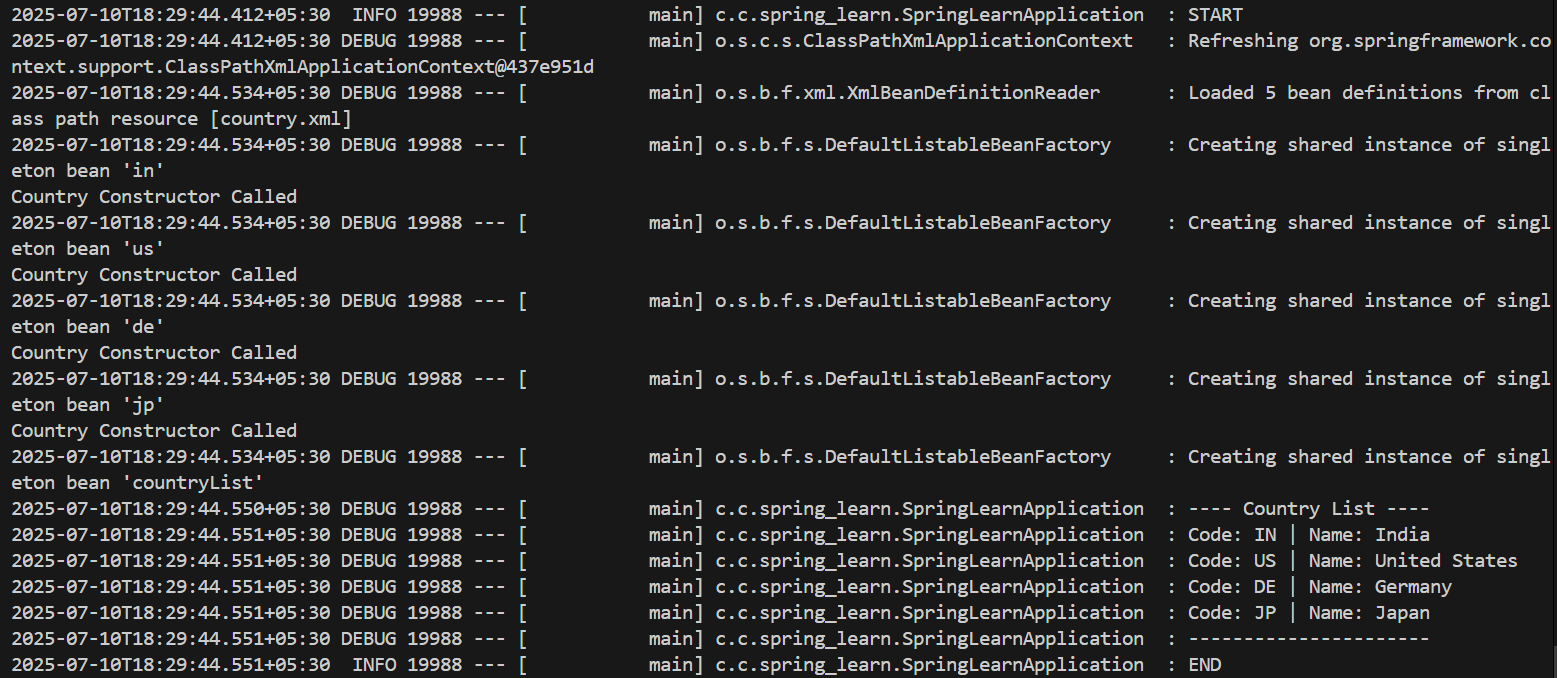
</plugin>

</plugins>

</build>

</project>

**Output:**



**HTTP Request Response**   
**pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>http-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>http-demo</name>

<description>Spring Boot HTTP Request Response Demo</description>

<packaging>jar</packaging>

<parent>

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

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

<version>3.2.4</version>

</parent>

<properties>

<java.version>21</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-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>3.2.4</version>

</plugin>

</plugins>

</build>

</project>

**HttpDemoApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication

@SpringBootApplication

public class HttpDemoApplication {

public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**DemoController.java**

package com.example.http\_demo;

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

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

@RestController

public class DemoController {

@GetMapping("/demo")

public String showHttpDemo() {

return "Hello! This is the HTTP Demo Response.";

}

}

**application.properties**

server.port=8080

**HttpDemoApplicationTests.java**

package com.example.http\_demo;

import org.junit.jupiter.api.Test;

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

@SpringBootTest

class HttpDemoApplicationTests {

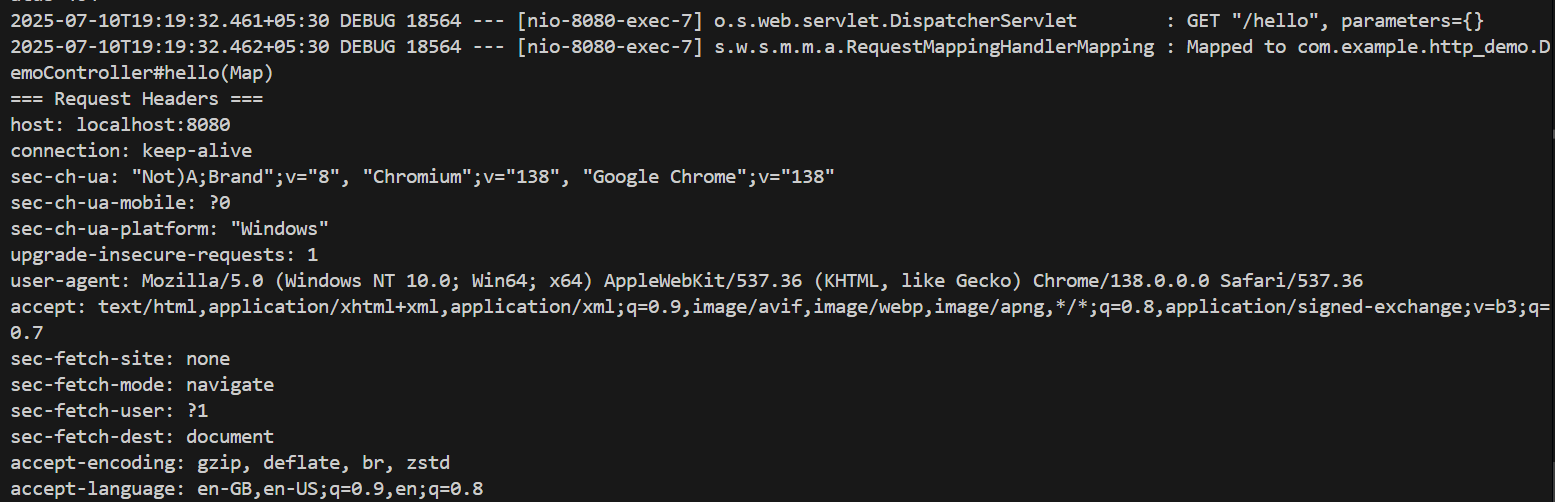
@Test

void contextLoads() {

}

}

**Output:**





**Hello World RESTful Web Service**

**pom.xml**

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

**HelloController.java**

package com.example.http-demo;

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 response = "Hello World!!";

LOGGER.info("END");

return response;

}

}

**application.properties**

server.port=8083

logging.level.root=DEBUG

**HttpDemoApplication.java**

package com.example.http-demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

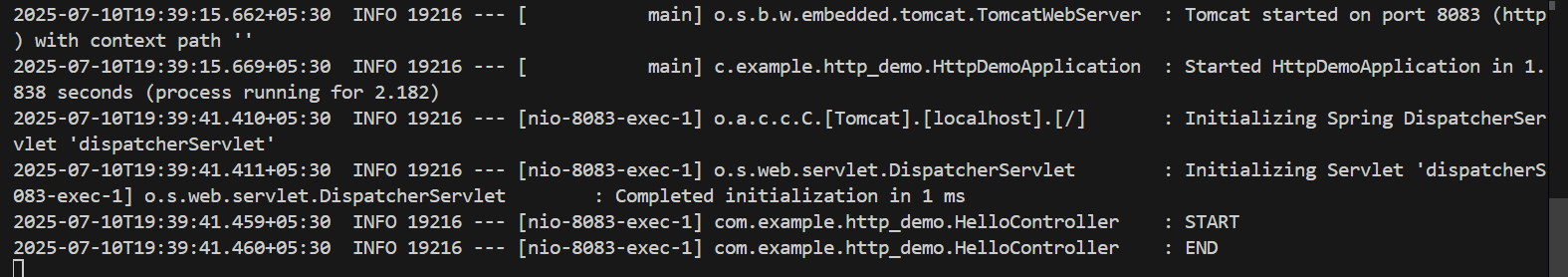
public static void main(String[] args) {

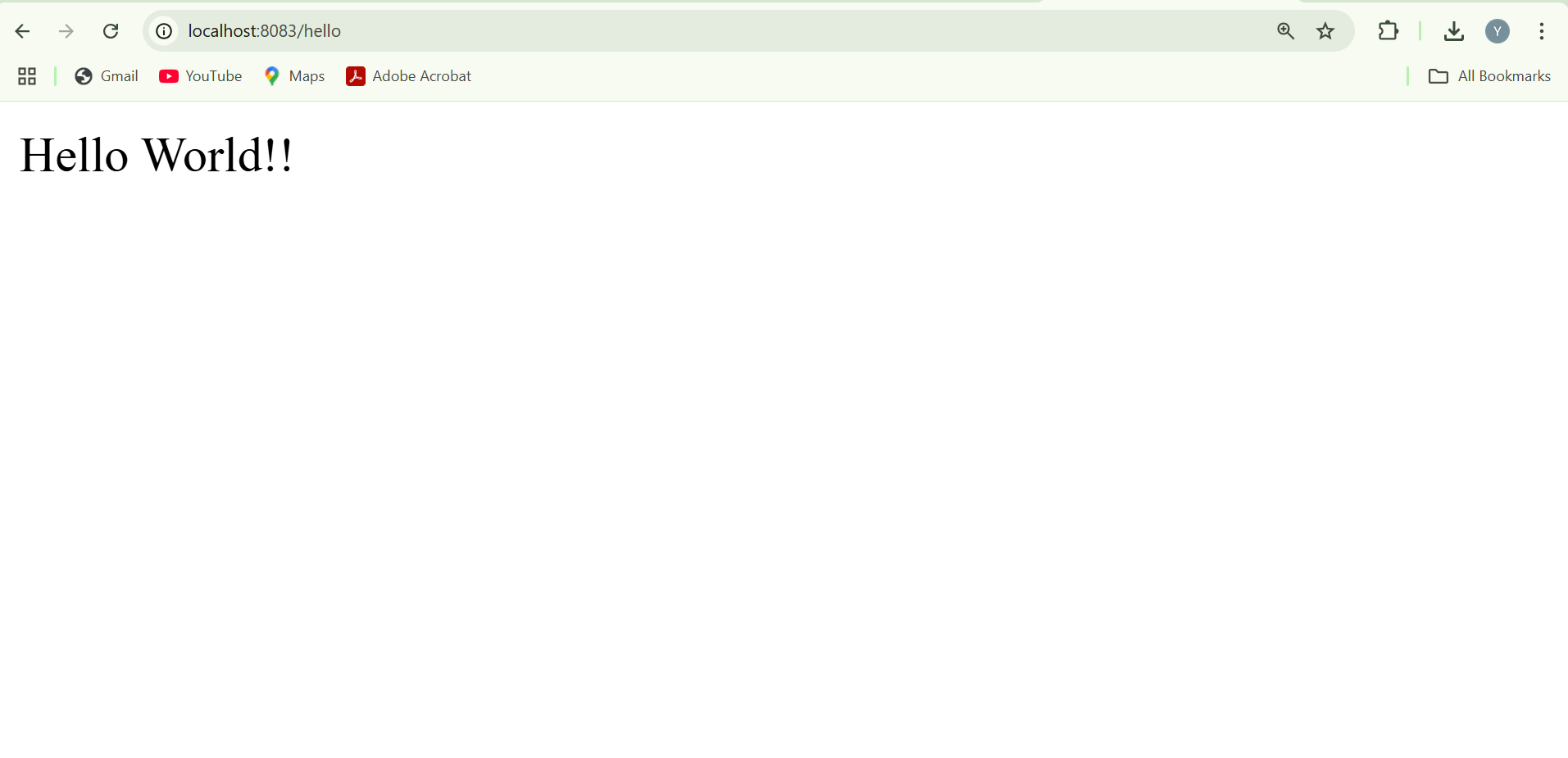
SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Output:**





**REST - Country Web Service**

**SpringLearnApplication.java**

package com.example.http\_demo;

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.example.http\_demo;

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;

}

}

**CountryController.java**

package com.example.http\_demo;

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

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

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

@RestController

public class CountryController {

@RequestMapping("/country")

public Country getCountryIndia() {

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

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

return 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

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

<bean id="in" class="com.example.http\_demo.Country">

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

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

</bean>

</beans>

**pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>http-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>http-demo</name>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

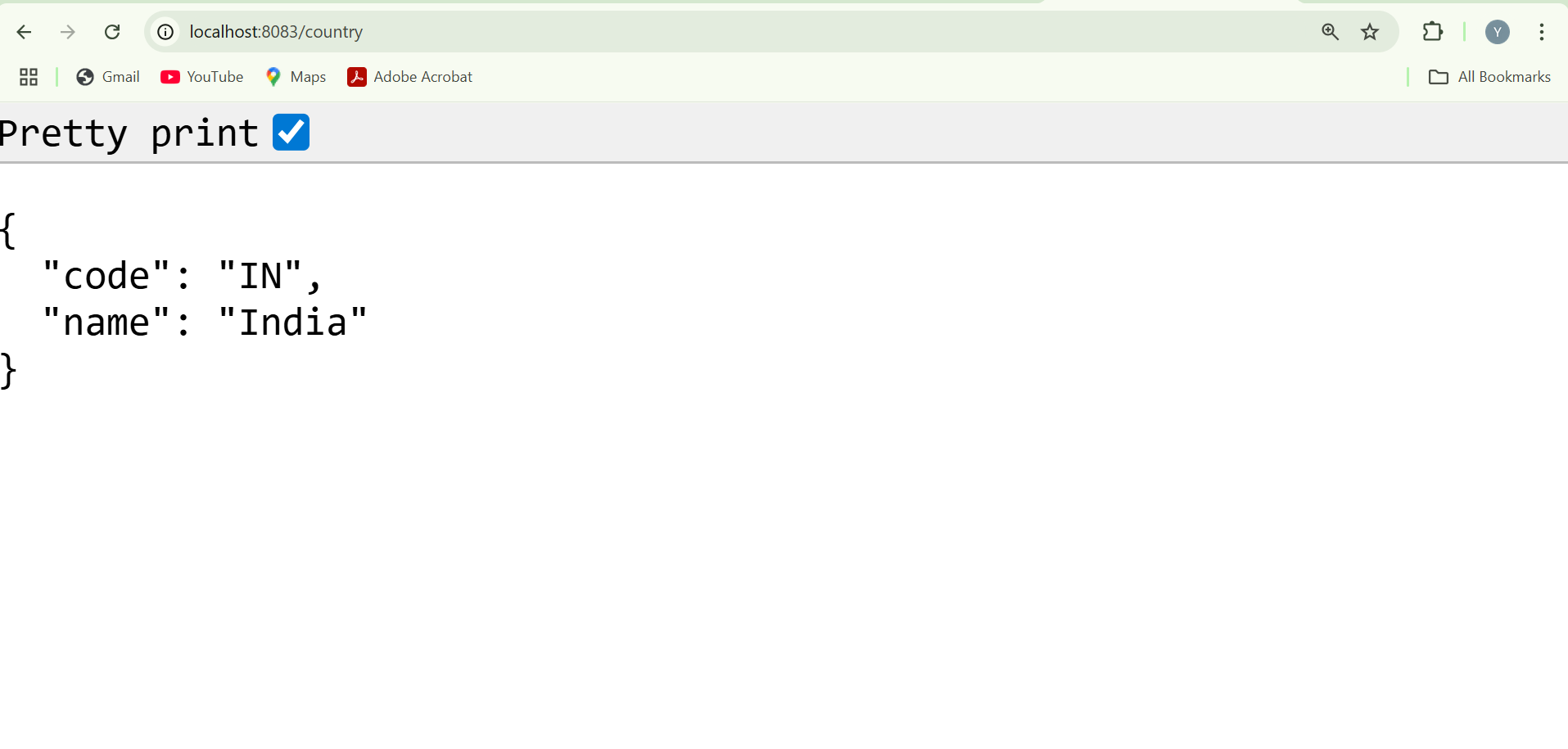
</plugins>

</build>

</project>

**Output:**





**REST - Get all countries**

**Country.java**

package com.example.http\_demo;

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;

}

}

**CountryController.java**

package com.example.http\_demo;

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;

import java.util.List;

@RestController

public class CountryController {

@GetMapping("/countries")

public List<Country> getAllCountries() {

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

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

return countries;

}

}

**SpringLearnApplication.java**

package com.example.http\_demo;

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.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="country1" class="com.example.http\_demo.Country">

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

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

</bean>

<bean id="country2" class="com.example.http\_demo.Country">

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

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

</bean>

<bean id="country3" class="com.example.http\_demo.Country">

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

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

</bean>

<bean id="country4" class="com.example.http\_demo.Country">

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

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

</bean>

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

<constructor-arg>

<list>

<ref bean="country1"/>

<ref bean="country2"/>

<ref bean="country3"/>

<ref bean="country4"/>

</list>

</constructor-arg>

</bean>

</beans>

**application.properties**

server.port=8083

**pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>http\_demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>http\_demo</name>

<parent>

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

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

<version>3.2.4</version>

<relativePath/>

</parent>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.4</version>

</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>

<version>3.2.4</version>

</plugin>

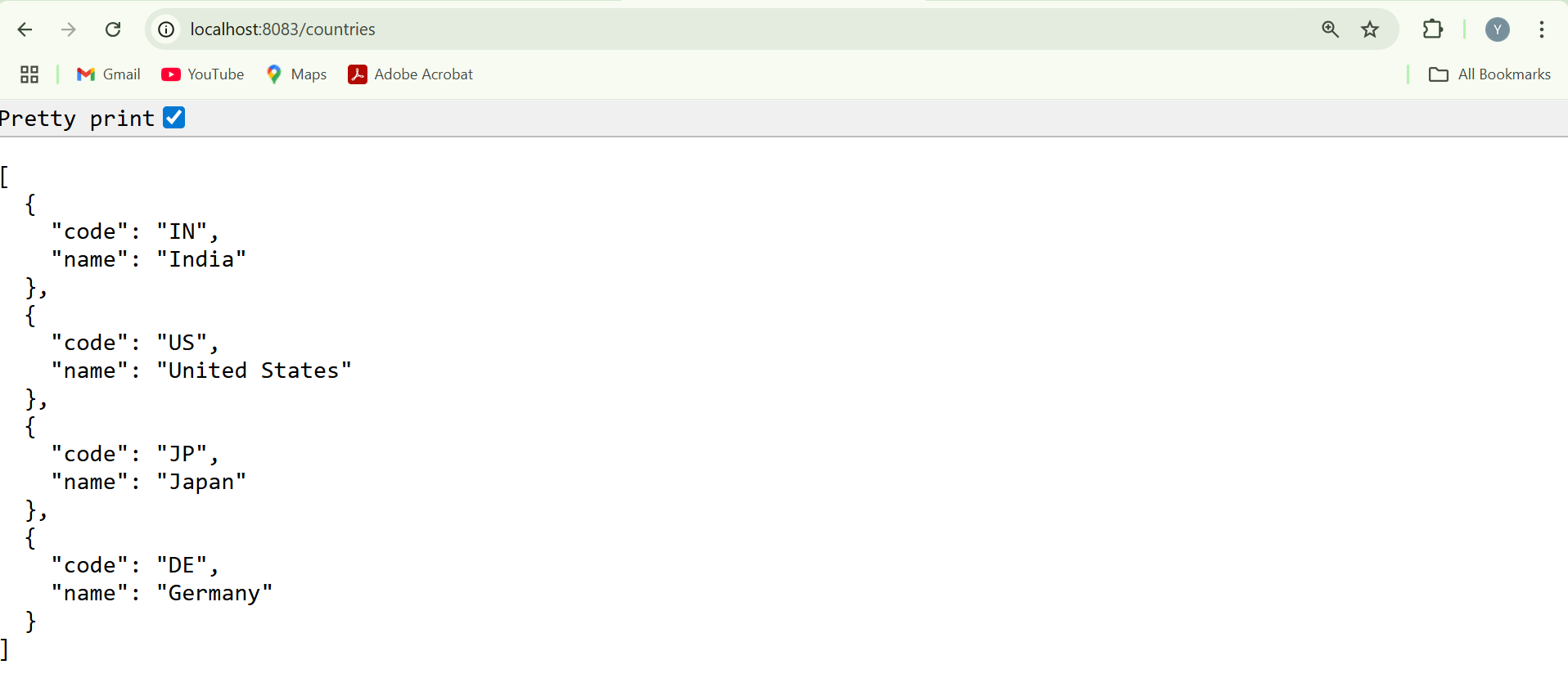
</plugins>

</build>

</project>

**Output:**





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

**Country.java**

package com.example.http\_demo;

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;

}

}

**CountryService.java**

package com.example.http\_demo;

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

return countries.stream()

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

.findFirst()

.orElse(null); // You can throw custom exception here

}

}

**CountryController.java**

package com.example.http\_demo;

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

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

import java.util.List;

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping("/countries")

public List<Country> getAllCountries() {

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

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

return countries;

}

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

public Country getCountry(@PathVariable String code) {

return countryService.getCountry(code);

}

}

**SpringLearnApplication.java**

package com.example.http\_demo;

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.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="country1" class="com.example.http\_demo.Country">

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

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

</bean>

<bean id="country2" class="com.example.http\_demo.Country">

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

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

</bean>

<bean id="country3" class="com.example.http\_demo.Country">

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

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

</bean>

<bean id="country4" class="com.example.http\_demo.Country">

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

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

</bean>

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

<constructor-arg>

<list>

<ref bean="country1"/>

<ref bean="country2"/>

<ref bean="country3"/>

<ref bean="country4"/>

</list>

</constructor-arg>

</bean>

</beans>

**application.properties**

server.port=8083

**pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>http\_demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>http\_demo</name>

<parent>

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

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

<version>3.2.4</version>

<relativePath/>

</parent>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.4</version>

</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>

<version>3.2.4</version>

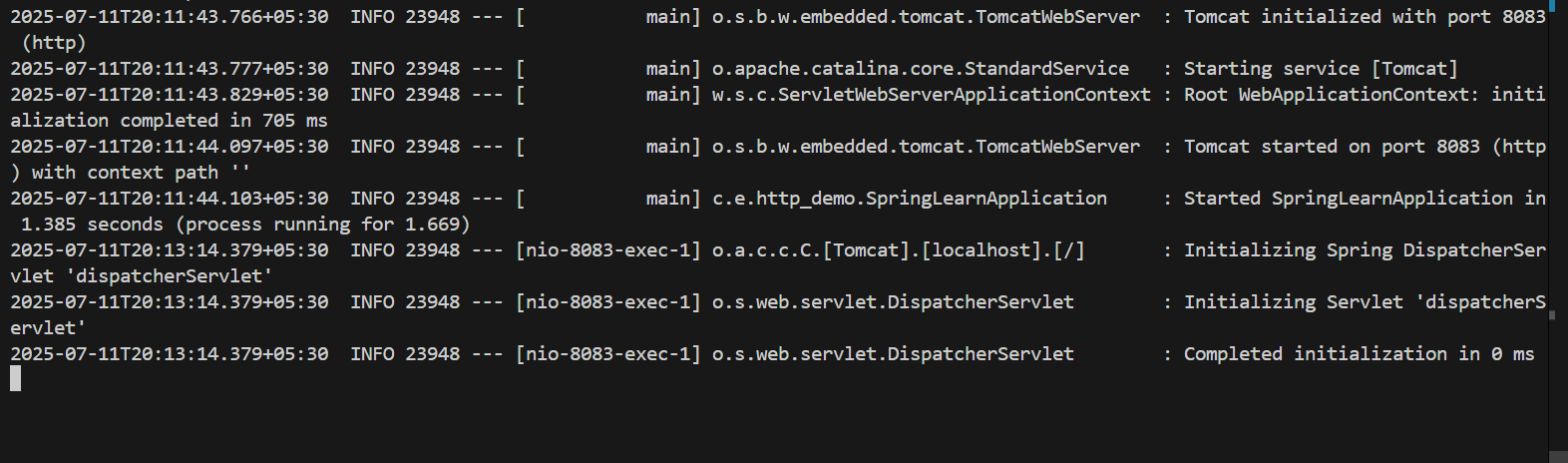
</plugin>

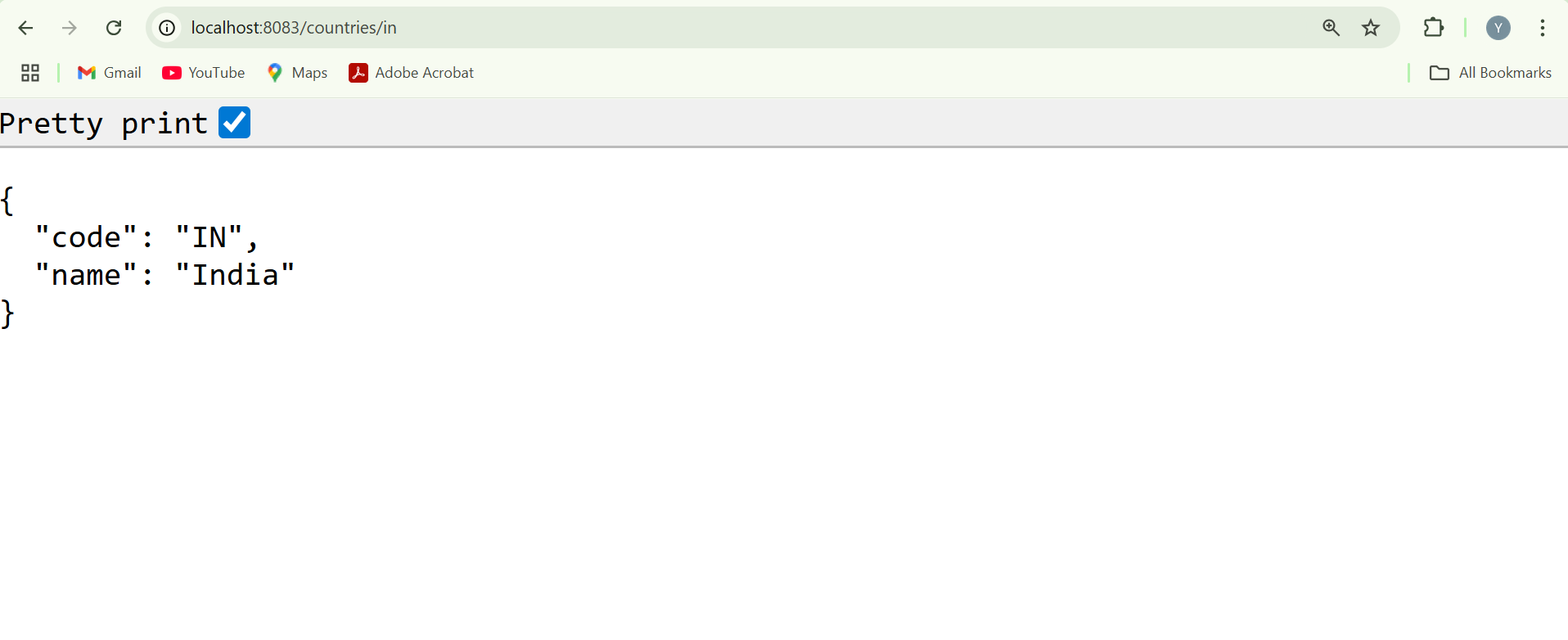
</plugins>

</build>

</project>

**Output:**





**REST - Get country exceptional scenario**   
  
**Country.java**

package com.example.http\_demo;

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;

}

}

**CountryNotFoundException.java**

package com.example.http\_demo;

public class CountryNotFoundException extends RuntimeException {

public CountryNotFoundException() {

super("Country not found");

}

}

**CountryService.java**

package com.example.http\_demo;

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

return countries.stream()

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

.findFirst()

.orElseThrow(CountryNotFoundException::new);

}

}

**CountryController.java**

package com.example.http\_demo;

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

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

import java.util.List;

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping("/countries")

public List<Country> getAllCountries() {

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

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

return countries;

}

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

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

return countryService.getCountry(code);

}

}

**SpringLearnApplication.java**

package com.example.http\_demo;

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.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="country1" class="com.example.http\_demo.Country">

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

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

</bean>

<bean id="country2" class="com.example.http\_demo.Country">

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

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

</bean>

<bean id="country3" class="com.example.http\_demo.Country">

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

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

</bean>

<bean id="country4" class="com.example.http\_demo.Country">

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

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

</bean>

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

<constructor-arg>

<list>

<ref bean="country1"/>

<ref bean="country2"/>

<ref bean="country3"/>

<ref bean="country4"/>

</list>

</constructor-arg>

</bean>

</beans>

**application.properties**

server.port=8083

**pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>http\_demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>http\_demo</name>

<parent>

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

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

<version>3.2.4</version>

<relativePath/>

</parent>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>6.1.4</version>

</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>

<version>3.2.4</version>

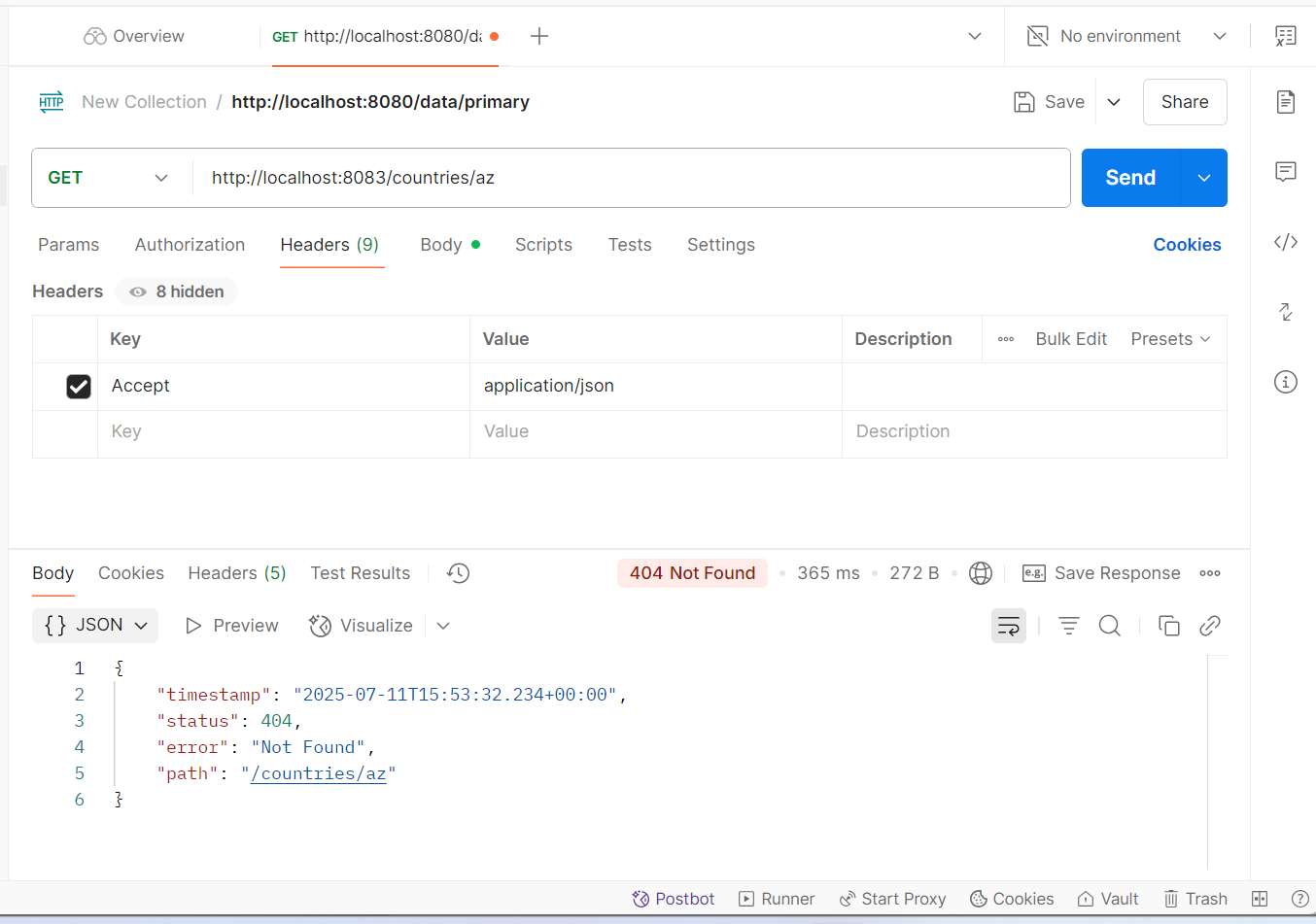
</plugin>

</plugins>

</build>

</project>

**Output:**



**MockMVC - Test get country service**

**SpringLearnApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ImportResource;

@SpringBootApplication

@ImportResource("classpath:country.xml")

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}}

**Country.java**

package com.example.http\_demo.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;

}

}

**CountryController.java**

package com.example.http\_demo.controller;

import com.example.http\_demo.model.Country;

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

import org.springframework.context.ApplicationContext;

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

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

@RestController

public class CountryController {

@Autowired

private ApplicationContext context;

public CountryController() {

System.out.println("CountryController constructor called");

}

@RequestMapping("/country")

public Country getCountryIndia() {

return (Country) context.getBean("in");

}

}

**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="in" class="com.example.http\_demo.model.Country">

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

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

</bean>

</beans>

**SpringLearnApplicationTests.java**

package com.example.http\_demo;

import com.example.http\_demo.controller.CountryController;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

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

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.ResultActions;

import static org.junit.jupiter.api.Assertions.assertNotNull;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;

@SpringBootTest

@AutoConfigureMockMvc

public class SpringLearnApplicationTests {

@Autowired

private CountryController countryController;

@Autowired

private MockMvc mvc;

@Test

public void contextLoads() {

assertNotNull(countryController);

}

@Test

public void testGetCountry() throws Exception {

ResultActions actions = mvc.perform(get("/country"));

actions.andExpect(status().isOk());

actions.andExpect(jsonPath("$.code").exists());

actions.andExpect(jsonPath("$.code").value("IN"));

actions.andExpect(jsonPath("$.name").exists());

actions.andExpect(jsonPath("$.name").value("India"));

}}

**pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>http\_demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>http\_demo</name>

<description>Spring Boot MockMVC Test Country Example</description>

<packaging>jar</packaging>

<parent>

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

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

<version>3.2.3</version>

<relativePath/>

</parent>

<properties>

<java.version>21</java.version>

<maven.compiler.source>${java.version}</maven.compiler.source>

<maven.compiler.target>${java.version}</maven.compiler.target>

</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-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<version>3.2.3</version>

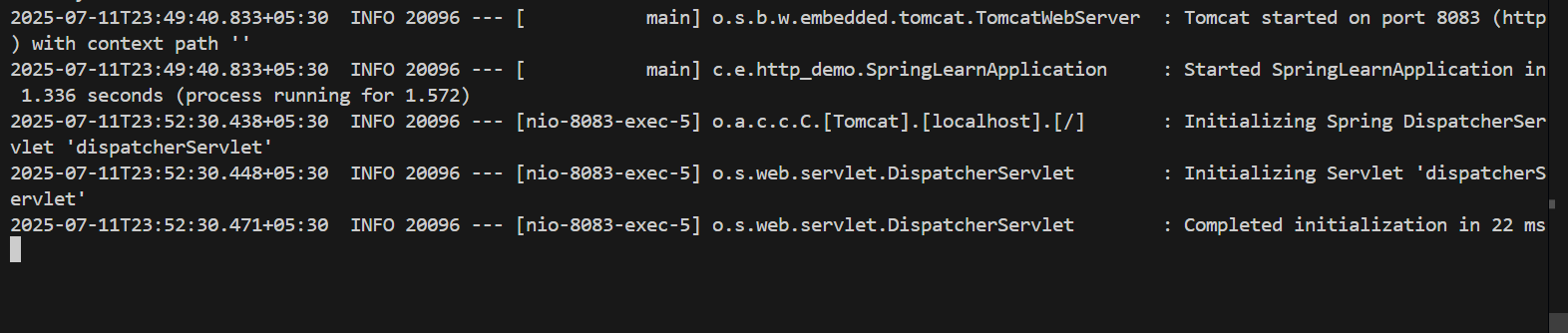
</plugin>

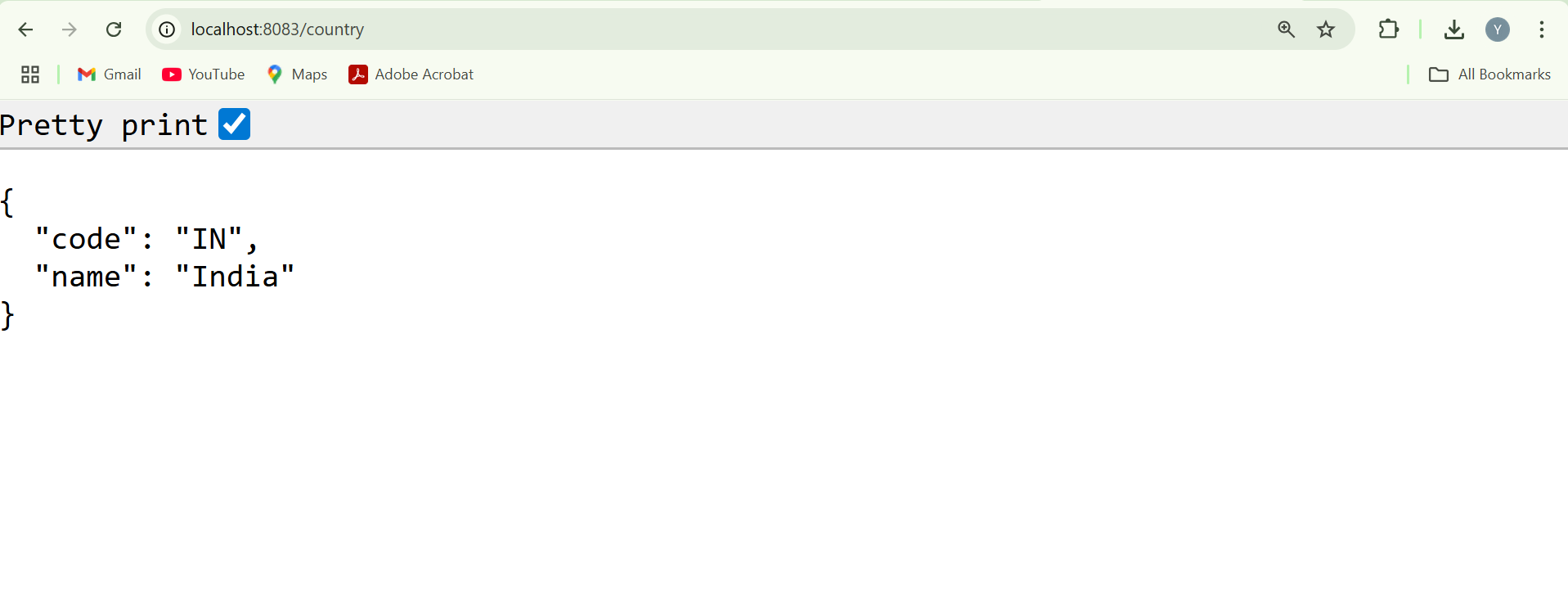
</plugins>

</build>

</project>

**Output:**





**MockMVC - Test get country service for exceptional scenario**

**SpringLearnApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.ImportResource;

@SpringBootApplication

@ImportResource("classpath:country.xml")

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Country.java**

package com.example.http\_demo;

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;

}

}

**CountryController.java**

package com.example.http\_demo;

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

import org.springframework.context.ApplicationContext;

import org.springframework.http.HttpStatus;

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

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

import org.springframework.web.server.ResponseStatusException;

@RestController

public class CountryController {

@Autowired

private ApplicationContext context;

public CountryController() {

System.out.println("CountryController constructor called");

}

@RequestMapping("/country")

public Country getCountryIndia() {

try {

return (Country) context.getBean("in");

} catch (Exception e) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST, "Country Not found"); } } }

**country.xml**

<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="in" class="com.example.http\_demo.Country">

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

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

</bean>

</beans>

**SpringLearnApplicationTests.java**

package com.example.http\_demo;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

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

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.ResultActions;

import static org.junit.jupiter.api.Assertions.assertNotNull;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;

@SpringBootTest

@AutoConfigureMockMvc

public class SpringLearnApplicationTests {

@Autowired

private CountryController countryController;

@Autowired

private MockMvc mvc;

@Test

public void contextLoads() {

assertNotNull(countryController);

}

@Test

public void testGetCountrySuccess() throws Exception {

ResultActions actions = mvc.perform(get("/country"));

actions.andExpect(status().isOk());

actions.andExpect(jsonPath("$.code").exists());

actions.andExpect(jsonPath("$.code").value("IN"));

actions.andExpect(jsonPath("$.name").exists());

actions.andExpect(jsonPath("$.name").value("India"));

}

@Test

public void testGetCountryException() throws Exception {

ResultActions actions = mvc.perform(get("/country"));

actions.andExpect(status().isBadRequest());

actions.andExpect(status().reason("Country Not found"));

}

}

**pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>http\_demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>http\_demo</name>

<parent>

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

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

<version>3.2.3</version>

<relativePath/>

</parent>

<properties>

<java.version>21</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-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

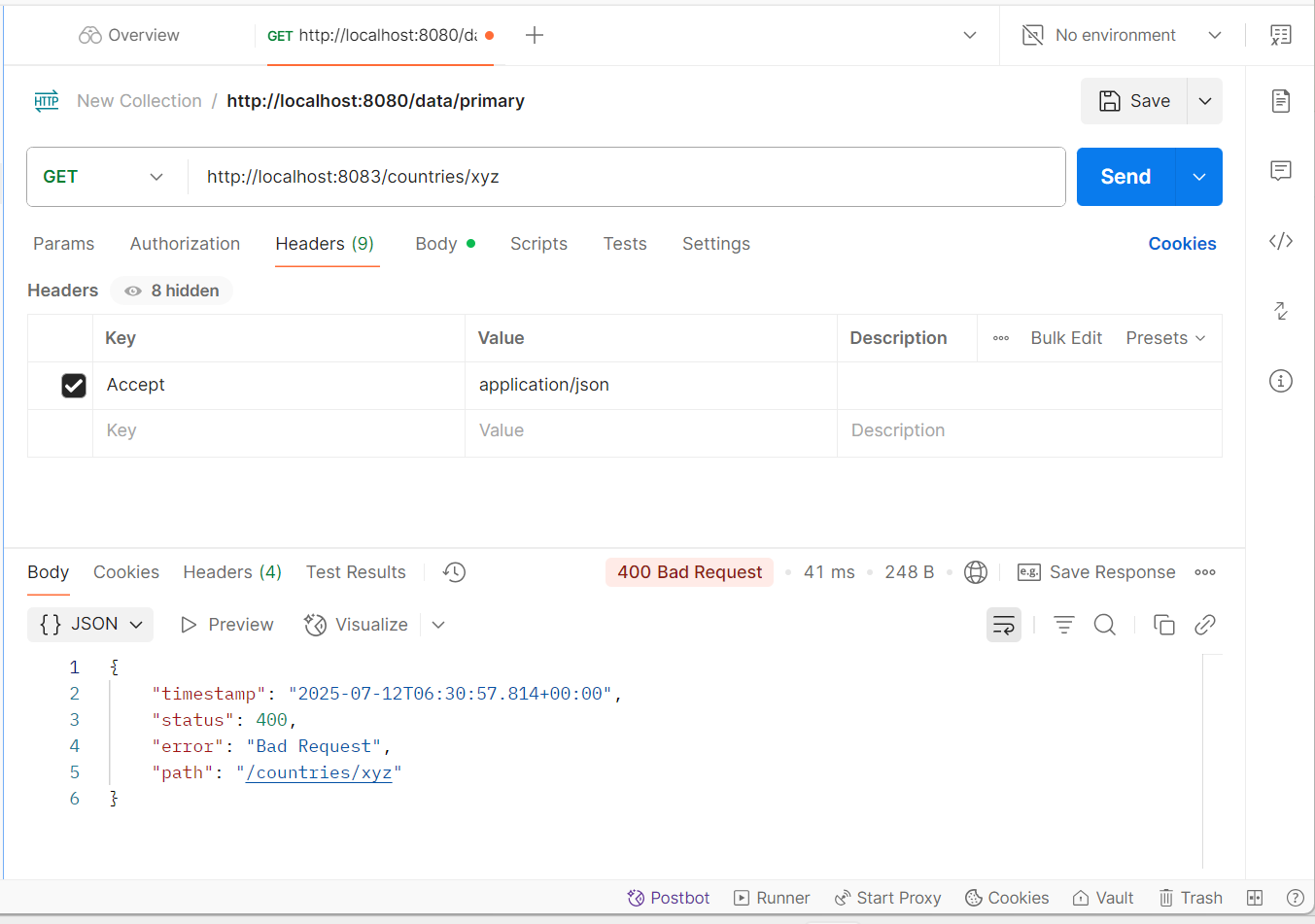
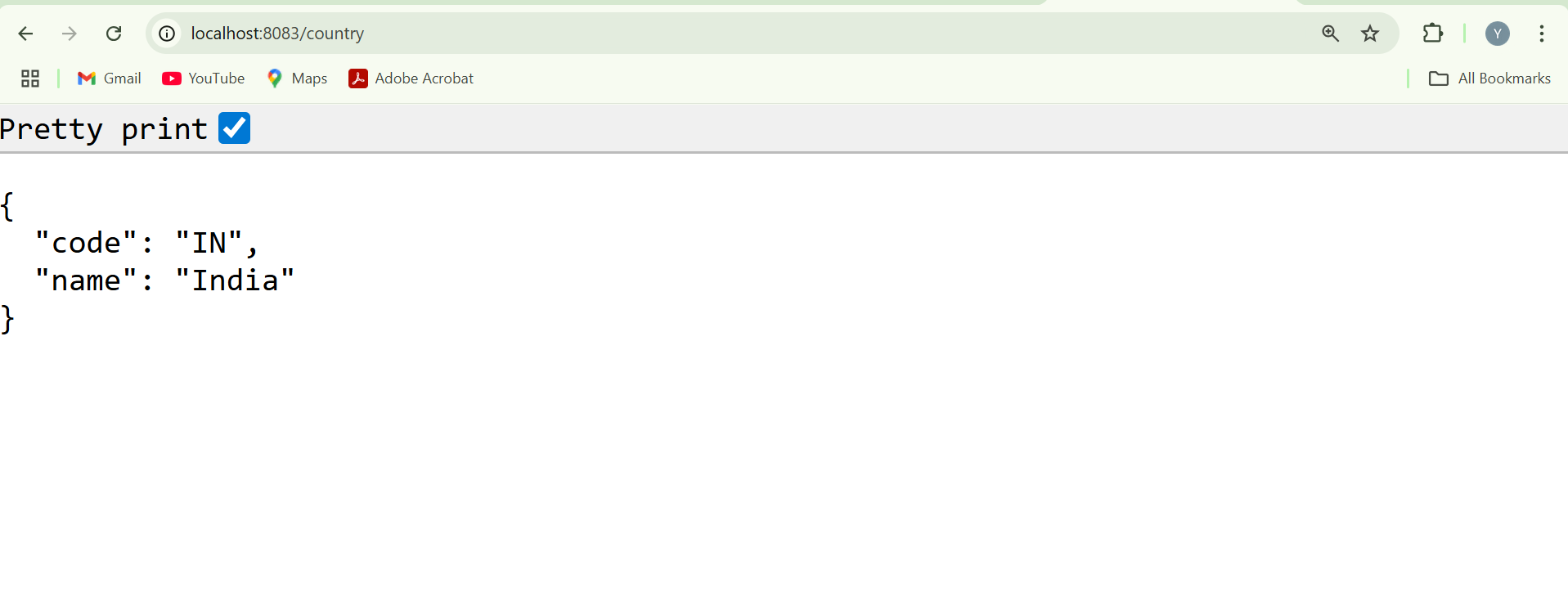
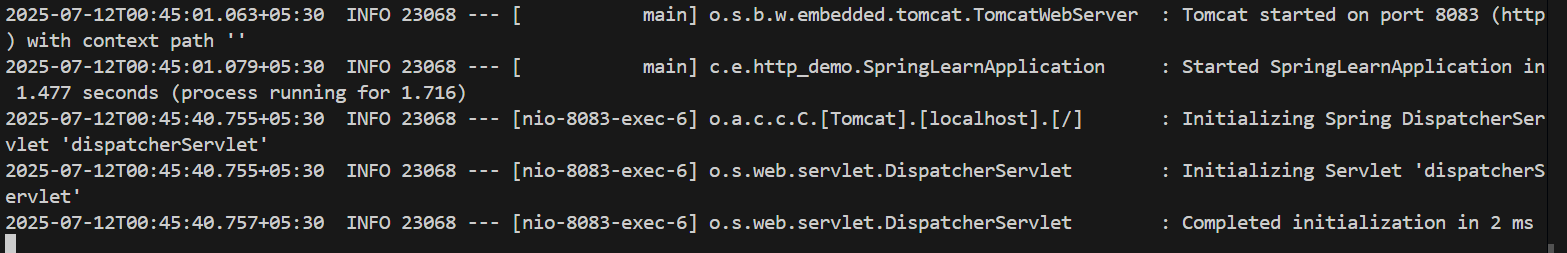
<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**Output:**

**Problem Statement - Display Employee List and Edit Employee form using RESTful Web Service**

**Country.java**

package com.example.http\_demo;

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;

} }

**CountryNotFoundException.java**

package com.example.http\_demo;

public class CountryNotFoundException extends RuntimeException {

public CountryNotFoundException(String message) {

super(message); } }

**CountryController.java**

package com.example.http\_demo;

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

@RestController

public class CountryController {

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

public Country getCountry(@PathVariable String code) {

if (!code.equalsIgnoreCase("in")) {

throw new CountryNotFoundException("Country Not found");

}

return new Country("in", "India");

}

}

**GlobalExceptionHandler.java**

package com.example.http\_demo;

import org.springframework.http.HttpStatus;

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

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

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(CountryNotFoundException.class)

public void handleCountryNotFound(HttpServletResponse response) throws IOException {

response.sendError(HttpStatus.BAD\_REQUEST.value(), "Country Not found"); } }

**SpringLearnApplication.java**

package com.example.http\_demo;

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);

}

}

**pom.xml**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

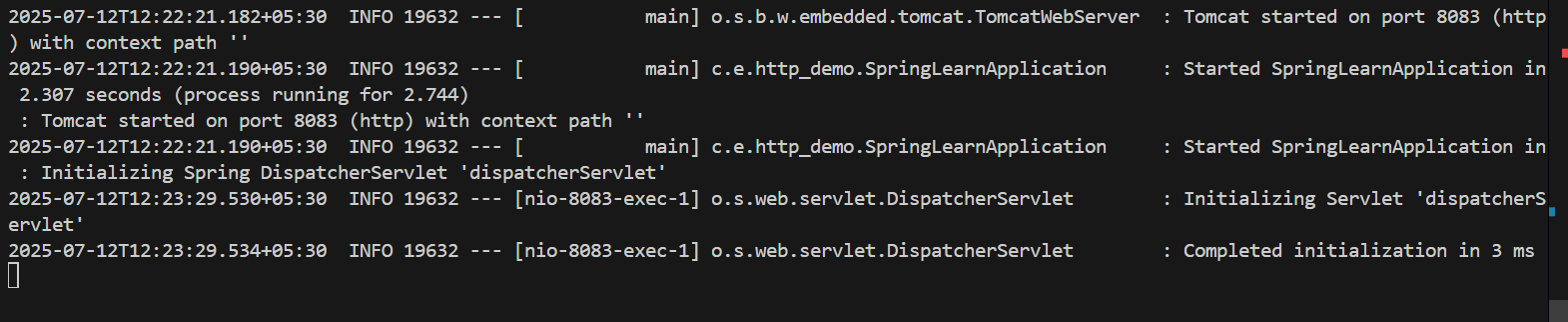
<artifactId>spring-boot-starter-test</artifactId>

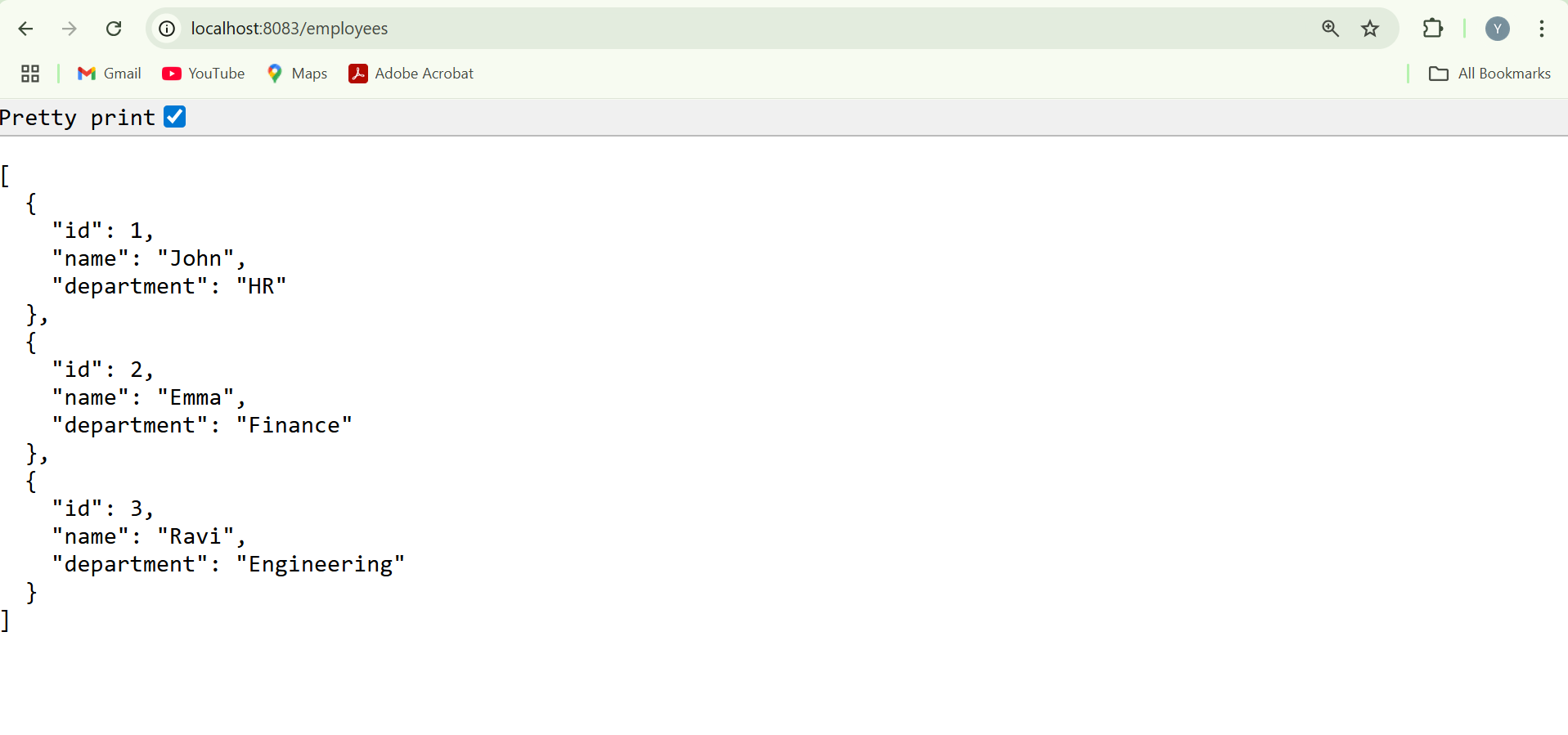
<scope>test</scope>

</dependency>

</dependencies>

**Output:**





**Create static employee list data using spring xml configuration**

**Employee.java**

package com.example.http\_demo;

import java.util.List;

public class Employee {

private int id;

private String name;

private double salary;

private boolean permanent;

private Department department;

private List<Skill> skills;

public Employee() {}

public Employee(int id, String name, double salary, boolean permanent, Department department, List<Skill> skills) {

this.id = id;

this.name = name;

this.salary = salary;

this.permanent = permanent;

this.department = department;

this.skills = skills;

}

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }

public boolean isPermanent() { return permanent; }

public void setPermanent(boolean permanent) { this.permanent = permanent; }

public Department getDepartment() { return department; }

public void setDepartment(Department department) { this.department = department; }

public List<Skill> getSkills() { return skills; }

public void setSkills(List<Skill> skills) { this.skills = skills; }

@Override

public String toString() {

return "Employee{" +

"id=" + id +

", name='" + name + '\'' +

", salary=" + salary +

", permanent=" + permanent +

", department=" + department +

", skills=" + skills +

'}';

}

}

**Department.java**

package com.example.http\_demo;

public class Department {

private int id;

private String name;

public Department() {}

public Department(int id, String name) {

this.id = id;

this.name = name;

}

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

@Override

public String toString() {

return "Department{" + "id=" + id + ", name='" + name + '\'' + '}';

}

}

**Skill.java**

package com.example.http\_demo.model;

public class Skill {

private String name;

public Skill() {}

public Skill(String name) { this.name = name; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

@Override

public String toString() {

return "Skill{" + "name='" + name + '\'' + '}';

} }

**employee.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="deptIT" class="com.example.http\_demo.model.Department">

<property name="id" value="1"/>

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

</bean>

<bean id="deptHR" class="com.example.http\_demo.model.Department">

<property name="id" value="2"/>

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

</bean>

<bean id="skillJava" class="com.example.http\_demo.model.Skill">

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

</bean>

<bean id="skillSpring" class="com.example.http\_demo.model.Skill">

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

</bean>

<bean id="skillExcel" class="com.example.http\_demo.model.Skill">

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

</bean>

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

<constructor-arg>

<list>

<bean class="com.example.http\_demo.model.Employee">

<property name="id" value="1"/>

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

<property name="salary" value="50000"/>

<property name="permanent" value="true"/>

<property name="department" ref="deptIT"/>

<property name="skills">

<list>

<ref bean="skillJava"/>

<ref bean="skillSpring"/>

</list>

</property>

</bean>

<bean class="com.example.http\_demo.model.Employee">

<property name="id" value="2"/>

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

<property name="salary" value="40000"/>

<property name="permanent" value="false"/>

<property name="department" ref="deptHR"/>

<property name="skills">

<list>

<ref bean="skillExcel"/>

</list>

</property>

</bean>

</list>

</constructor-arg>

</bean>

</beans>

**EmployeeDao.java**

package com.example.http\_demo.;

import com.example.http\_demo. Employee;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.List;

public class EmployeeDao {

public static List<Employee> EMPLOYEE\_LIST;

static {

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

EMPLOYEE\_LIST = (List<Employee>) context.getBean("employeeList");

}

public List<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

}

**EmployeeController.java**

package com.example.http\_demo;

import com.example.http\_demo.dao.EmployeeDao;

import com.example.http\_demo.model.Employee;

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

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

import java.util.List;

@RestController

public class EmployeeController {

private final EmployeeDao employeeDao;

public EmployeeController() {

this.employeeDao = new EmployeeDao();

}

@GetMapping("/employees")

public List<Employee> getEmployees() {

return employeeDao.getAllEmployees();

}

}

**SpringLearnApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

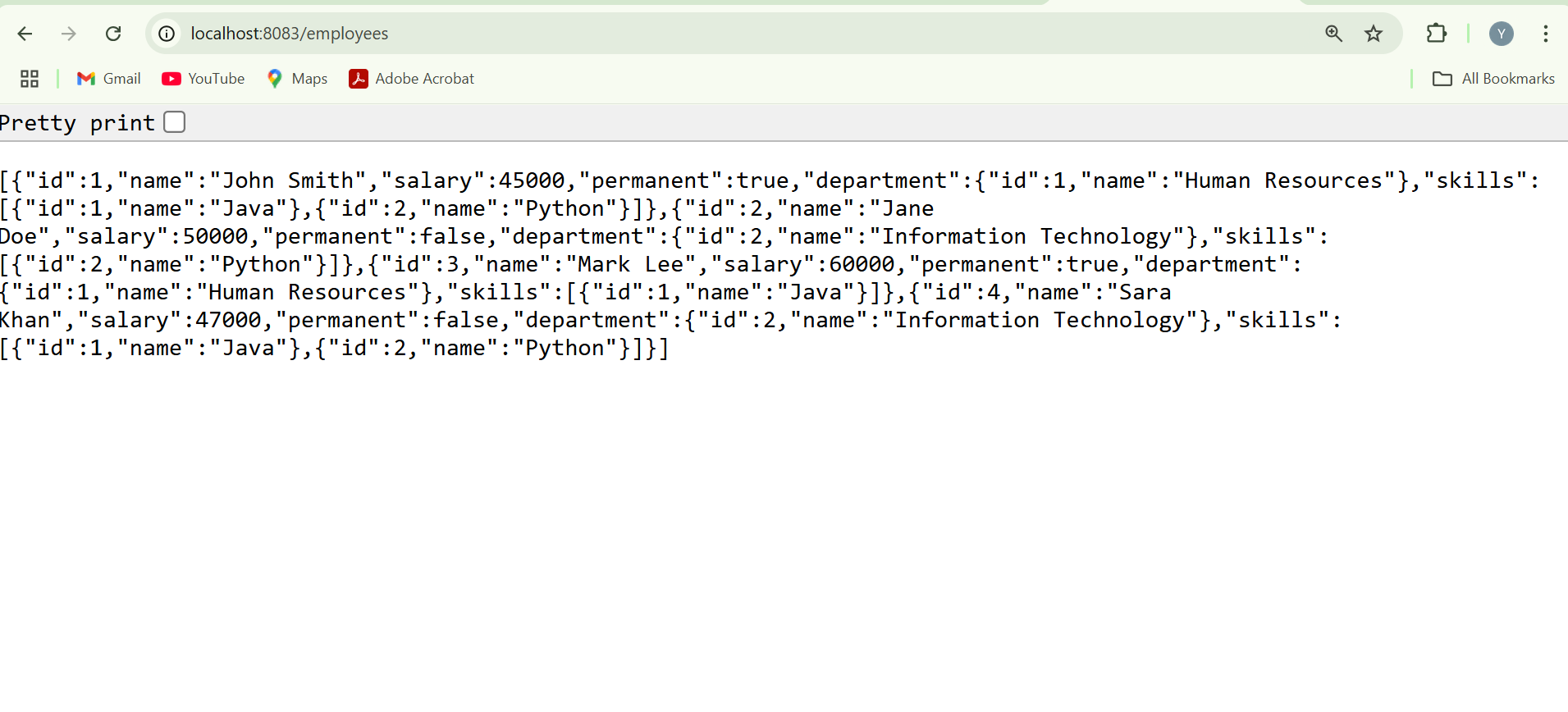
public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Output:**



**Create REST service to gets all employees**

**SpringLearnApplication.java**

package com.example.http\_demo;

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);

}

}

**SecurityConfig.java**

package com.example.http\_demo;

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.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable().authorizeHttpRequests().anyRequest().permitAll();

return http.build();

}

}

**Employee.java**

package com.example.http\_demo;

import java.util.List;

public class Employee {

private int id;

private String name;

private double salary;

private boolean permanent;

private Department department;

private List<Skill> skillList;

public Employee(int id, String name, double salary, boolean permanent, Department department, List<Skill> skillList) {

this.id = id;

this.name = name;

this.salary = salary;

this.permanent = permanent;

this.department = department;

this.skillList = skillList;

}

public int getId() { return id; }

public String getName() { return name; }

public double getSalary() { return salary; }

public boolean isPermanent() { return permanent; }

public Department getDepartment() { return department; }

public List<Skill> getSkillList() { return skillList; }

public void setId(int id) { this.id = id; }

public void setName(String name) { this.name = name; }

public void setSalary(double salary) { this.salary = salary; }

public void setPermanent(boolean permanent) { this.permanent = permanent; }

public void setDepartment(Department department) { this.department = department; }

public void setSkillList(List<Skill> skillList) { this.skillList = skillList; }

}

**Department.java**

package com.example.http\_demo;

public class Department {

private int id;

private String name;

public Department(int id, String name) {

this.id = id;

this.name = name;

}

public int getId() { return id; }

public String getName() { return name; }

public void setId(int id) { this.id = id; }

public void setName(String name) { this.name = name; }

}

**Skill.java**

package com.example.http\_demo;

public class Skill {

private int id;

private String name;

public Skill(int id, String name) {

this.id = id;

this.name = name;

}

public int getId() { return id; }

public String getName() { return name; }

public void setId(int id) { this.id = id; }

public void setName(String name) { this.name = name; }

}

**EmployeeDao.java**

package com.example.http\_demo;

import org.springframework.stereotype.Component;

import java.util.Arrays;

import java.util.List;

@Component

public class EmployeeDao {

private static final List<Employee> EMPLOYEE\_LIST = Arrays.asList(

new Employee(1, "John Doe", 50000, true,

new Department(1, "HR"),

Arrays.asList(new Skill(1, "Java"), new Skill(2, "Spring Boot"))

),

new Employee(2, "Jane Smith", 60000, false,

new Department(2, "Finance"),

Arrays.asList(new Skill(3, "Python"))

)

);

public List<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

}

**EmployeeService.java**

package com.example.http\_demo;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service

public class EmployeeService {

private final EmployeeDao employeeDao;

public EmployeeService(EmployeeDao employeeDao) {

this.employeeDao = employeeDao;

}

@Transactional

public List<Employee> getAllEmployees() {

return employeeDao.getAllEmployees();

}

}

**EmployeeController.java**

package com.example.http\_demo;

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

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

import java.util.List;

@RestController

public class EmployeeController {

private final EmployeeService employeeService;

public EmployeeController(EmployeeService employeeService) {

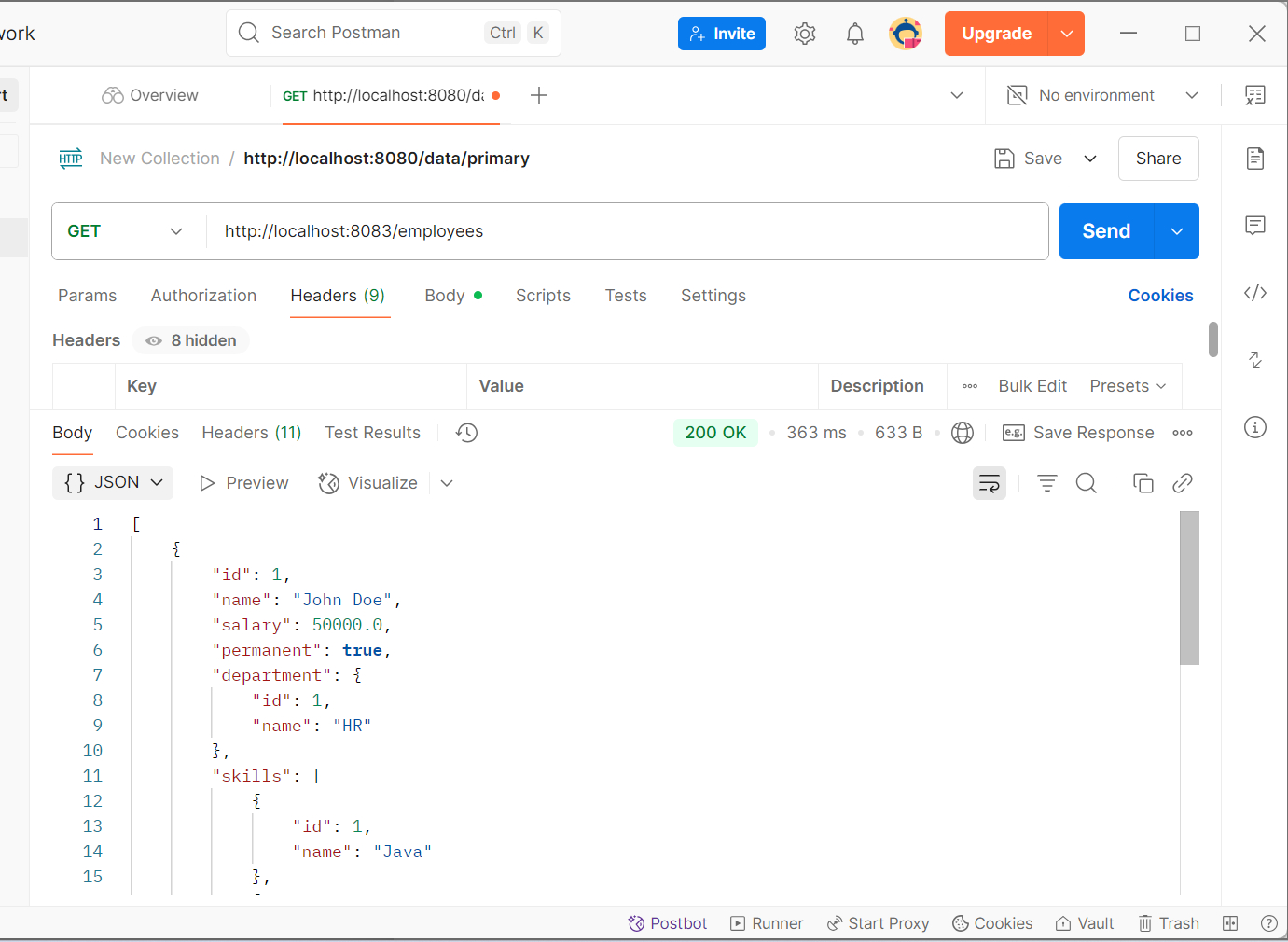
this.employeeService = employeeService;

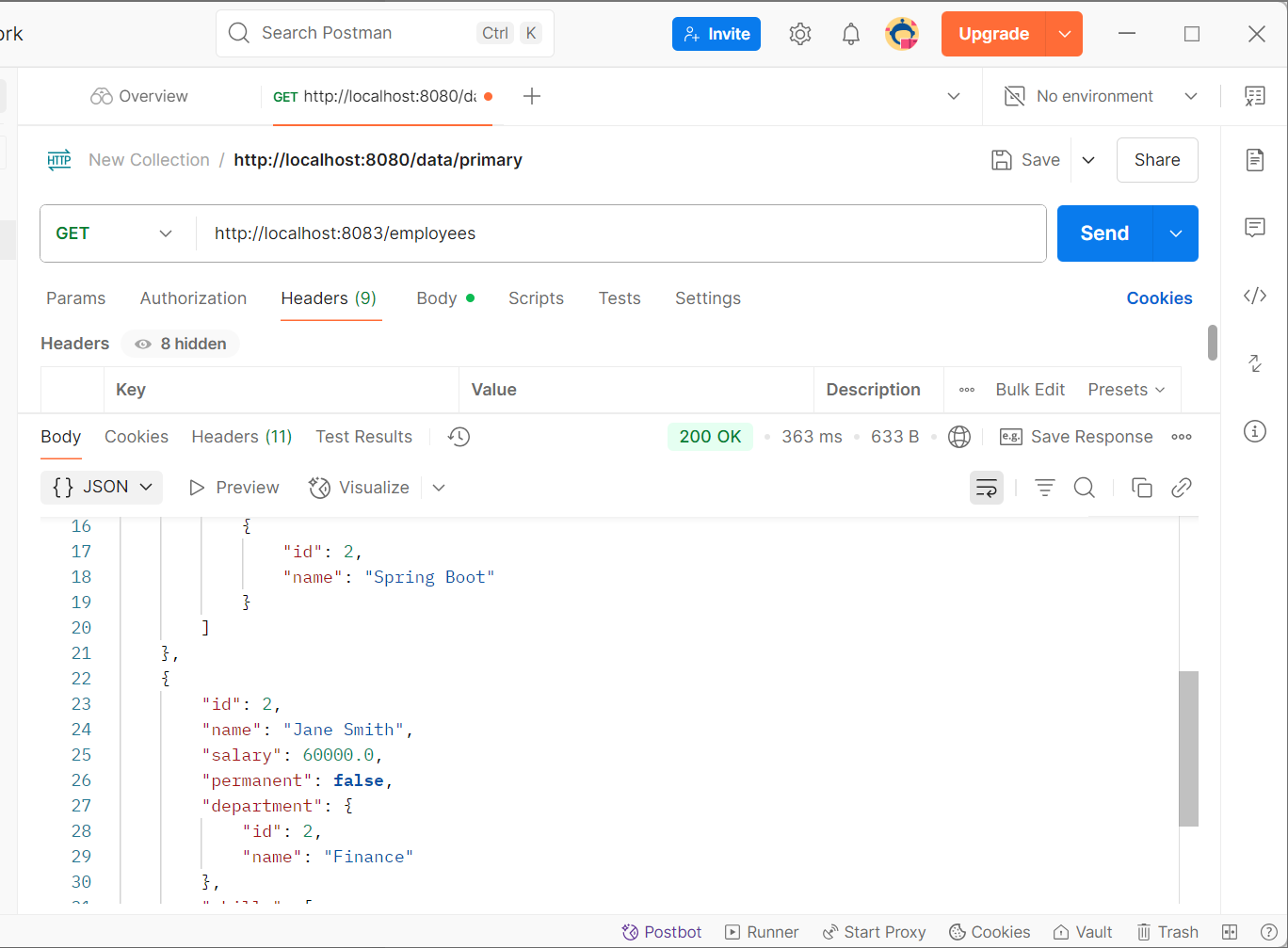
}

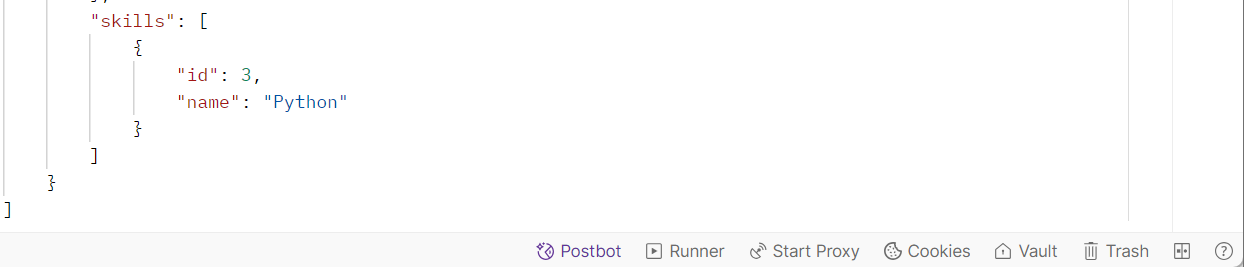
@GetMapping("/employees")

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees(); }

**Output:**





**Create REST service for department**

**Department.java**

package com.example.http\_demo;

public class Department {

private int id;

private String name;

public Department() {}

public Department(int id, String name) {

this.id = id;

this.name = name;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**DepartmentDao.java**

package com.example.http\_demo;

import java.util.ArrayList;

import java.util.List;

public class DepartmentDao {

public static List<Department> DEPARTMENT\_LIST;

static {

DEPARTMENT\_LIST = new ArrayList<>();

DEPARTMENT\_LIST.add(new Department(1, "Computer Science"));

DEPARTMENT\_LIST.add(new Department(2, "Information Technology"));

DEPARTMENT\_LIST.add(new Department(3, "Human Resources"));

}

public List<Department> getAllDepartments() {

return DEPARTMENT\_LIST;

}

}

**DepartmentService.java**

package com.example.http\_demo;

import java.util.List;

import org.springframework.stereotype.Service;

@Service

public class DepartmentService {

DepartmentDao departmentDao = new DepartmentDao();

public List<Department> getAllDepartments() {

return departmentDao.getAllDepartments();

}

}

**DepartmentController.java**

package com.example.http\_demo;

import java.util.List;

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

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

@RestController

public class DepartmentController {

@Autowired

private DepartmentService departmentService;

@GetMapping("/departments")

public List<Department> getAllDepartments() {

return departmentService.getAllDepartments();

}

}

**SpringLearnApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

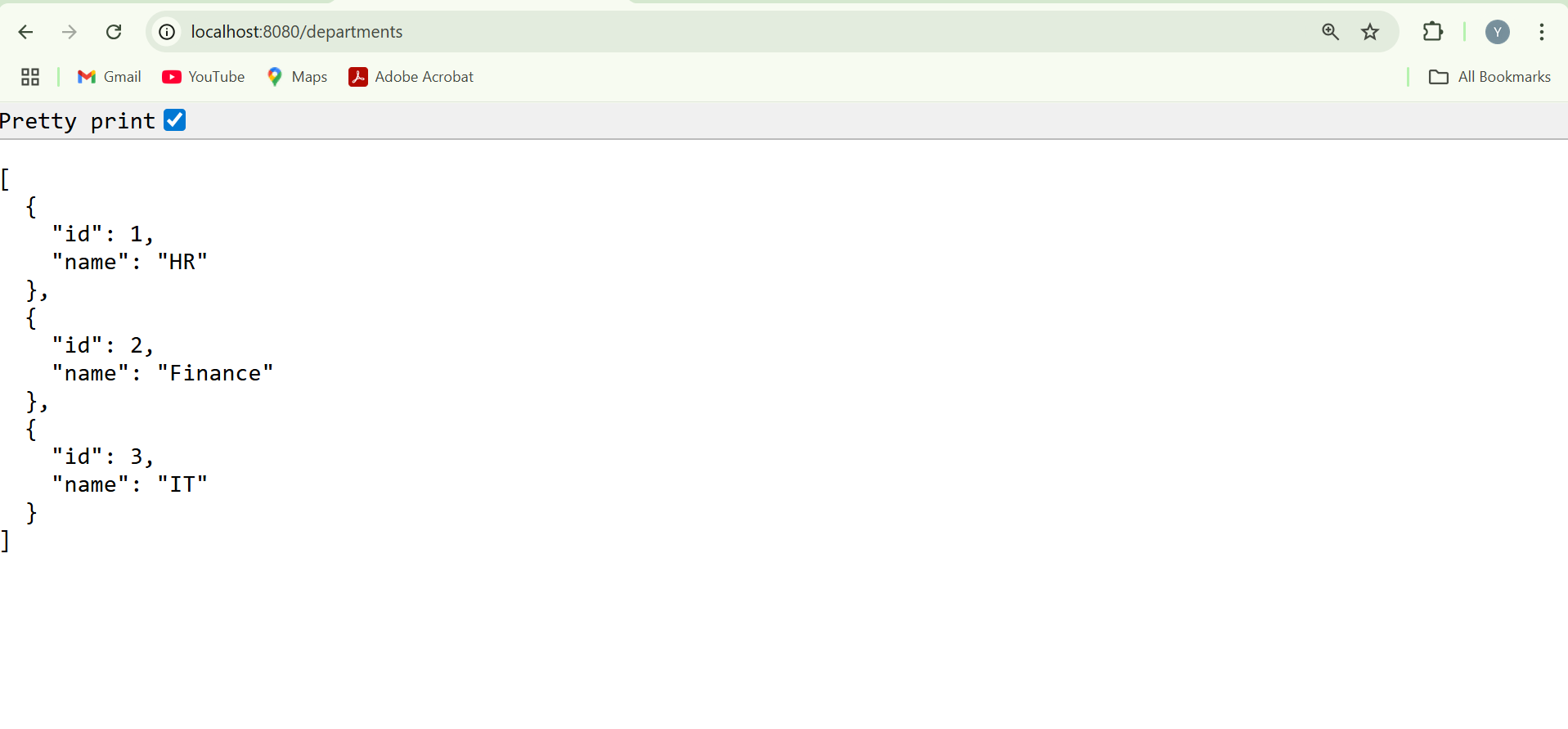
public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Output:**



**Significance of HTTP Method Types in RESTful Web Services** 

HTTP methods (GET, POST, PUT, DELETE) are crucial in RESTful APIs as they define the **type of operation** to be performed on a resource. They provide **semantic clarity**, enable **standardized communication**, and align directly with **CRUD operations**. While these methods signal the intended action, the **actual logic and persistence must be implemented in the application**. This separation ensures a clean, maintainable, and interoperable API design.

| **CRUD Operation** | **RESTful Method** | **Description** |
| --- | --- | --- |
| Create | POST | Add a new resource |
| Read | GET | Retrieve resource(s) |
| Update | PUT | Modify an existing resource |
| Delete | DELETE | Remove a resource |
|  |  |  |

**RESTful Web Service resource naming guidelines**

**HttpDemoApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HttpDemoApplication {

public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**Country.java**

package com.example.http\_demo;

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;

}

}

**CountryService.java**

package com.example.http\_demo;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

@Service

public class CountryService {

private static final List<Country> countryList = new ArrayList<>();

static {

countryList.add(new Country("IN", "India"));

countryList.add(new Country("US", "United States"));

countryList.add(new Country("JP", "Japan"));

}

public List<Country> getAllCountries() {

return countryList;

}

public Country getCountryByCode(String code) {

return countryList.stream()

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

.findFirst()

.orElse(null);

}

public Country addCountry(Country country) {

countryList.add(country);

return country;

}

public Country updateCountry(Country country) {

for (int i = 0; i < countryList.size(); i++) {

if (countryList.get(i).getCode().equalsIgnoreCase(country.getCode())) {

countryList.set(i, country);

return country;

}

}

return null;

}

public void deleteCountry(String code) {

countryList.removeIf(c -> c.getCode().equalsIgnoreCase(code));

}

}

**CountryController.java**

package com.example.http\_demo;

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

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

import java.util.List;

@RestController

@RequestMapping("/countries")

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping

public List<Country> getAllCountries() {

return countryService.getAllCountries();

}

@GetMapping("/{code}")

public Country getCountryByCode(@PathVariable String code) {

return countryService.getCountryByCode(code);

}

@PostMapping

public Country addCountry(@RequestBody Country country) {

return countryService.addCountry(country);

}

@PutMapping

public Country updateCountry(@RequestBody Country country) {

return countryService.updateCountry(country);

}

@DeleteMapping("/{code}")

public void deleteCountry(@PathVariable String code) {

countryService.deleteCountry(code);

}

}

**pom.xml**

<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

http://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>3.1.5</version>

<relativePath/>

</parent>

<groupId>com.example</groupId>

<artifactId>http-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>http-demo</name>

<description>RESTful Country Web Service</description>

<packaging>jar</packaging>

<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-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<mainClass>com.example.http\_demo.HttpDemoApplication</mainClass>

</configuration>

</plugin>

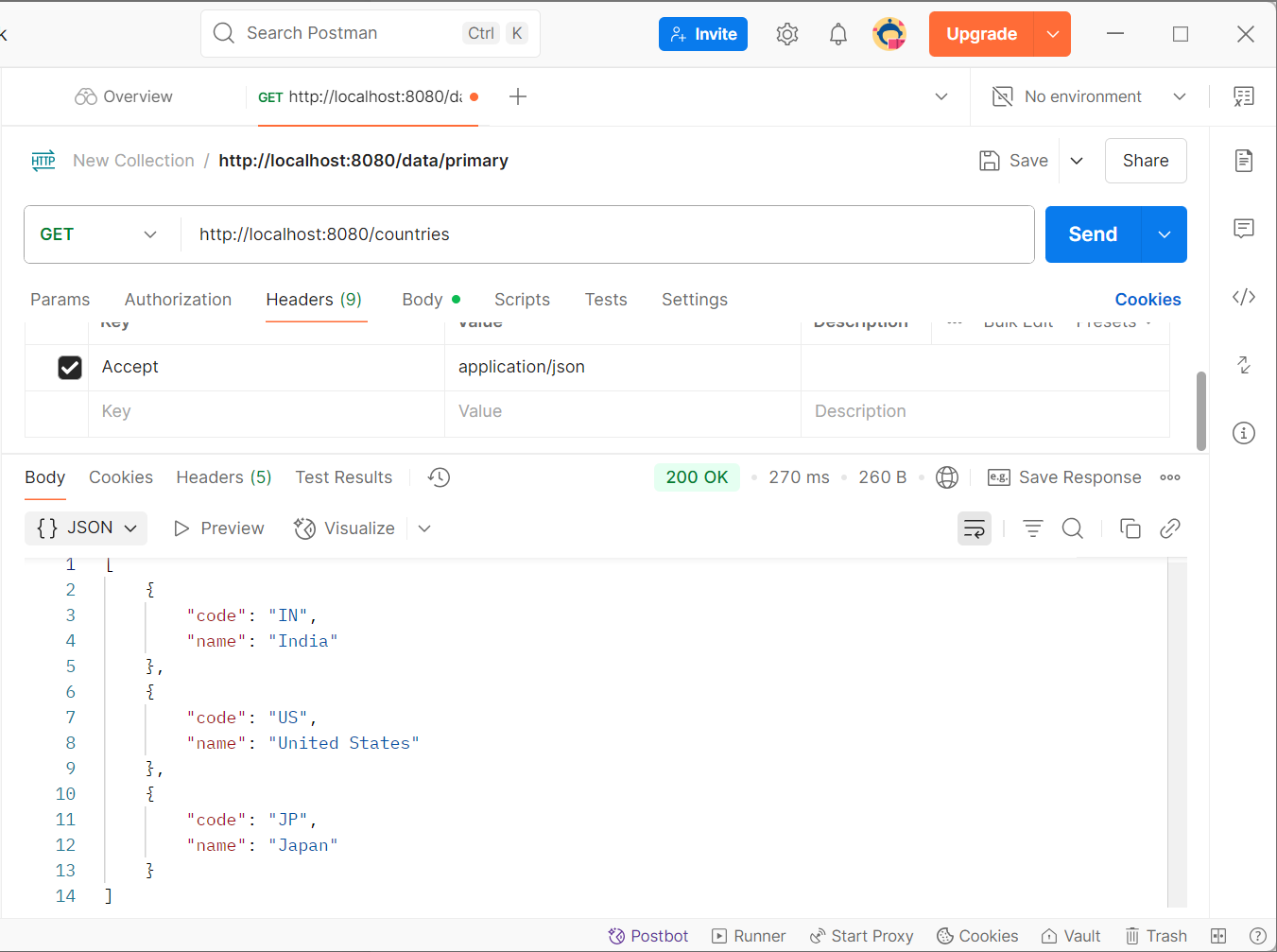
</plugins>

</build>

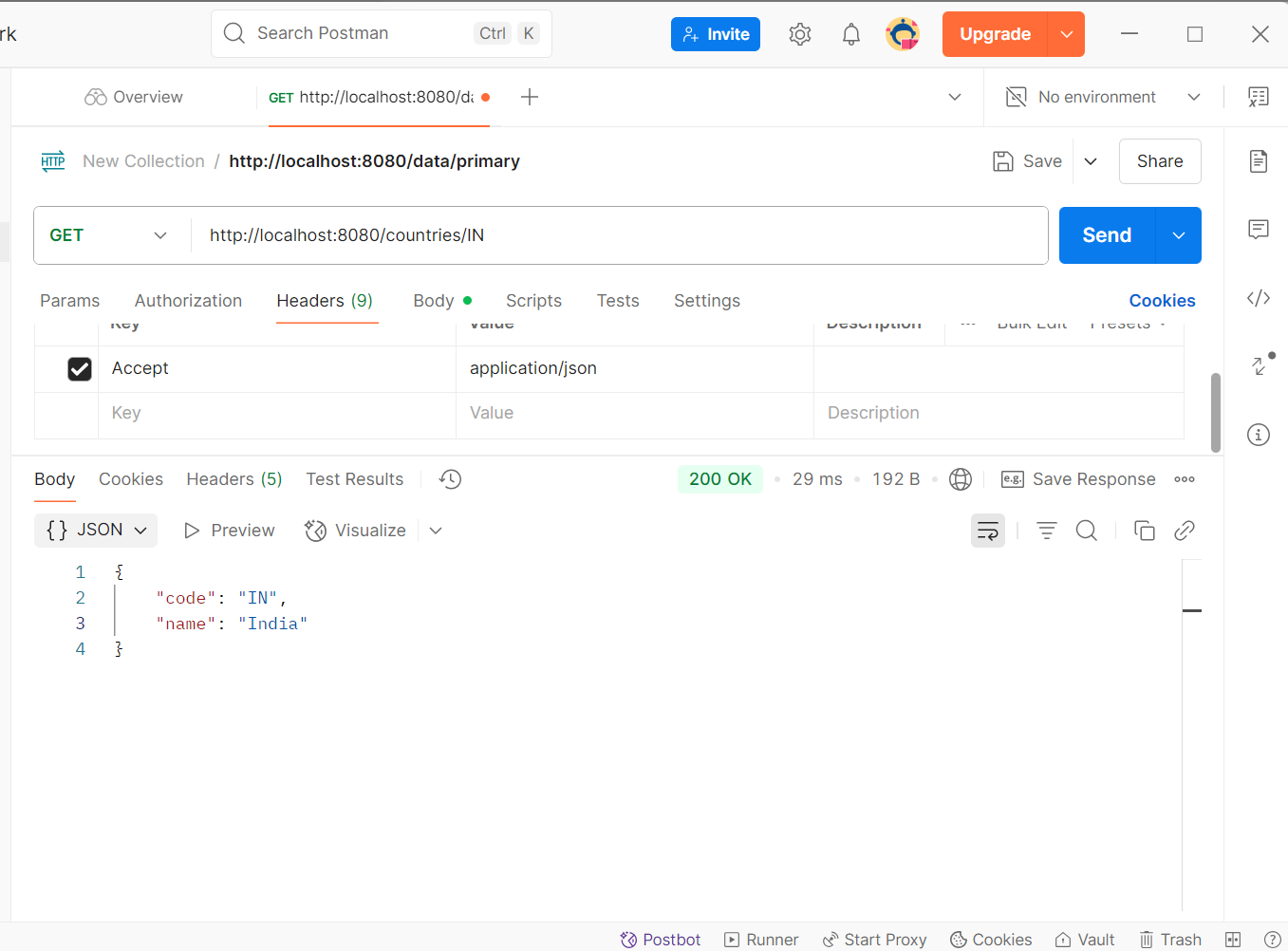
</project>

**Output:**

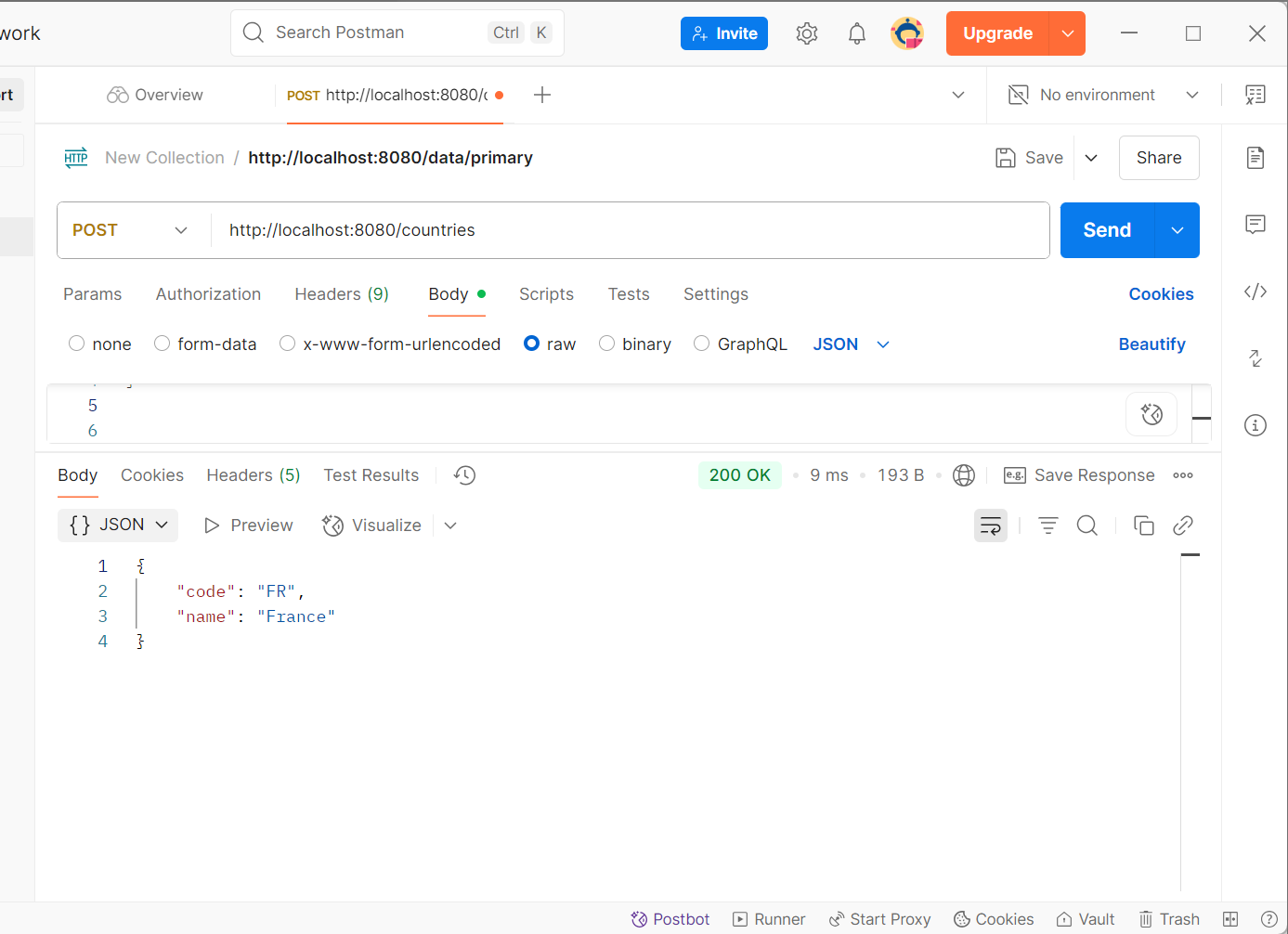
1. **Get all countries**



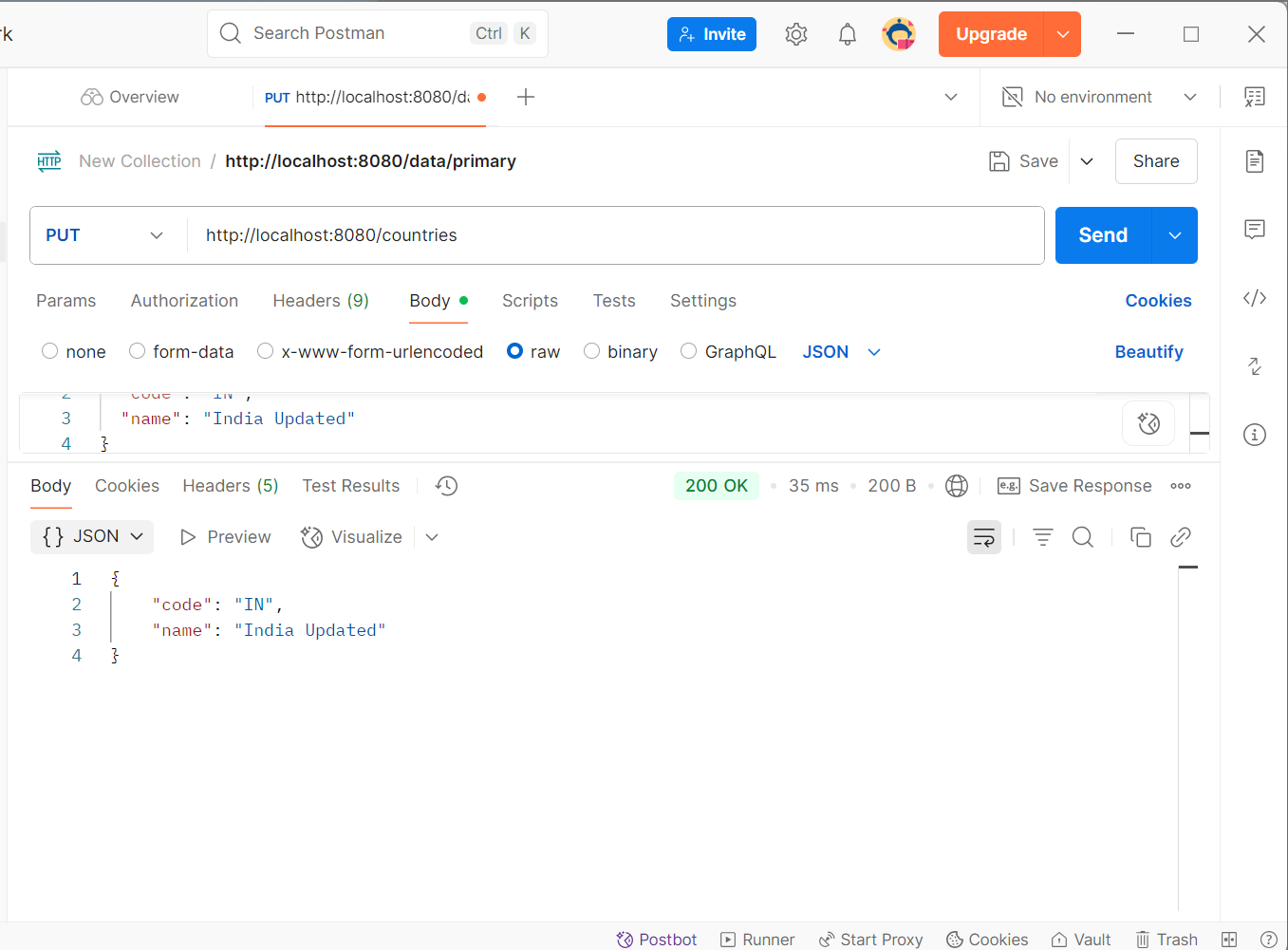
1. **Get country by code**



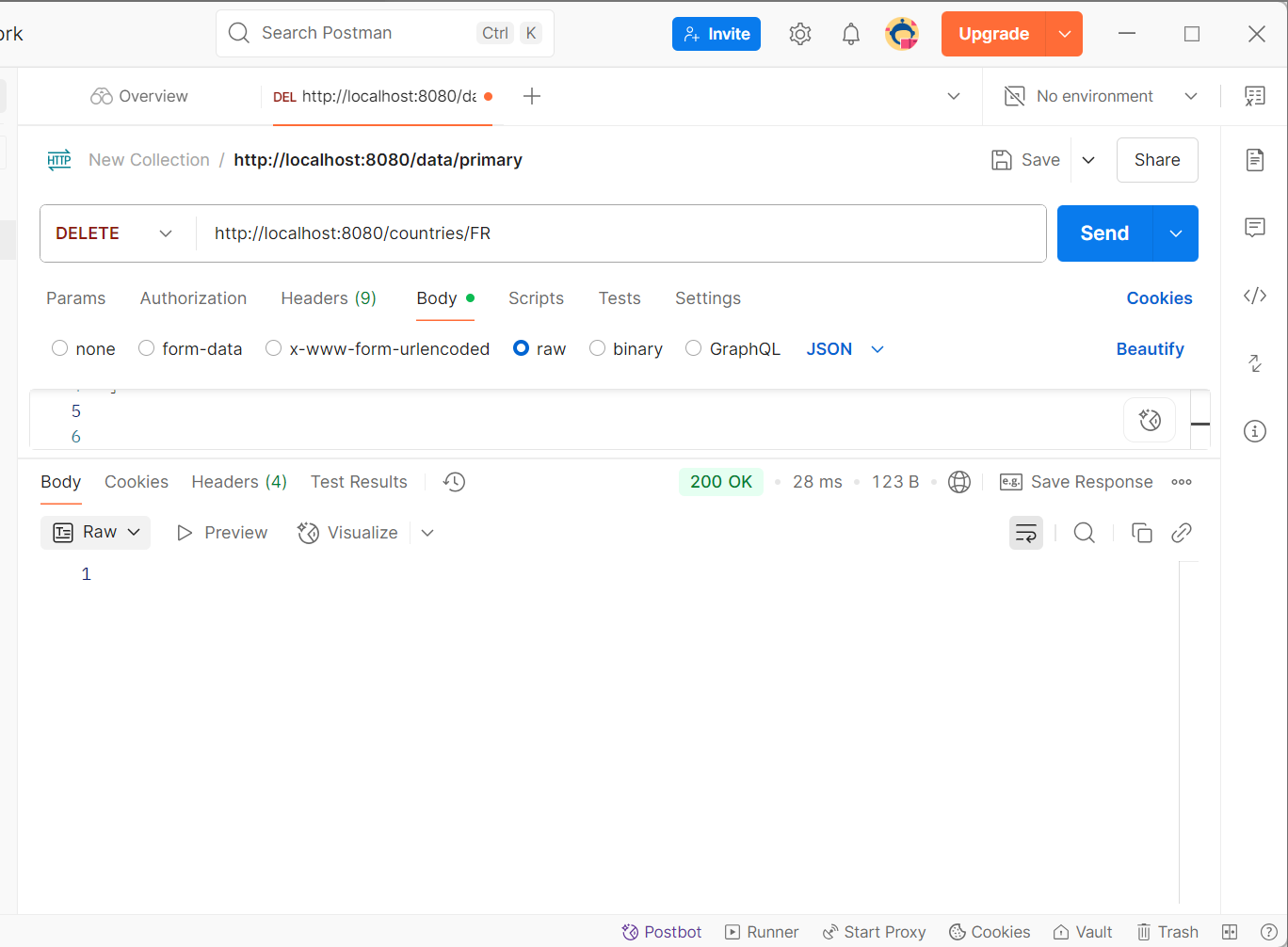
**3.Create a country**



**4.Update a country**



**5.Delete a country:**



**Create RESTful Web Service to handle POST request of Country**   
  
**HttpDemoApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HttpDemoApplication {

public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**CountryController.java**

package com.example.http\_demo;

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

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

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

@RestController

@RequestMapping("/countries")

public class CountryController {

@PostMapping

public void addCountry() {

System.out.println("Start");

}

}

**pom.xml**

<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

http://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>3.1.5</version>

<relativePath/>

</parent>

<groupId>com.example</groupId>

<artifactId>http-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>http-demo</name>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<mainClass>com.example.http\_demo.HttpDemoApplication</mainClass>

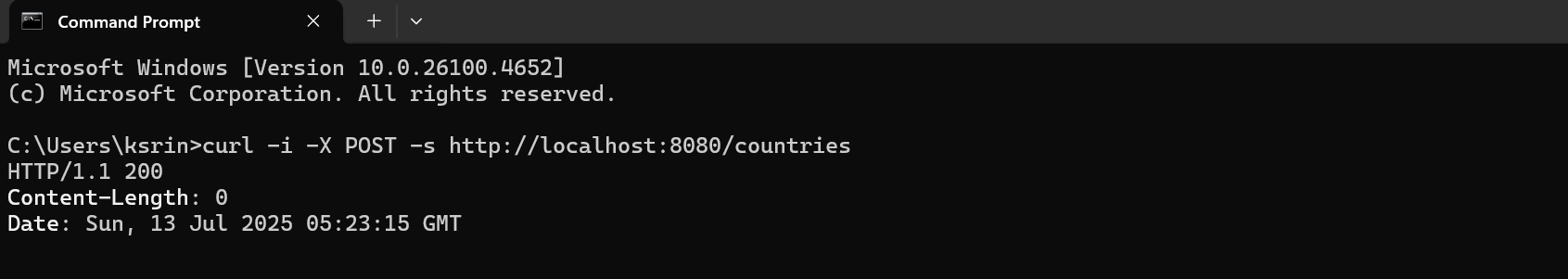
</configuration>

</plugin>

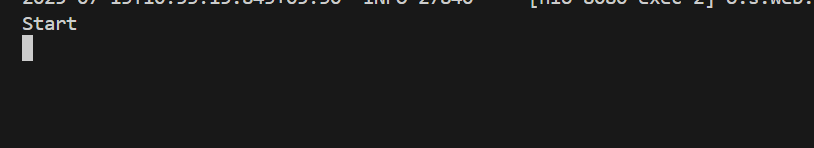
</plugins>

</build>

</project>

**Output:**  






**Read country data as a bean in RESTful Web Service**

**HttpDemoApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HttpDemoApplication {

public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**Country.java**

package com.example.http\_demo;

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 String getName() {

return name;

}

public void setCode(String code) {

this.code = code;

}

public void setName(String name) {

this.name = name;

}

}

**CountryController.java**

package com.example.http\_demo;

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

@RestController

@RequestMapping("/countries")

public class CountryController {

@PostMapping

public Country addCountry(@RequestBody Country country) {

System.out.println("Start");

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

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

return country;

}

}

**pom.xml**

<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

http://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>3.1.5</version>

<relativePath/>

</parent>

<groupId>com.example</groupId>

<artifactId>http-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<mainClass>com.example.http\_demo.HttpDemoApplication</mainClass>

</configuration>

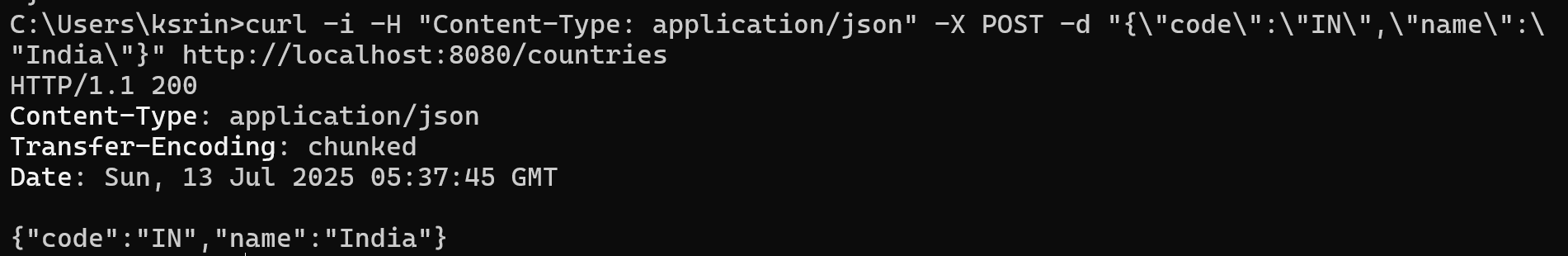
</plugin>

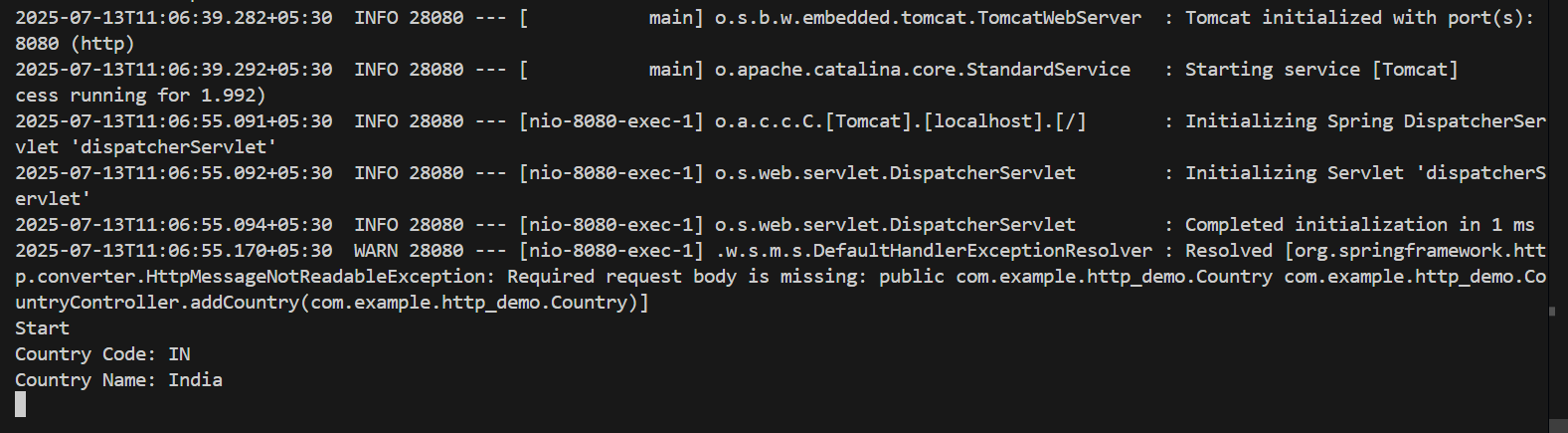
</plugins>

</build>

</project>

**Output:**





**1. How Spring converts the request payload into a Country bean**

When a client sends a JSON payload in a POST request, Spring automatically converts that JSON into a Java object using a process called **data binding**. This is made possible by the @RequestBody annotation in the controller.

@PostMapping

public Country addCountry(@RequestBody Country country) {

...

}

This tells Spring:  
 "Take the JSON request body and bind it to a Country object."

**2. Spring uses Jackson parser**

Internally, Spring Boot uses a JSON processing library called **Jackson**. It automatically deserializes JSON into Java objects and serializes Java objects back into JSON.

Jackson is included by default in Spring Boot via the spring-boot-starter-web dependency.

**3. How JSON fields map to methods: Reflection + InitCaps + get/set**

For each field in the JSON payload:

* Spring looks at the **field name**
* Capitalizes it using **InitCaps** (e.g., name → Name)
* Prepends it with set to get setName()
* Uses **Java Reflection API** to find and call this setter on the object

Example:

{

"code": "IN",

"name": "India"

}

Jackson will call:

country.setCode("IN");

country.setName("India");

**4. Object Creation and Method Invocation**

The flow:

1. Spring creates a new empty Country object using the **default constructor**.
2. For each key in the JSON:
   * Spring identifies the corresponding setter (setCode, setName, etc.)
   * Invokes these setters with the respective values
3. After population, the fully initialized Country object is passed to the controller method.

**5. Controller method execution**

Once the Country object is ready, Spring calls your controller method and **injects** the object as an argument.

public Country addCountry(@RequestBody Country country)

You can now use this country object directly.

**6. Bean Naming Conventions**

Spring follows JavaBean naming conventions for automatic binding:

* Private fields
* Public getters and setters
* No-argument constructor (important for deserialization)
* Method names must follow the getX() and setX() pattern where X is the capitalized field name.

private String name;

public String getName() { return name; }

public void setName(String name) { this.name = name; }

This allows Spring + Jackson to serialize and deserialize objects correctly.

**Validating country code**   
  
**HttpDemoApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HttpDemoApplication {

public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**Country.java**

package com.example.http\_demo;

import jakarta.validation.constraints.NotNull;

import jakarta.validation.constraints.Size;

public class Country {

@NotNull

@Size(min = 2, max = 2, message = "Country code should be 2 characters")

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;

}

}

**CountryController.java**

package com.example.http\_demo;

import org.springframework.http.HttpStatus;

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

import org.springframework.web.server.ResponseStatusException;

import jakarta.validation.Validation;

import jakarta.validation.Validator;

import jakarta.validation.ValidatorFactory;

import jakarta.validation.ConstraintViolation;

import java.util.ArrayList;

import java.util.List;

import java.util.Set;

@RestController

@RequestMapping("/countries")

public class CountryController {

@PostMapping

public Country addCountry(@RequestBody Country country) {

System.out.println("Start");

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

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

ValidatorFactory factory = Validation.buildDefaultValidatorFactory();

Validator validator = factory.getValidator();

Set<ConstraintViolation<Country>> violations = validator.validate(country);

List<String> errors = new ArrayList<>();

for (ConstraintViolation<Country> violation : violations) {

errors.add(violation.getMessage());

}

if (!errors.isEmpty()) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST, errors.toString());

}

return country;

}

}

**pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>http-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<parent>

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

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

<version>3.1.5</version>

<relativePath/>

</parent>

<properties>

<java.version>21</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-starter-validation</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<mainClass>com.example.http\_demo.HttpDemoApplication</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Output:**



If EmployeeController also needs validation:

* You must **repeat** the same manual validation logic (ValidatorFactory, Set<ConstraintViolation>, etc.).

**Disadvantage:**

* Code duplication
* Low reusability
* Hard to maintain
* Violates DRY principle

**Include global exception handler for validation errors**

**Country.java**

package com.example.http\_demo;

import jakarta.validation.constraints.NotNull;

import jakarta.validation.constraints.Size;

public class Country {

@NotNull

@Size(min = 2, max = 2, message = "Country code should be 2 characters")

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;

}

}

**CountryController.java**

package com.example.http\_demo;

import jakarta.validation.Valid;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

@RestController

@RequestMapping("/countries")

public class CountryController {

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

@PostMapping

public Country addCountry(@RequestBody @Valid Country country) {

LOGGER.info("CountryController: Start");

LOGGER.info("Country Data: code={}, name={}", country.getCode(), country.getName());

LOGGER.info("CountryController: End");

return country;

}

}

**GlobalExceptionHandler.java**

package com.example.http\_demo;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatusCode;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.MethodArgumentNotValidException;

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

import org.springframework.web.context.request.WebRequest;

import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import java.util.\*;

import java.util.stream.Collectors;

@ControllerAdvice

public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {

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

@Override

protected ResponseEntity<Object> handleMethodArgumentNotValid(

MethodArgumentNotValidException ex,

HttpHeaders headers,

HttpStatusCode status,

WebRequest request) {

LOGGER.info("GlobalExceptionHandler: Start");

Map<String, Object> body = new LinkedHashMap<>();

body.put("timestamp", new Date());

body.put("status", status.value());

List<String> errors = ex.getBindingResult()

.getFieldErrors()

.stream()

.map(fieldError -> fieldError.getDefaultMessage())

.collect(Collectors.toList());

body.put("errors", errors);

LOGGER.info("GlobalExceptionHandler: End");

return new ResponseEntity<>(body, headers, status);

}

}

**HttpDemoApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HttpDemoApplication {

public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**pom.xml**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>jakarta.validation</groupId>

<artifactId>jakarta.validation-api</artifactId>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

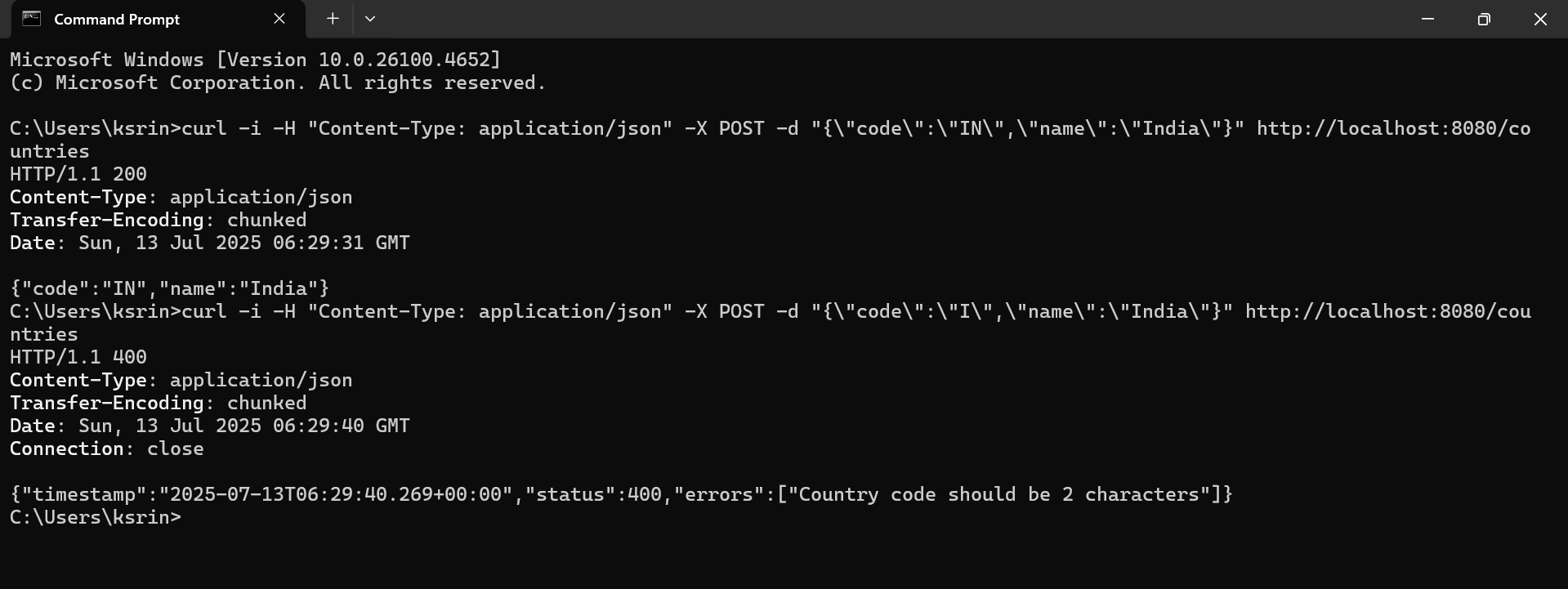
<groupId>org.springframework.boot</groupId>

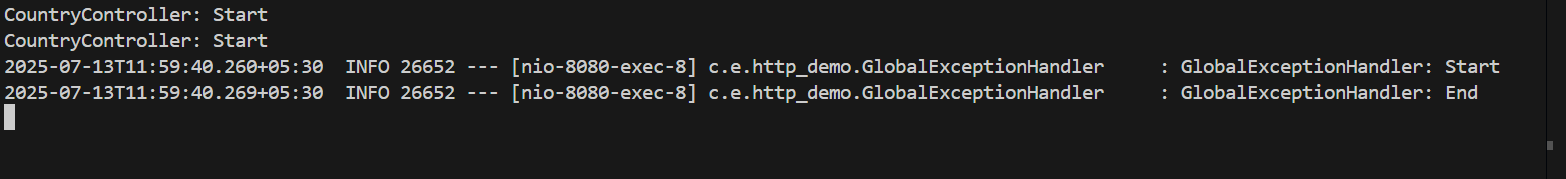
<artifactId>spring-boot-starter-logging</artifactId>

</dependency>

</dependencies>

**Output:**





**Implement REST service for updating an employee**

**Employee.java**

package com.example.http\_demo;

import com.fasterxml.jackson.annotation.JsonFormat;

import jakarta.validation.constraints.\*;

import java.util.Date;

import java.util.List;

public class Employee {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

@NotNull

@Min(0)

private Double salary;

@NotNull

private Boolean permanent;

@JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd/MM/yyyy")

private Date dateOfBirth;

@NotNull

private Department department;

@NotNull

private List<Skill> skills;

public Integer getId() { return id; }

public void setId(Integer id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public Double getSalary() { return salary; }

public void setSalary(Double salary) { this.salary = salary; }

public Boolean getPermanent() { return permanent; }

public void setPermanent(Boolean permanent) { this.permanent = permanent; }

public Date getDateOfBirth() { return dateOfBirth; }

public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth = dateOfBirth; }

public Department getDepartment() { return department; }

public void setDepartment(Department department) { this.department = department; }

public List<Skill> getSkills() { return skills; }

public void setSkills(List<Skill> skills) { this.skills = skills; }

}

**Department.java**

package com.example.http\_demo;

import jakarta.validation.constraints.\*;

public class Department {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

public Integer getId() { return id; }

public void setId(Integer id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

}

**Skill.java**

package com.example.http\_demo;

import jakarta.validation.constraints.\*;

public class Skill {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

public Integer getId() { return id; }

public void setId(Integer id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

}

**EmployeeNotFoundException.java**

package com.example.http\_demo;

import org.springframework.http.HttpStatus;

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

@ResponseStatus(HttpStatus.NOT\_FOUND)

public class EmployeeNotFoundException extends RuntimeException {

public EmployeeNotFoundException(String message) {

super(message);

}

}

**EmployeeDao.java**

package com.example.http\_demo;

import java.util.\*;

public class EmployeeDao {

public static List<Employee> EMPLOYEE\_LIST = new ArrayList<>();

public void updateEmployee(Employee employee) {

for (int i = 0; i < EMPLOYEE\_LIST.size(); i++) {

if (EMPLOYEE\_LIST.get(i).getId().equals(employee.getId())) {

EMPLOYEE\_LIST.set(i, employee);

return;

}

}

throw new EmployeeNotFoundException("Employee with ID " + employee.getId() + " not found.");

}

public List<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

}

**EmployeeService.java**

package com.example.http\_demo;

import org.springframework.stereotype.Service;

@Service

public class EmployeeService {

private EmployeeDao employeeDao = new EmployeeDao();

public void updateEmployee(Employee employee) {

employeeDao.updateEmployee(employee);

}

public java.util.List<Employee> getAllEmployees() {

return employeeDao.getAllEmployees();

}

}

**EmployeeController.java**

package com.example.http\_demo;

import jakarta.validation.Valid;

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

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

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PutMapping

public void updateEmployee(@RequestBody @Valid Employee employee) {

employeeService.updateEmployee(employee);

}

@GetMapping

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

}

**GlobalExceptionHandler.java**

package com.example.http\_demo;

import com.fasterxml.jackson.databind.exc.InvalidFormatException;

import org.springframework.http.\*;

import org.springframework.web.bind.MethodArgumentNotValidException;

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

import org.springframework.web.context.request.WebRequest;

import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import java.util.\*;

import java.util.stream.Collectors;

@ControllerAdvice

public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {

@Override

protected ResponseEntity<Object> handleMethodArgumentNotValid(

MethodArgumentNotValidException ex, HttpHeaders headers,

HttpStatusCode status, WebRequest request) {

Map<String, Object> body = new LinkedHashMap<>();

body.put("timestamp", new Date());

body.put("status", status.value());

List<String> errors = ex.getBindingResult().getFieldErrors().stream()

.map(err -> err.getDefaultMessage())

.collect(Collectors.toList());

body.put("errors", errors);

return new ResponseEntity<>(body, headers, status);

}

@Override

protected ResponseEntity<Object> handleHttpMessageNotReadable(

HttpMessageNotReadableException ex, HttpHeaders headers,

HttpStatusCode status, WebRequest request) {

Map<String, Object> body = new LinkedHashMap<>();

body.put("timestamp", new Date());

body.put("status", status.value());

body.put("error", "Bad Request");

if (ex.getCause() instanceof InvalidFormatException cause) {

for (InvalidFormatException.Reference ref : cause.getPath()) {

body.put("message", "Incorrect format for field '" + ref.getFieldName() + "'");

}

}

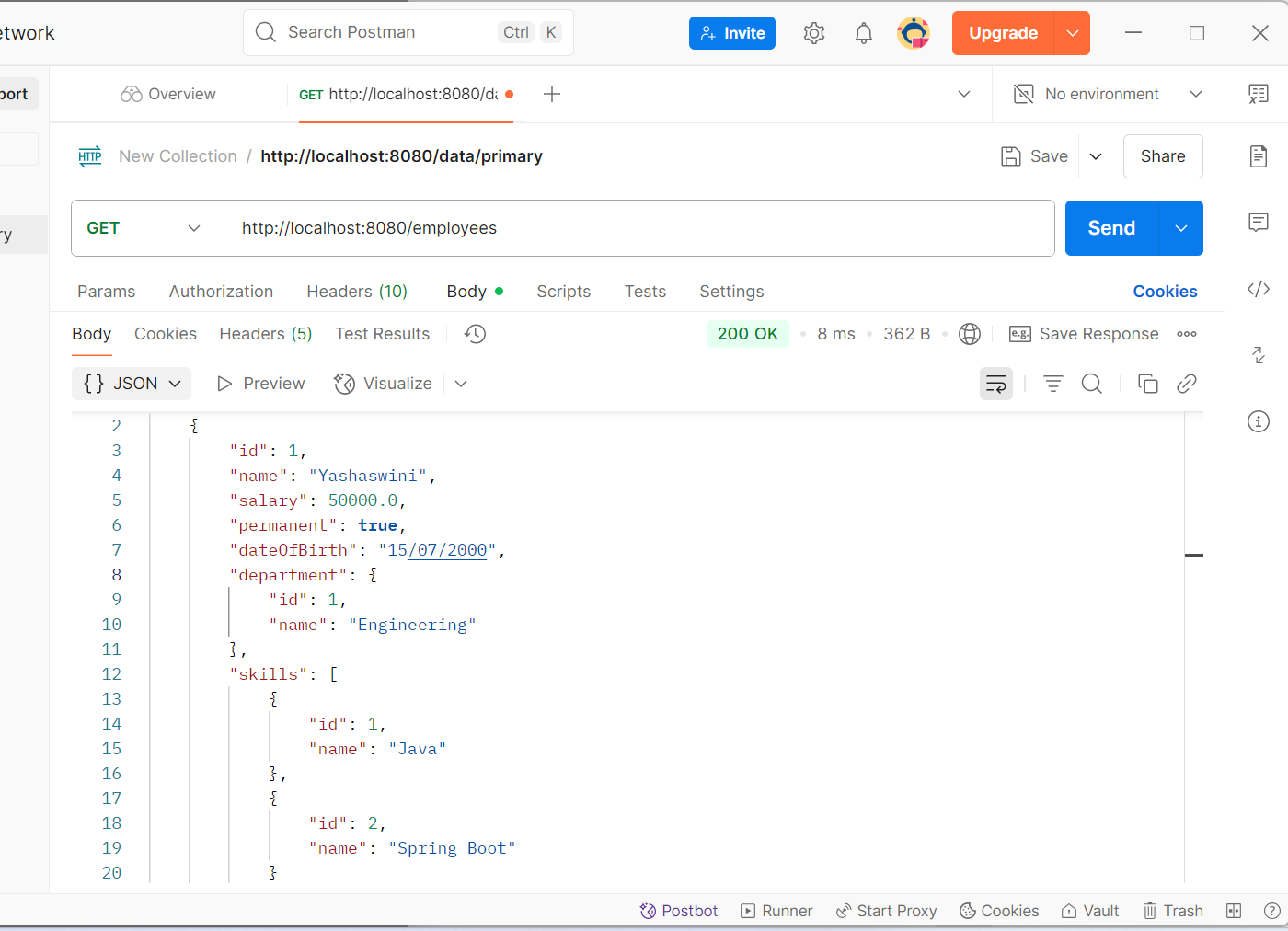
return new ResponseEntity<>(body, headers, status);

}

}

Output:





**Implement REST DELETE Service**

**Employee.java**

package com.example.http\_demo;

import com.fasterxml.jackson.annotation.JsonFormat;

import jakarta.validation.constraints.\*;

import java.util.Date;

import java.util.List;

public class Employee {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

@NotNull

@Min(0)

private Double salary;

@NotNull

private Boolean permanent;

@JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd/MM/yyyy")

private Date dateOfBirth;

@NotNull

private Department department;

@NotNull

private List<Skill> skills;

public Integer getId() { return id; }

public void setId(Integer id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public Double getSalary() { return salary; }

public void setSalary(Double salary) { this.salary = salary; }

public Boolean getPermanent() { return permanent; }

public void setPermanent(Boolean permanent) { this.permanent = permanent; }

public Date getDateOfBirth() { return dateOfBirth; }

public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth = dateOfBirth; }

public Department getDepartment() { return department; }

public void setDepartment(Department department) { this.department = department; }

public List<Skill> getSkills() { return skills; }

public void setSkills(List<Skill> skills) { this.skills = skills; }

}

**Department.java**

package com.example.http\_demo;

import jakarta.validation.constraints.\*;

public class Department {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

public Integer getId() { return id; }

public void setId(Integer id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

}

**Skill.java**

package com.example.http\_demo;

import jakarta.validation.constraints.\*;

public class Skill {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

public Integer getId() { return id; }

public void setId(Integer id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

}

**EmployeeNotFoundException.java**

package com.example.http\_demo;

import org.springframework.http.HttpStatus;

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

@ResponseStatus(HttpStatus.NOT\_FOUND)

public class EmployeeNotFoundException extends RuntimeException {

public EmployeeNotFoundException(String message) {

super(message);

}

}

**EmployeeDao.java**

package com.example.http\_demo;

import java.util.\*;

public class EmployeeDao {

public static List<Employee> EMPLOYEE\_LIST = new ArrayList<>();

public void deleteEmployee(int id) {

Iterator<Employee> iterator = EMPLOYEE\_LIST.iterator();

while (iterator.hasNext()) {

if (iterator.next().getId() == id) {

iterator.remove();

return;

}

}

throw new EmployeeNotFoundException("Employee with ID " + id + " not found.");

}

public List<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

public void addEmployee(Employee employee) {

EMPLOYEE\_LIST.add(employee);

}

}

**EmployeeService.java**

package com.example.http\_demo;

import org.springframework.stereotype.Service;

@Service

public class EmployeeService {

private EmployeeDao dao = new EmployeeDao();

public void deleteEmployee(int id) {

dao.deleteEmployee(id);

}

public void addEmployee(Employee employee) {

dao.addEmployee(employee);

}

}

**EmployeeController.java**

package com.example.http\_demo;

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

@RestController

@RequestMapping("/employees")

public class EmployeeController {

private final EmployeeService service;

public EmployeeController(EmployeeService service) {

this.service = service;

}

@DeleteMapping("/{id}")

public void deleteEmployee(@PathVariable int id) {

service.deleteEmployee(id);

}

@PostMapping("/test-add")

public void addEmployee(@RequestBody Employee employee) {

service.addEmployee(employee);

}

}

**HttpDemoApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HttpDemoApplication {

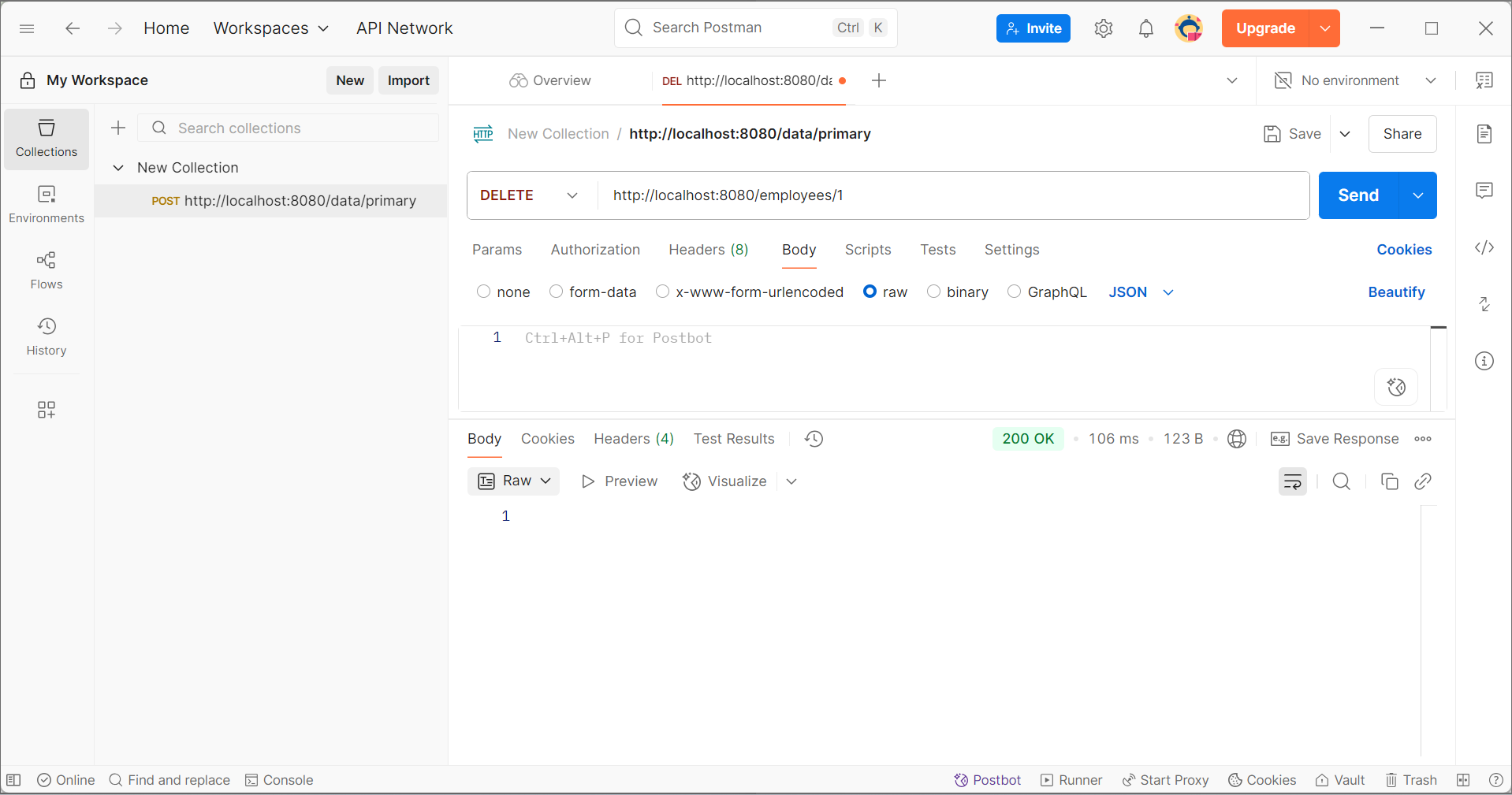
public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**Output:**



**Securing RESTful Web Services with Spring Security**

**pom.xml**

<dependency>

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

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

</dependency>

**SecurityConfig.java**

package com.example.http\_demo;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.web.SecurityFilterChain;

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

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public InMemoryUserDetailsManager userDetailsService(PasswordEncoder passwordEncoder) {

return new InMemoryUserDetailsManager(

User.withUsername("user")

.password(passwordEncoder.encode("password"))

.roles("USER")

.build()

);

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeHttpRequests(authz -> authz

.anyRequest().authenticated()

)

.httpBasic();

return http.build();

}

}

**Employee.java**

package com.example.http\_demo;

import com.fasterxml.jackson.annotation.JsonFormat;

import jakarta.validation.constraints.\*;

import java.util.Date;

import java.util.List;

public class Employee {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

@NotNull

@Min(0)

private Double salary;

@NotNull

private Boolean permanent;

@JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd/MM/yyyy")

private Date dateOfBirth;

@NotNull

private Department department;

@NotNull

private List<Skill> skills;

public Integer getId() { return id; }

public void setId(Integer id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public Double getSalary() { return salary; }

public void setSalary(Double salary) { this.salary = salary; }

public Boolean getPermanent() { return permanent; }

public void setPermanent(Boolean permanent) { this.permanent = permanent; }

public Date getDateOfBirth() { return dateOfBirth; }

public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth = dateOfBirth; }

public Department getDepartment() { return department; }

public void setDepartment(Department department) { this.department = department; }

public List<Skill> getSkills() { return skills; }

public void setSkills(List<Skill> skills) { this.skills = skills; }

}

**Department.java**

package com.example.http\_demo;

import jakarta.validation.constraints.\*;

public class Department {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

public Integer getId() { return id; }

public void setId(Integer id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }}

**Skill.java**

package com.example.http\_demo;

import jakarta.validation.constraints.\*;

public class Skill {

@NotNull

private Integer id;

@NotBlank

@Size(min = 1, max = 30)

private String name;

public Integer getId() { return id; }

public void setId(Integer id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

}

**EmployeeDao.java**

package com.example.http\_demo;

import java.util.ArrayList;

import java.util.List;

public class EmployeeDao {

public static List<Employee> EMPLOYEE\_LIST = new ArrayList<>();

public List<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

public void addEmployee(Employee employee) {

EMPLOYEE\_LIST.add(employee);

}

public void deleteEmployee(int id) {

Employee found = EMPLOYEE\_LIST.stream()

.filter(emp -> emp.getId() == id)

.findFirst()

.orElseThrow(() -> new EmployeeNotFoundException("Employee not found with ID: " + id));

EMPLOYEE\_LIST.remove(found);

}

}

**EmployeeService.java**

package com.example.http\_demo;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class EmployeeService {

private final EmployeeDao dao = new EmployeeDao();

public List<Employee> getAllEmployees() {

return dao.getAllEmployees();

}

public void addEmployee(Employee employee) {

dao.addEmployee(employee);

}

public void deleteEmployee(int id) {

dao.deleteEmployee(id);

}

}

**EmployeeController.java**

package com.example.http\_demo;

import jakarta.validation.Valid;

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

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

private final EmployeeService service;

public EmployeeController(EmployeeService service) {

this.service = service;

}

@GetMapping

public List<Employee> getEmployees() {

return service.getAllEmployees();

}

@PostMapping

public void addEmployee(@RequestBody @Valid Employee employee) {

service.addEmployee(employee);

}

@DeleteMapping("/{id}")

public void deleteEmployee(@PathVariable int id) {

service.deleteEmployee(id);

}

}

**EmployeeNotFoundException.java**

package com.example.http\_demo;

import org.springframework.http.HttpStatus;

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

@ResponseStatus(HttpStatus.NOT\_FOUND)

public class EmployeeNotFoundException extends RuntimeException {

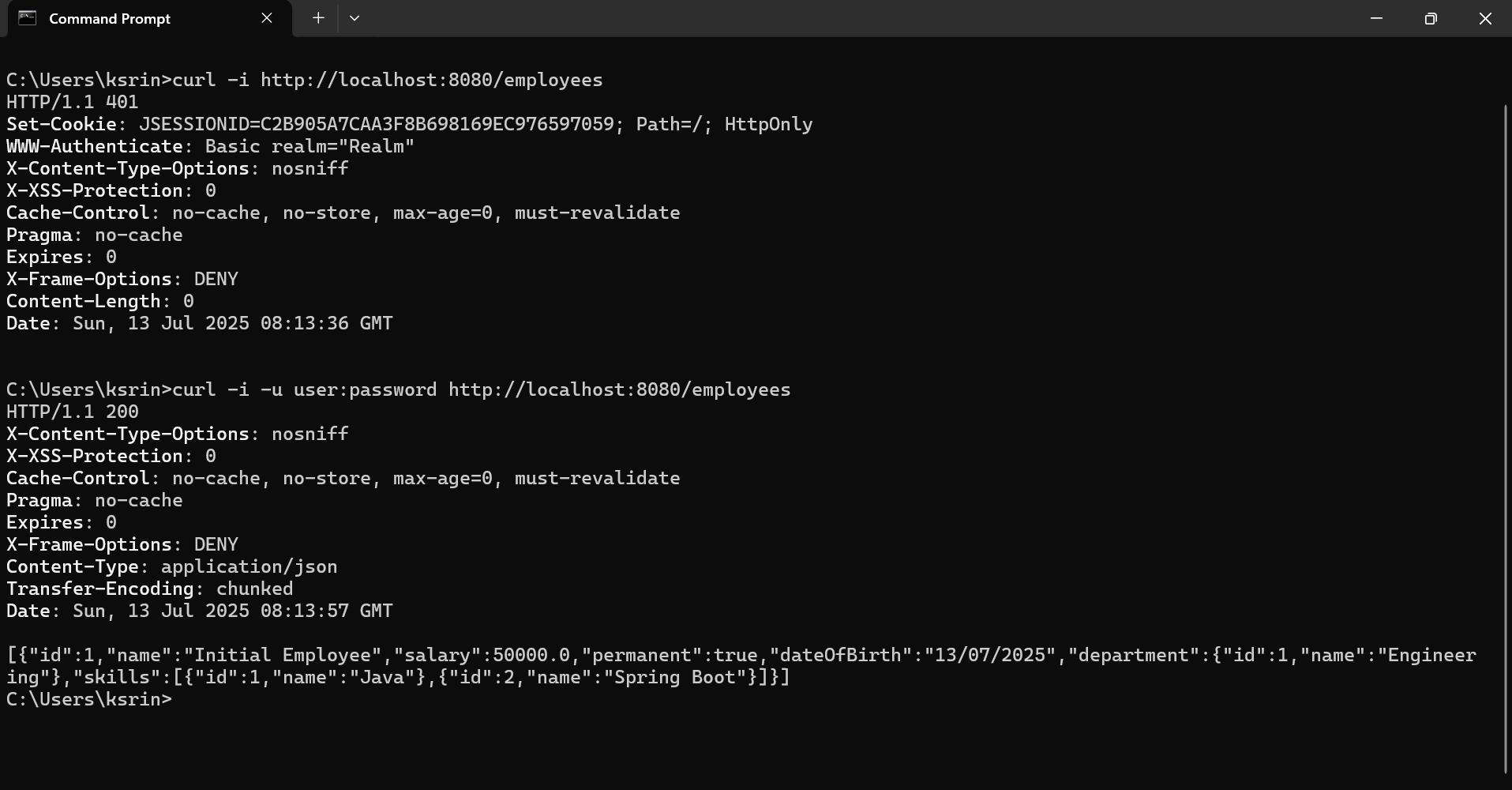
public EmployeeNotFoundException(String message) {

super(message);

}

}

**Output:**



**Creating users and roles in Spring Security**

**SecurityConfig.java**

package com.example.http\_demo;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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

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

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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 {

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

@Bean

public PasswordEncoder passwordEncoder() {

LOGGER.info("Start passwordEncoder");

return new BCryptPasswordEncoder();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.httpBasic()

.and()

.authorizeHttpRequests()

.requestMatchers("/employees").hasRole("USER")

.anyRequest().authenticated();

return http.build();

}

@Bean

public org.springframework.security.core.userdetails.UserDetailsService userDetailsService(PasswordEncoder encoder) {

var userService = new org.springframework.security.provisioning.InMemoryUserDetailsManager();

var user = org.springframework.security.core.userdetails.User

.withUsername("user")

.password(encoder.encode("pwd"))

.roles("USER")

.build();

var admin = org.springframework.security.core.userdetails.User

.withUsername("admin")

.password(encoder.encode("pwd"))

.roles("ADMIN")

.build();

userService.createUser(user);

userService.createUser(admin);

return userService;

}}

**EmployeeController.java**

package com.example.http\_demo;

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

import java.util.\*;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@GetMapping

public List<Employee> getAllEmployees() {

return EmployeeDao.EMPLOYEE\_LIST;

}

}

**Employee.java**

package com.example.http\_demo;

public class Employee {

private int id;

private String name;

public Employee() {}

public Employee(int id, String name) {

this.id = id;

this.name = name;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**EmployeeDao.java**

package com.example.http\_demo;

import java.util.\*;

public class EmployeeDao {

public static List<Employee> EMPLOYEE\_LIST = new ArrayList<>();

static {

EMPLOYEE\_LIST.add(new Employee(1, "Alice"));

EMPLOYEE\_LIST.add(new Employee(2, "Bob"));

}

}

HttpDemoApplication.java

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HttpDemoApplication {

public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**pom.xml**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

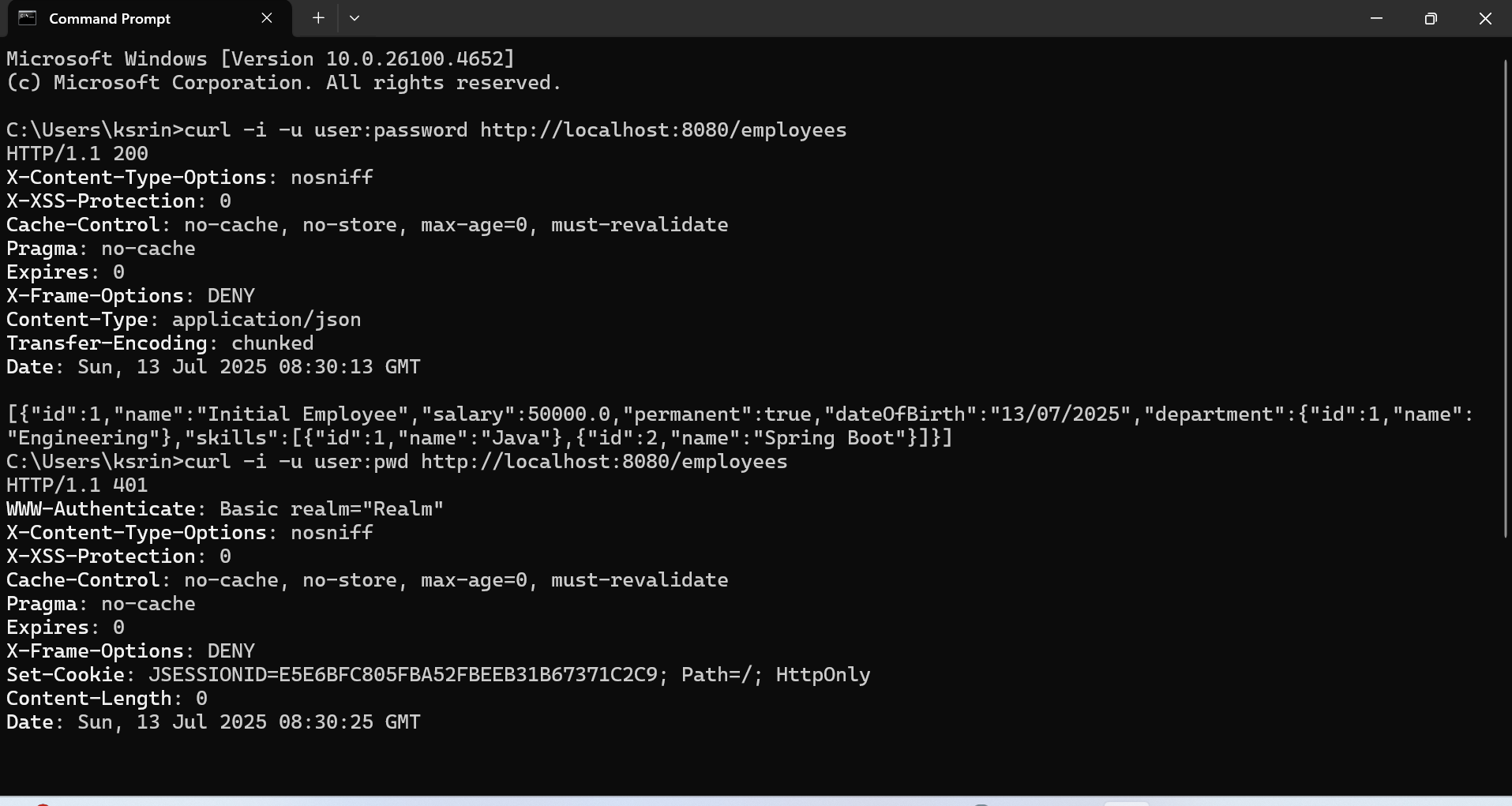
<groupId>org.springframework.boot</groupId>

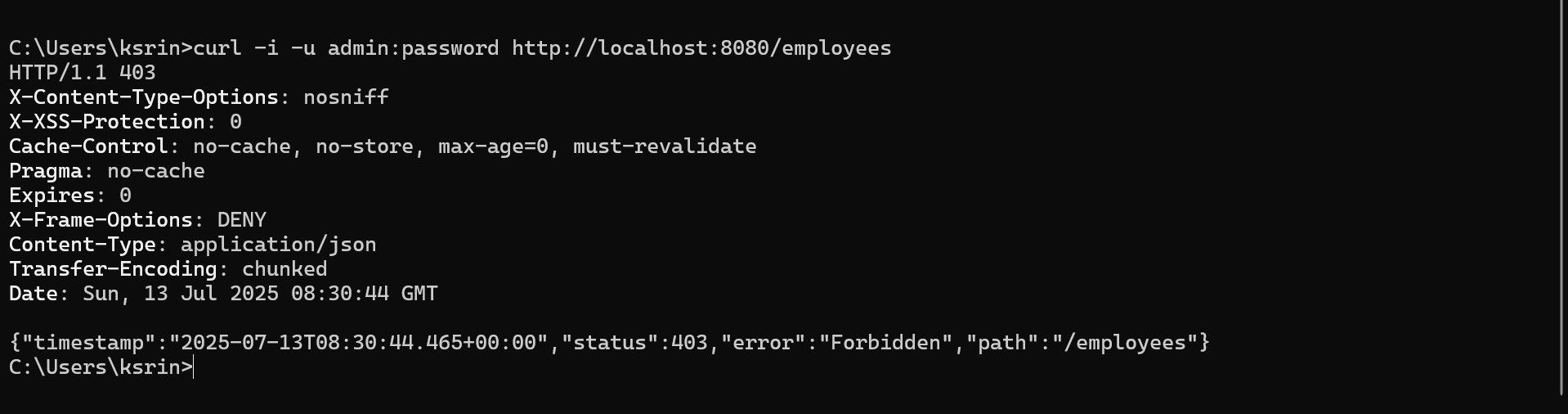
<artifactId>spring-boot-starter-security</artifactId>

</dependency>

</dependencies>

**Output:**





**Understanding JWT**

JWT (JSON Web Token) is an open standard (IETF RFC 7519) used for securely transmitting information between parties as a JSON object. It is commonly used in RESTful web services for authentication and authorization. A JWT consists of three parts: the Header, which specifies the algorithm used for signing (e.g., HS256); the Payload, which contains user-specific data like user ID and roles; and the Signature, which is generated by encoding the header and payload using a secret key. The typical flow involves a client sending a username and password to the server, which, upon validation, returns a JWT token. The client then includes this token in the Authorization header of subsequent requests. The server validates the token on every request, ensuring secure access. An exercise to understand this process involves copying header and payload examples from the JWT Wikipedia page, pasting them into [jwt.io](https://jwt.io), entering "secretkey" as the signature key, and observing how the encoded token matches the example provided on Wikipedia. This illustrates how JWTs are structured and verified.

**Create authentication service that returns JWT**   
**HttpDemoApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HttpDemoApplication {

public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**SecurityConfig.java**

package com.example.http\_demo;

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.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeHttpRequests(auth -> auth

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

.anyRequest().authenticated()

)

.httpBasic();

return http.build();

}

}

**JwtUtil.java**

package com.example.http\_demo;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

@Component

public class JwtUtil {

private String secret = "secretkey";

public String generateToken(String username) {

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

return Jwts.builder()

.setClaims(claims)

.setSubject(username)

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

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

.signWith(SignatureAlgorithm.HS256, secret)

.compact();

}

}

**AuthenticationController.java**

package com.example.http\_demo;

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

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

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@PostMapping("/authenticate")

public Map<String, String> generateToken(@RequestHeader("Authorization") String authHeader) {

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

byte[] credDecoded = java.util.Base64.getDecoder().decode(base64Credentials);

String credentials = new String(credDecoded);

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

String username = values[0];

String password = values[1];

if (!username.equals("user") || !password.equals("password")) {

throw new RuntimeException("Invalid credentials");

}

String token = jwtUtil.generateToken(username);

Map<String, String> response = new HashMap<>();

response.put("token", token);

return response;

}

}

**pom.xml**

<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>

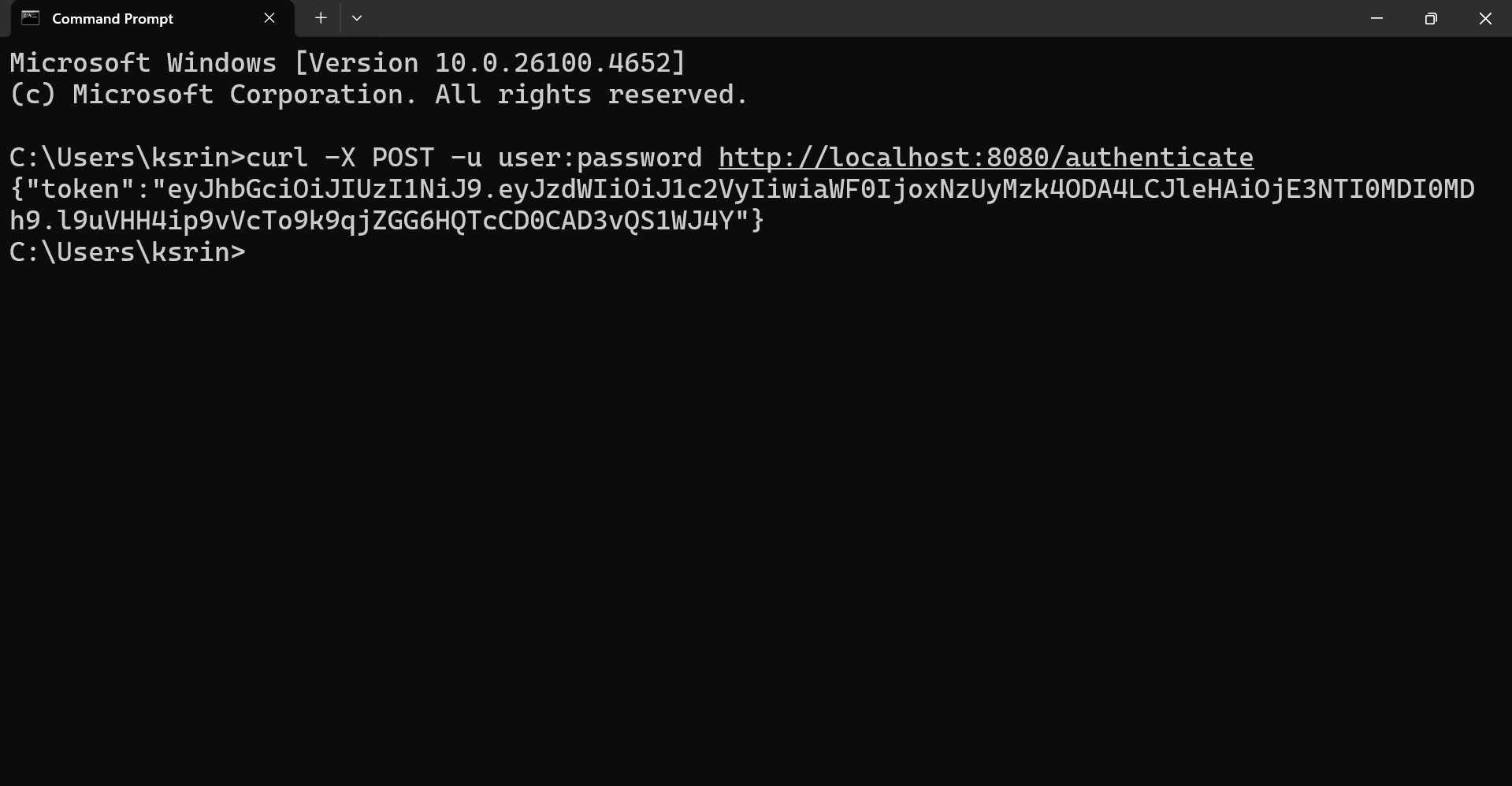
<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

**Output:**



**Create authentication controller and configure it in SecurityConfig**   
**AuthenticationController.java**

package com.example.http\_demo;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

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

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

LOGGER.info("START /authenticate");

LOGGER.debug("Authorization Header: {}", authHeader);

Map<String, String> map = new HashMap<>();

map.put("token", "");

LOGGER.info("END /authenticate");

return map;

}

}

**SecurityConfig.java**

package com.example.http\_demo;

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.configurers.AbstractHttpConfigurer;

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.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf(AbstractHttpConfigurer::disable)

.authorizeHttpRequests(auth -> auth

.requestMatchers("/countries").hasRole("USER")

.requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

)

.httpBasic();

return http.build();

}

@Bean

public UserDetailsService userDetailsService() {

UserDetails user = User.withUsername("user")

.password("{noop}pwd")

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

}

**HttpDemoApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HttpDemoApplication {

public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**pom.xml**

<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>

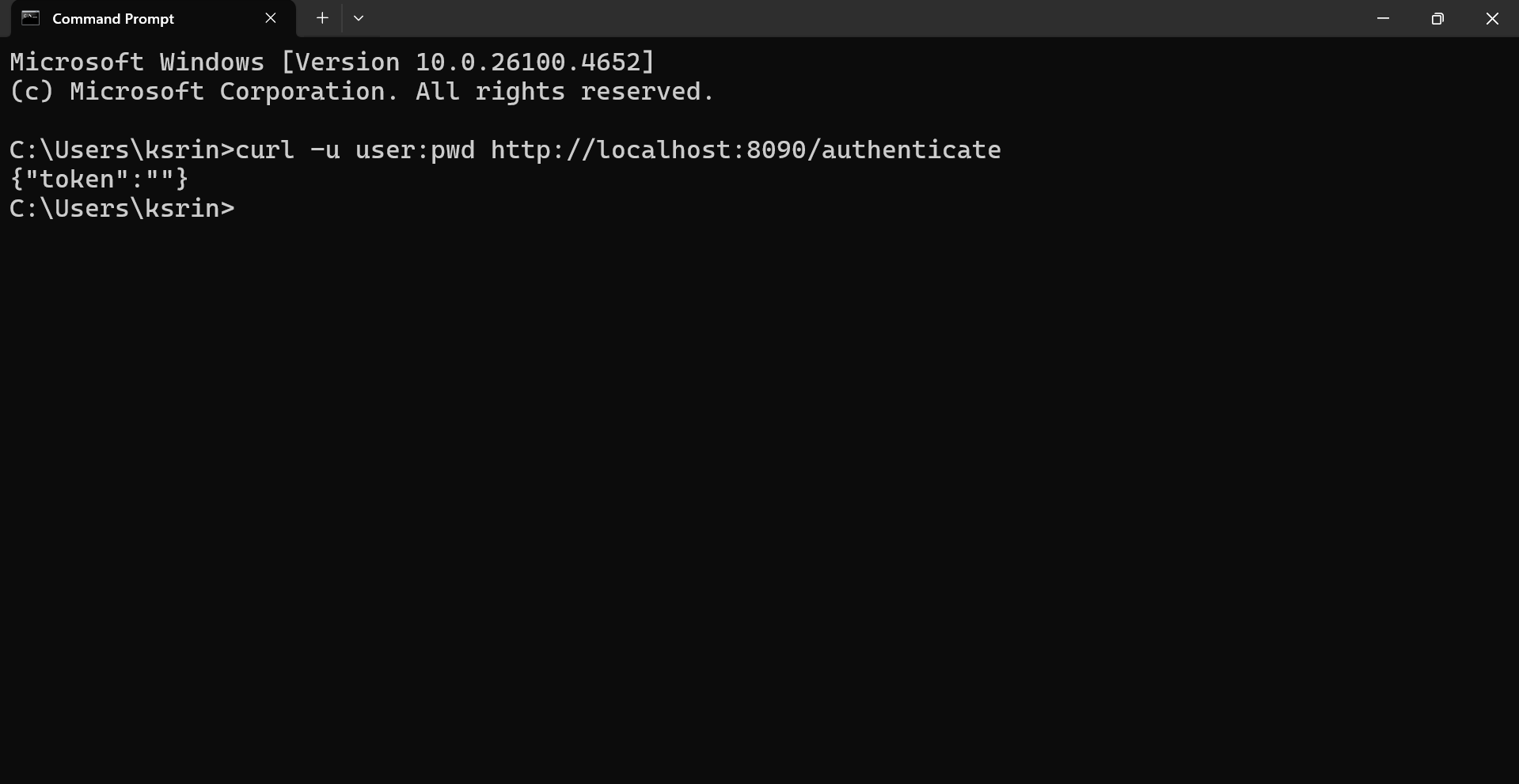
<dependency>

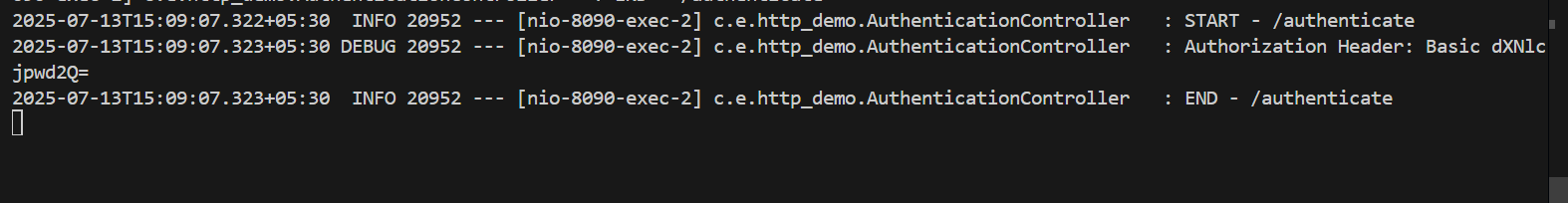
<groupId>org.springframework.boot</groupId>

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

</dependency>

</dependencies>

**Output:**



**Read Authorization header and decode the username and password**

**AuthenticationController.java**

package com.example.http\_demo;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

@RestController

public class AuthenticationController {

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

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

LOGGER.info("START /authenticate");

LOGGER.debug("Authorization Header: {}", authHeader);

String user = getUser(authHeader);

LOGGER.debug("Extracted User: {}", user);

Map<String, String> map = new HashMap<>();

map.put("token", "");

LOGGER.info("END /authenticate");

return map;

}

private String getUser(String authHeader) {

LOGGER.debug("Inside getUser method");

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

LOGGER.debug("Encoded Credentials: {}", encodedCredentials);

byte[] decodedBytes = Base64.getDecoder().decode(encodedCredentials);

String decodedString = new String(decodedBytes);

LOGGER.debug("Decoded String: {}", decodedString); // "user:pwd"

String username = decodedString.split(":")[0];

LOGGER.debug("Extracted Username: {}", username);

return username;

}

}

**SecurityConfig.java**

package com.example.http\_demo;

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.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

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

.authorizeHttpRequests(auth -> auth

.requestMatchers("/countries").hasRole("USER")

.requestMatchers("/authenticate").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

)

.httpBasic();

return http.build();

}

}

**application.properties**

server.port=8090

logging.level.com.example.http\_demo=DEBUG

**HttpDemoApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HttpDemoApplication {

public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**pom.xml**

<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>

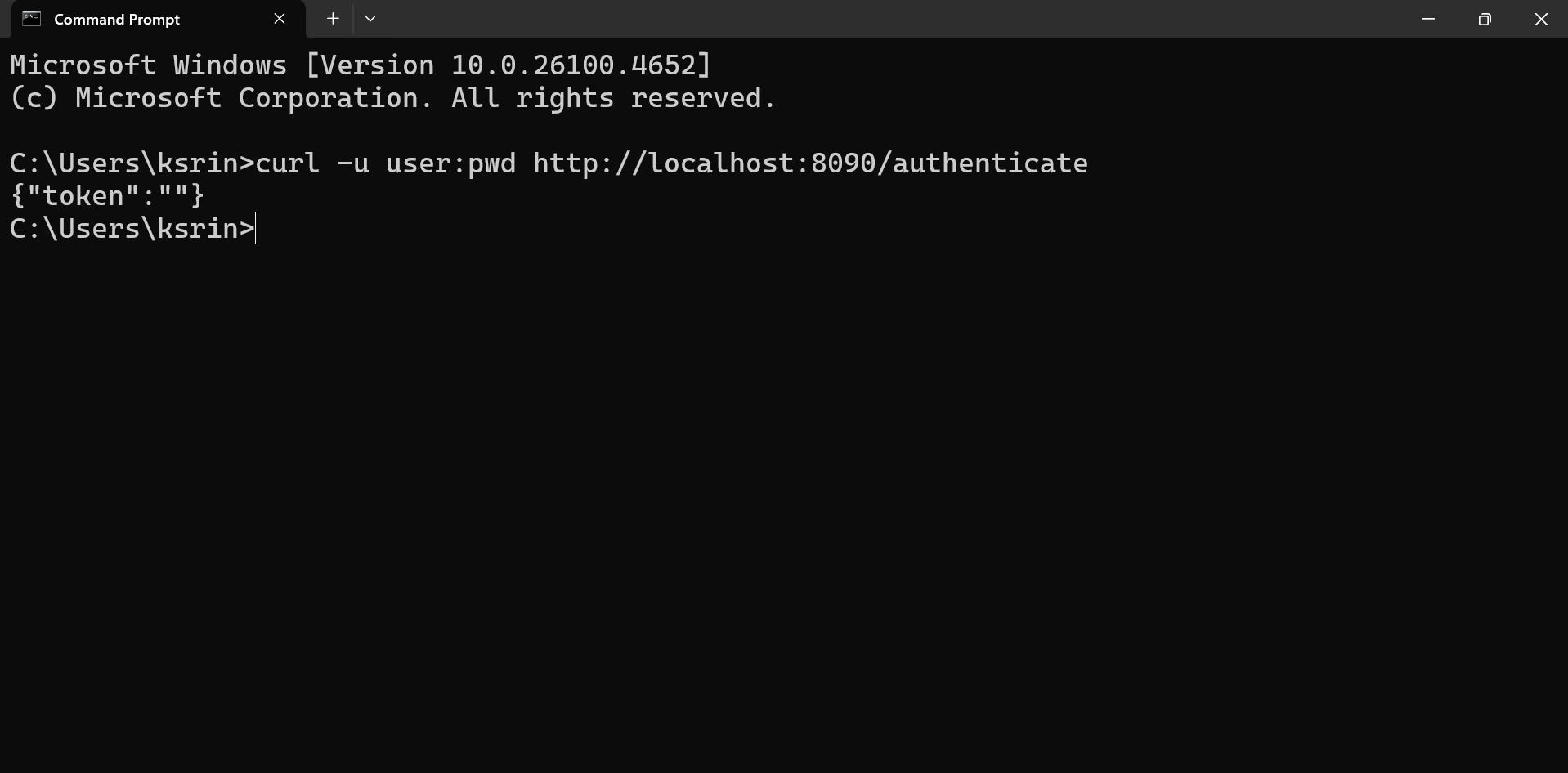
<groupId>org.slf4j</groupId>

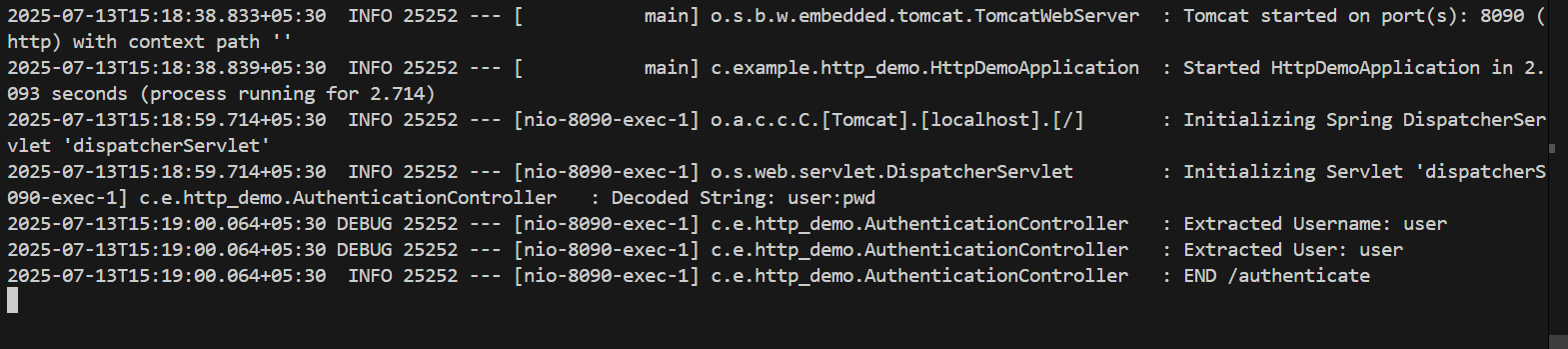
<artifactId>slf4j-api</artifactId>

</dependency>

</dependencies>

**Output:**





**Generate token based on the user**  
**HttpDemoApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HttpDemoApplication {

public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**AuthenticationController.java**

package com.example.http\_demo;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import io.jsonwebtoken.security.Keys;

import jakarta.annotation.PostConstruct;

import jakarta.servlet.http.HttpServletRequest;

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

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

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import java.util.Base64;

import java.util.Date;

@RestController

public class AuthenticationController {

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

private String secret;

private SecretKey secretKey;

@PostConstruct

public void init() {

this.secretKey = new SecretKeySpec(secret.getBytes(), SignatureAlgorithm.HS256.getJcaName());

}

@GetMapping("/authenticate")

public String authenticate(HttpServletRequest request) {

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

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

throw new RuntimeException("Missing or invalid Authorization header.");

}

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

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

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

String username = credentials[0];

String password = credentials[1];

if ("user".equals(username) && "pwd".equals(password)) {

return generateJwt(username);

} else {

throw new RuntimeException("Invalid credentials");

}

}

public String generateJwt(String username) {

long expirationTime = 1000 \* 60 \* 60;

return Jwts.builder()

.setSubject(username)

.setExpiration(new Date(System.currentTimeMillis() + expirationTime))

.signWith(secretKey, SignatureAlgorithm.HS256)

.compact();

}

}

**SecurityConfig.java**

package com.example.http\_demo;

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.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(authz -> authz

.anyRequest().authenticated()

)

.httpBasic()

.and()

.csrf().disable();

return http.build();

}

}

**application.properties**

jwt.secret=mysupersecretkey1234567890abcd

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>http-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>http-demo</name>

<description>HTTP Demo</description>

<properties>

<java.version>21</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-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>

</dependencies>

<build>

<plugins>

<plugin>

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

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

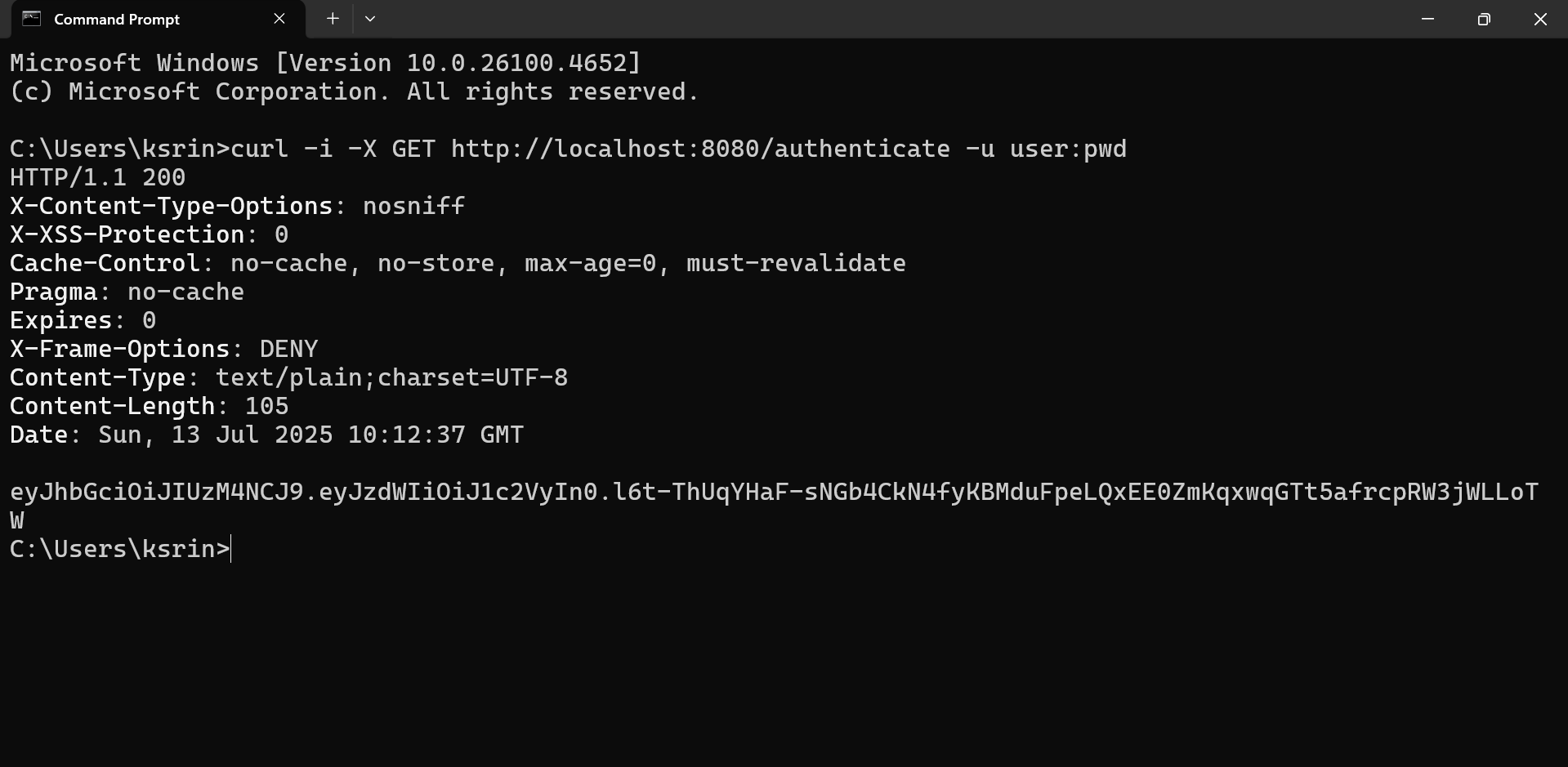
</plugin>

</plugins>

</build>

</project>

**Output:**



**Authorize based on JWT**

**HttpDemoApplication.java**

package com.example.http\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class HttpDemoApplication {

public static void main(String[] args) {

SpringApplication.run(HttpDemoApplication.class, args);

}

}

**SecurityConfig.java**

package com.example.http\_demo;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

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

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

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public UserDetailsService userDetailsService(PasswordEncoder encoder) {

UserDetails user = User.withUsername("user")

.password(encoder.encode("pwd"))

.roles("USER")

.build();

UserDetails admin = User.withUsername("admin")

.password(encoder.encode("pwd"))

.roles("ADMIN")

.build();

return new InMemoryUserDetailsManager(user, admin);

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

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

AuthenticationManagerBuilder auth = http.getSharedObject(AuthenticationManagerBuilder.class);

auth.userDetailsService(userDetailsService).passwordEncoder(encoder);

return auth.build();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http, AuthenticationManager authManager) throws Exception {

JwtAuthorizationFilter jwtFilter = new JwtAuthorizationFilter(authManager);

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

.authorizeHttpRequests(auth -> auth

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

.requestMatchers("/employees").hasAnyRole("USER", "ADMIN")

.anyRequest().authenticated()

)

.addFilter(jwtFilter);

return http.build();

}

}

**JwtUtil.java**

package com.example.http\_demo;

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";

private final long expirationMs = 1000 \* 60 \* 60;

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + expirationMs))

.signWith(SignatureAlgorithm.HS256, secret)

.compact();

}

public String validateToken(String token) {

return Jwts.parser()

.setSigningKey(secret)

.parseClaimsJws(token)

.getBody()

.getSubject();

}

}

**JwtAuthorizationFilter.java**

package com.example.http\_demo;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

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

import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

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

import java.io.IOException;

public class JwtAuthorizationFilter extends BasicAuthenticationFilter {

@Autowired

private JwtUtil jwtUtil;

@Autowired

private UserDetailsService userDetailsService;

public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

super(authenticationManager);

}

@Override

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain chain)

throws IOException, ServletException {

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

if (header == null || !header.startsWith("Bearer ")) {

chain.doFilter(request, response);

return;

}

String token = header.replace("Bearer ", "");

String username = jwtUtil.validateToken(token);

if (username != null) {

var userDetails = userDetailsService.loadUserByUsername(username);

var auth = new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());

SecurityContextHolder.getContext().setAuthentication(auth);

}

chain.doFilter(request, response);

}

}

**AuthenticationController.java**

package com.example.http\_demo;

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

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

import java.util.Map;

@RestController

public class AuthenticationController {

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private JwtUtil jwtUtil;

@PostMapping("/authenticate")

public Map<String, String> authenticate(@RequestBody Map<String, String> credentials) {

authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(credentials.get("username"), credentials.get("password"))

);

String token = jwtUtil.generateToken(credentials.get("username"));

return Map.of("token", token);

}

}

**EmployeeController.java**

package com.example.http\_demo;

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

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

import java.util.List;

@RestController

public class EmployeeController {

@GetMapping("/employees")

public List<String> getEmployees() {

return List.of("John", "Jane", "Alice", "Bob");

}

}

**pom.xml**

<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</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

**Output:**



