**Hands on 1**

**Create a Spring Web Project using Maven**   
  
Follow steps below to create a project: 

1. Go to <https://start.spring.io/>
2. Change Group as “com.cognizant”
3. Change Artifact Id as “spring-learn”
4. Select Spring Boot DevTools and Spring Web
5. Create and download the project as zip
6. Extract the zip in root folder to Eclipse Workspace
7. Build the project using ‘mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456’ command in command line
8. Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"
9. Include logs to verify if main() method of SpringLearnApplication.
10. Run the SpringLearnApplication class.

SME to walk through the following aspects related to the project created:

1. src/main/java - Folder with application code
2. src/main/resources - Folder for application configuration
3. src/test/java - Folder with code for testing the application
4. SpringLearnApplication.java - Walkthrough the main() method.
5. Purpose of @SpringBootApplication annotation
6. pom.xml
   1. Walkthrough all the configuration defined in XML file
   2. Open 'Dependency Hierarchy' and show the dependency tree.

**OUTPUT:**

**SME Walkthrough Points**

**1. src/main/java**

* Contains the **main application code**.
* Typically includes:
  + SpringLearnApplication.java
  + Controllers, Services, Models, etc.

**2. src/main/resources**

* For application configuration files.
* Common files:
  + application.properties or application.yml
  + Templates, static content, and messages

**3. src/test/java**

* For **unit and integration tests**.
* JUnit-based test classes are placed here.

**4. SpringLearnApplication.java**

java

CopyEdit

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

* This is the **entry point** for Spring Boot.

**5. @SpringBootApplication Annotation**

This annotation is a **convenience annotation** that combines:

* @Configuration – Declares this class as a config class.
* @EnableAutoConfiguration – Enables auto configuration based on dependencies.
* @ComponentScan – Scans current package and subpackages for components.

**6. pom.xml**

**Sample Breakdown:**

xml

CopyEdit

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

<description>Demo project for Spring Boot</description>

<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>

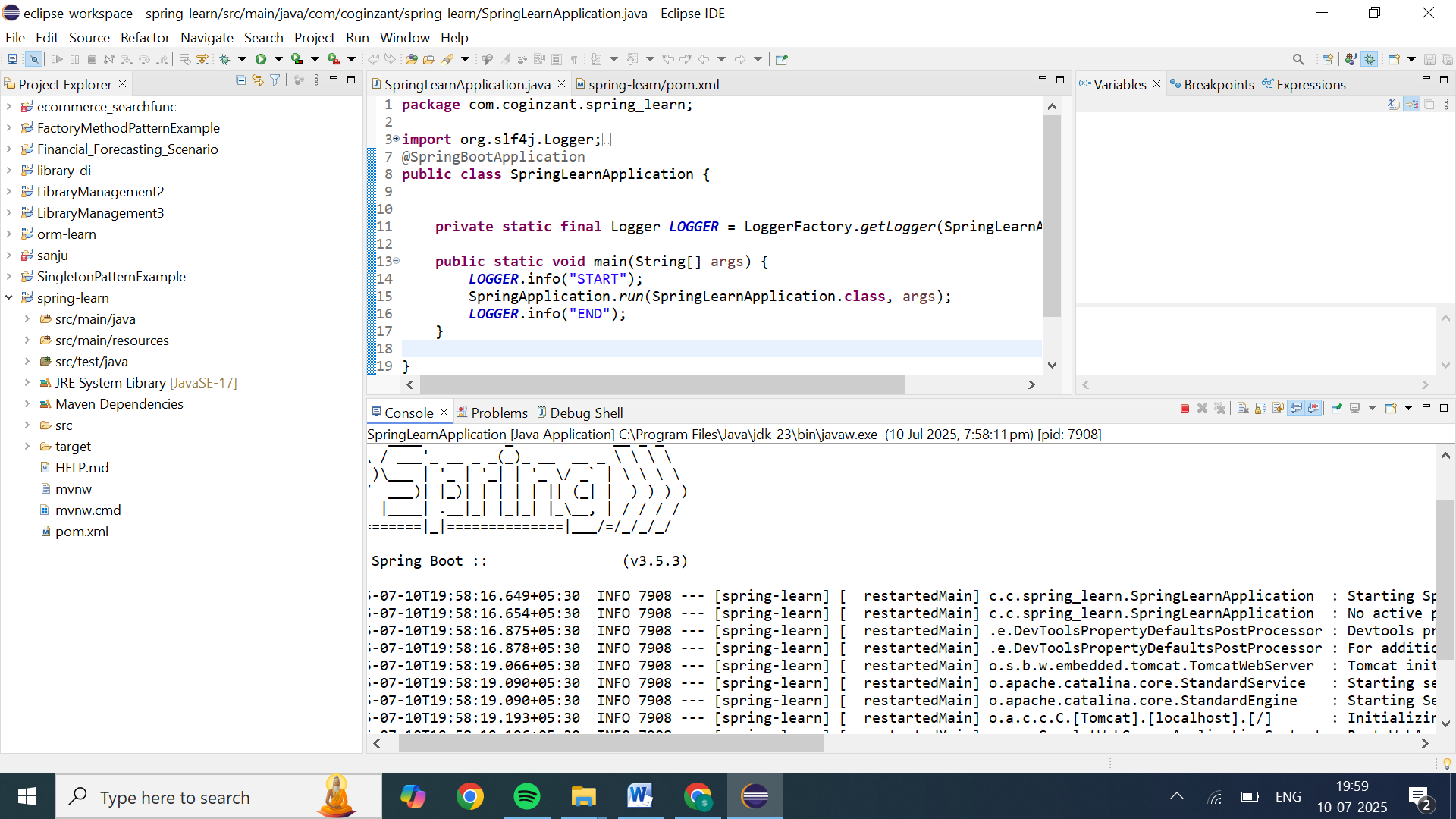
<scope>test</scope>

</dependency>

</dependencies>

**7. Dependency Hierarchy in Eclipse**

* Right-click the project in Eclipse > **Maven > Show Dependency Hierarchy**
* You can see the **transitive dependencies** of:
  + spring-boot-starter-web (which includes spring-web, spring-core, etc.)
  + spring-boot-devtools
  + spring-boot-starter-test



Exact output displayed is:

19:58:14.844 [main] INFO com.coginzant.spring\_learn.SpringLearnApplication -- START

19:58:15.456 [restartedMain] INFO com.coginzant.spring\_learn.SpringLearnApplication -- START

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

=========|\_|==============|\_\_\_/=/\_/\_/\_/

:: Spring Boot :: (v3.5.3)

2025-07-10T19:58:16.649+05:30 INFO 7908 --- [spring-learn] [ restartedMain] c.c.spring\_learn.SpringLearnApplication : Starting SpringLearnApplication using Java 23.0.1 with PID 7908 (C:\Users\HOME\eclipse-workspace\spring-learn\target\classes started by HOME in C:\Users\HOME\eclipse-workspace\spring-learn)

2025-07-10T19:58:16.654+05:30 INFO 7908 --- [spring-learn] [ restartedMain] c.c.spring\_learn.SpringLearnApplication : No active profile set, falling back to 1 default profile: "default"

2025-07-10T19:58:16.875+05:30 INFO 7908 --- [spring-learn] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable

2025-07-10T19:58:16.878+05:30 INFO 7908 --- [spring-learn] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' property to 'DEBUG'

2025-07-10T19:58:19.066+05:30 INFO 7908 --- [spring-learn] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)

2025-07-10T19:58:19.090+05:30 INFO 7908 --- [spring-learn] [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]

2025-07-10T19:58:19.090+05:30 INFO 7908 --- [spring-learn] [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]

2025-07-10T19:58:19.193+05:30 INFO 7908 --- [spring-learn] [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext

2025-07-10T19:58:19.196+05:30 INFO 7908 --- [spring-learn] [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2307 ms

2025-07-10T19:58:19.909+05:30 INFO 7908 --- [spring-learn] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729

2025-07-10T19:58:19.970+05:30 INFO 7908 --- [spring-learn] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'

2025-07-10T19:58:19.996+05:30 INFO 7908 --- [spring-learn] [ restartedMain] c.c.spring\_learn.SpringLearnApplication : Started SpringLearnApplication in 4.494 seconds (process running for 5.882)

2025-07-10T19:58:20.004+05:30 INFO 7908 --- [spring-learn] [ restartedMain] c.c.spring\_learn.SpringLearnApplication : END

**Hands on 2**

**Spring Core – Load SimpleDateFormat from Spring Configuration XML**   
  
SimpleDateFormat with the pattern ‘dd/MM/yyyy’ is created in multiple places of an application. To avoid creation of SimpleDateFormat in multiple places, define a bean in Spring XML Configuration file and retrieve the date.  
  
Follow steps below to implement:

* Create spring configuration file date-format.xml in src/main/resources folder of 'spring-learn' project
* Open https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-factory-metadata
* Copy the XML defined in the section of previous step URL and paste it into date-format.xml
* Define bean tag in the XML with for date format. Refer code below.

<?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>

* Create new method displayDate() in SpringLearnApplication.java
* In displayDate() method create the ApplicationContext. Refer code below:

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

* Get the dateFormat using getBean() method. Refer code below.

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

* Using the format variable try to parse string '31/12/2018' to Date class and display the result using System.out.println.
* Run the application as 'Java Application' and check the result in console log output.

**Troubleshooting Tips**   
  
If the tomcat port has a conflict and the server is not starting include the below property in application.properties file in src/main/resources folder.

### 1. Create the Spring XML config

**src/main/resources/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>

### 2. Load and use the bean in Java

In your main class, say SpringLearnApplication.java, add the following method:

**package** com.coginzant.spring\_learn;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**public** **class** SpringLearnApplication {

**public** **static** **void** main(String[] args) {

*displayDate*();

}

**public** **static** **void** displayDate() {

**try** (ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("date-format.xml")) {

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

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

System.***out***.println(date);

} **catch** (Exception e) {

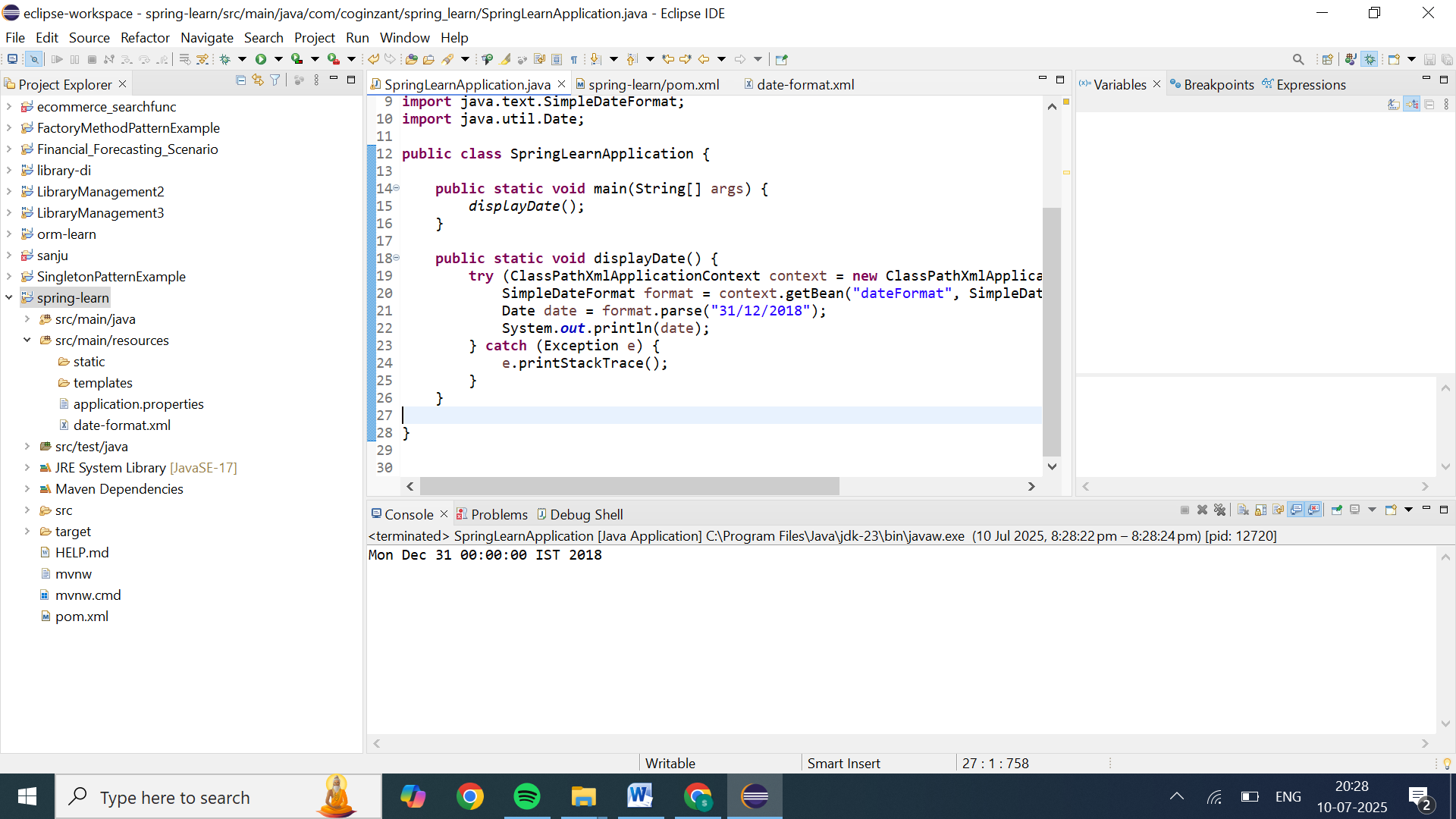
e.printStackTrace();

}

}

}

OUTPUT:



**Hands on 3**

**Hello World RESTful Web Service**   
  
Write a REST service in the spring learn application created earlier, that returns the text "Hello World!!" using Spring Web Framework. Refer details below:  
  
**Method:** GET  
**URL:** /hello  
**Controller:** com.cognizant.spring-learn.controller.HelloController  
**Method Signature:** public String sayHello()  
**Method Implementation:** return hard coded string "Hello World!!"  
**Sample Request**: http://localhost:8083/hello  
**Sample Response:** Hello World!!   
  
**IMPORTANT NOTE**: Don't forget to include start and end log in the sayHello() method.  
  
Try the URL http://localhost:8083/hello in both chrome browser and postman.  
  
SME to explain the following aspects:

* In network tab of developer tools show the HTTP header details received
* In postman click on "Headers" tab to view the HTTP header details received

### Step 1: Create the Controller

In your Spring Boot application, create a new controller class:

**package** com.cognizant.spring\_learn.controller;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

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

@RestController

**public** **class** HelloController {

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

@GetMapping("/hello")

**public** String sayHello() {

***logger***.info("Start of sayHello()");

***logger***.info("End of sayHello()");

**return** "Hello World!!";

}

}

}

### Step 2: Run the Application

Ensure your main application class is properly set up

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.boot.CommandLineRunner;

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

@SpringBootApplication

@ComponentScan("com.cognizant.spring\_learn")

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

@Bean

public CommandLineRunner showMappings(RequestMappingHandlerMapping mapping) {

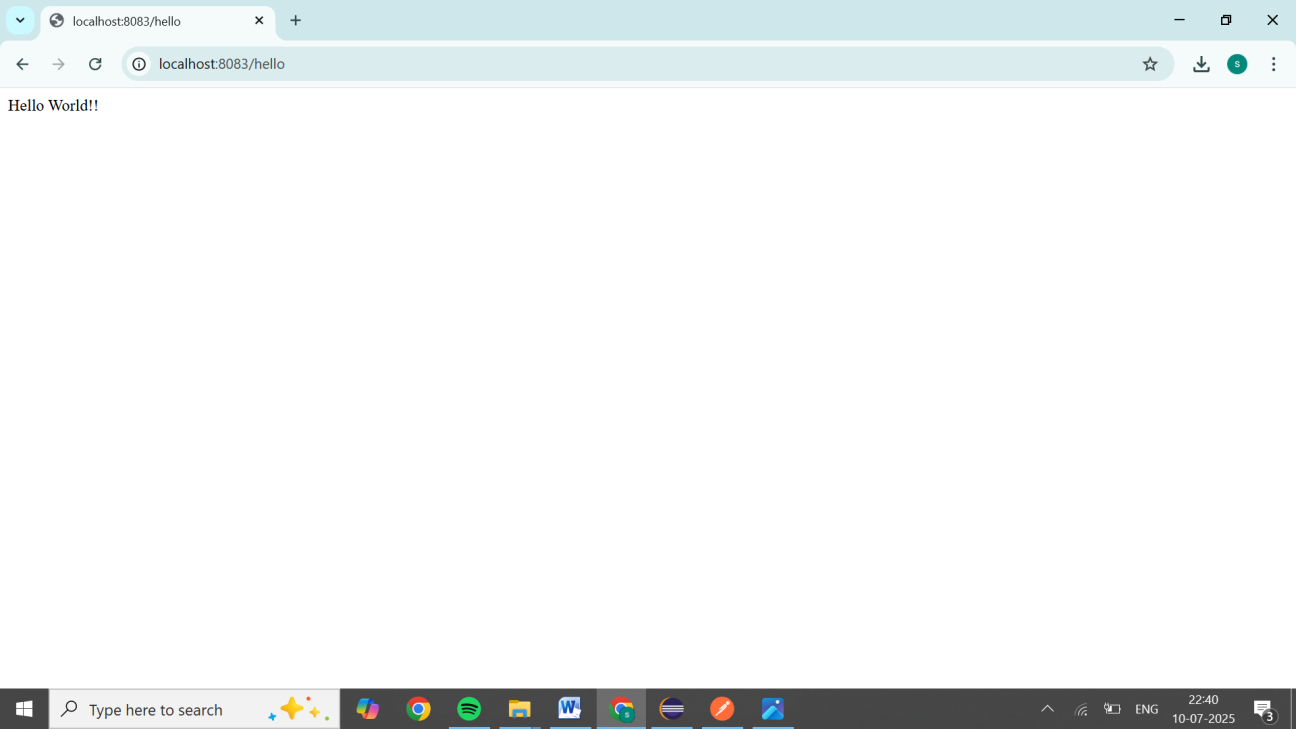
return args -> mapping.getHandlerMethods().keySet()

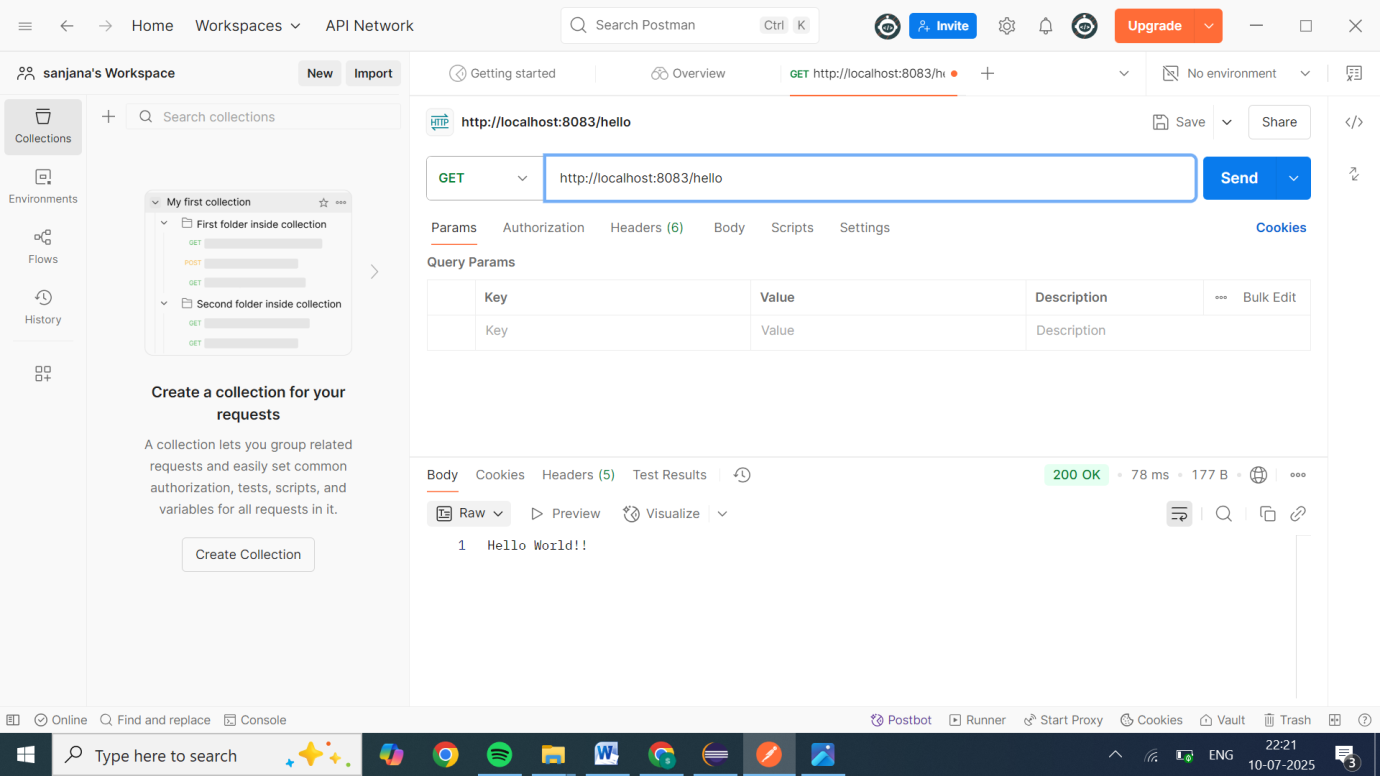
.forEach(info -> System.out.println("Mapped: " + info));

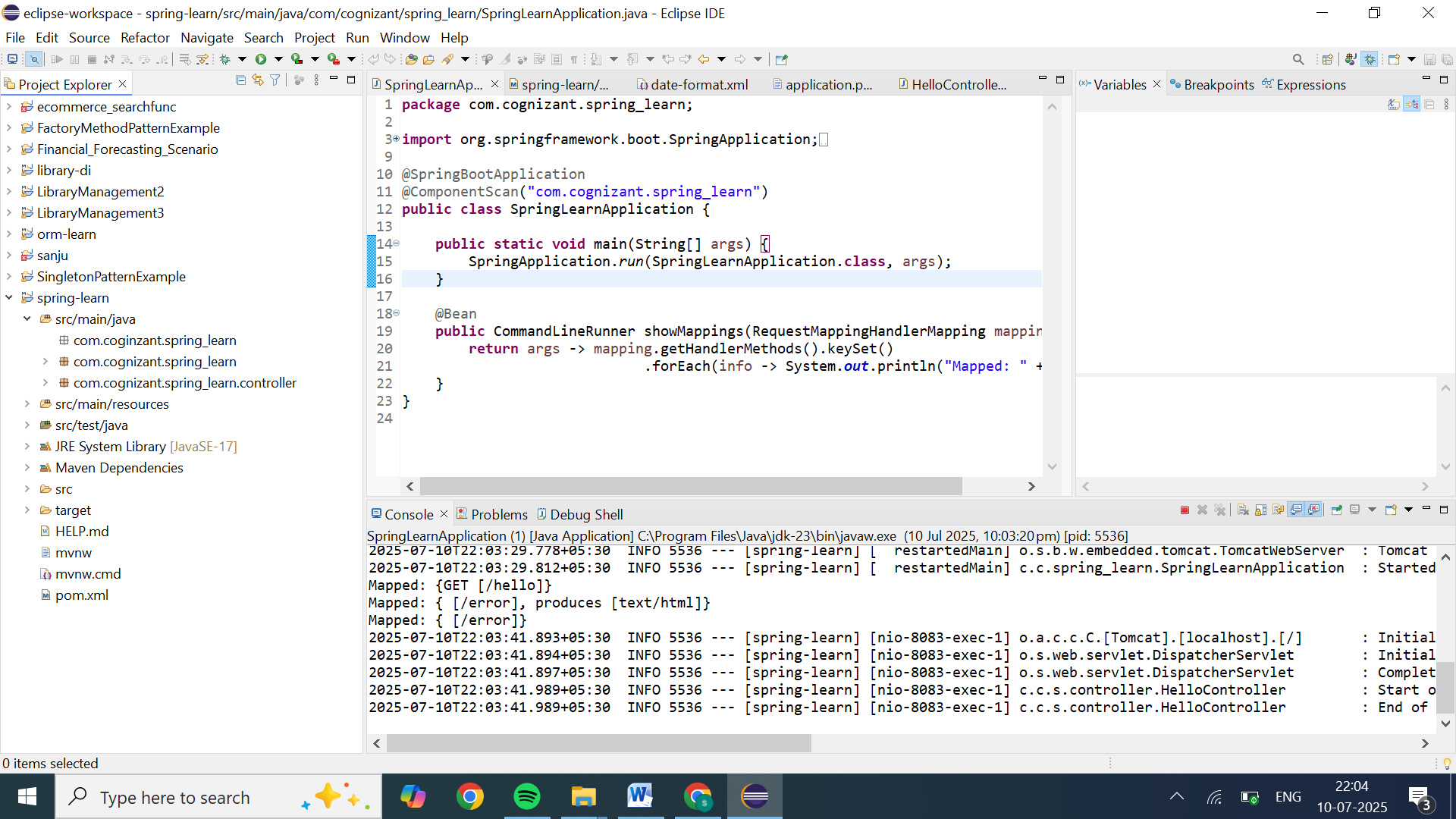
}

#### In Chrome Browser:

1. Open Chrome and navigate to http://localhost:8083/hello.
2. You should see the response: Hello World!!







**Hands on 4**

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

{

  "code": "IN",

  "name": "India"

}

SME to explain the following aspects:

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

Code:xml file

<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">

<!-- Define a String bean -->

<bean id="inCode" class="java.lang.String">

<constructor-arg value="IN"/>

</bean>

<!-- Define the Country bean -->

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

<constructor-arg ref="inCode"/>

<constructor-arg value="India"/>

</bean>

</beans>

Spring application coe:

**package** com.cognizant.spring\_learn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.context.ApplicationContext;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

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

**import** org.springframework.boot.CommandLineRunner;

**import** org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping;

@SpringBootApplication

@ImportResource("classpath:applicationContext.xml")

**public** **class** SpringLearnApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringLearnApplication.**class**, args);

}

@Bean

**public** CommandLineRunner showMappings(RequestMappingHandlerMapping mapping) {

**return** args -> mapping.getHandlerMethods().keySet()

.forEach(info -> System.***out***.println("Mapped: " + info));

}

}

Country context code:

**package** com.cognizant.spring\_learn.controller;

**import** org.springframework.context.ApplicationContext;

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

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

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

**import** com.cognizant.spring\_learn.model.Country;

@RestController

**public** **class** CountryController {

@Autowired

**private** ApplicationContext context;

@RequestMapping("/country")

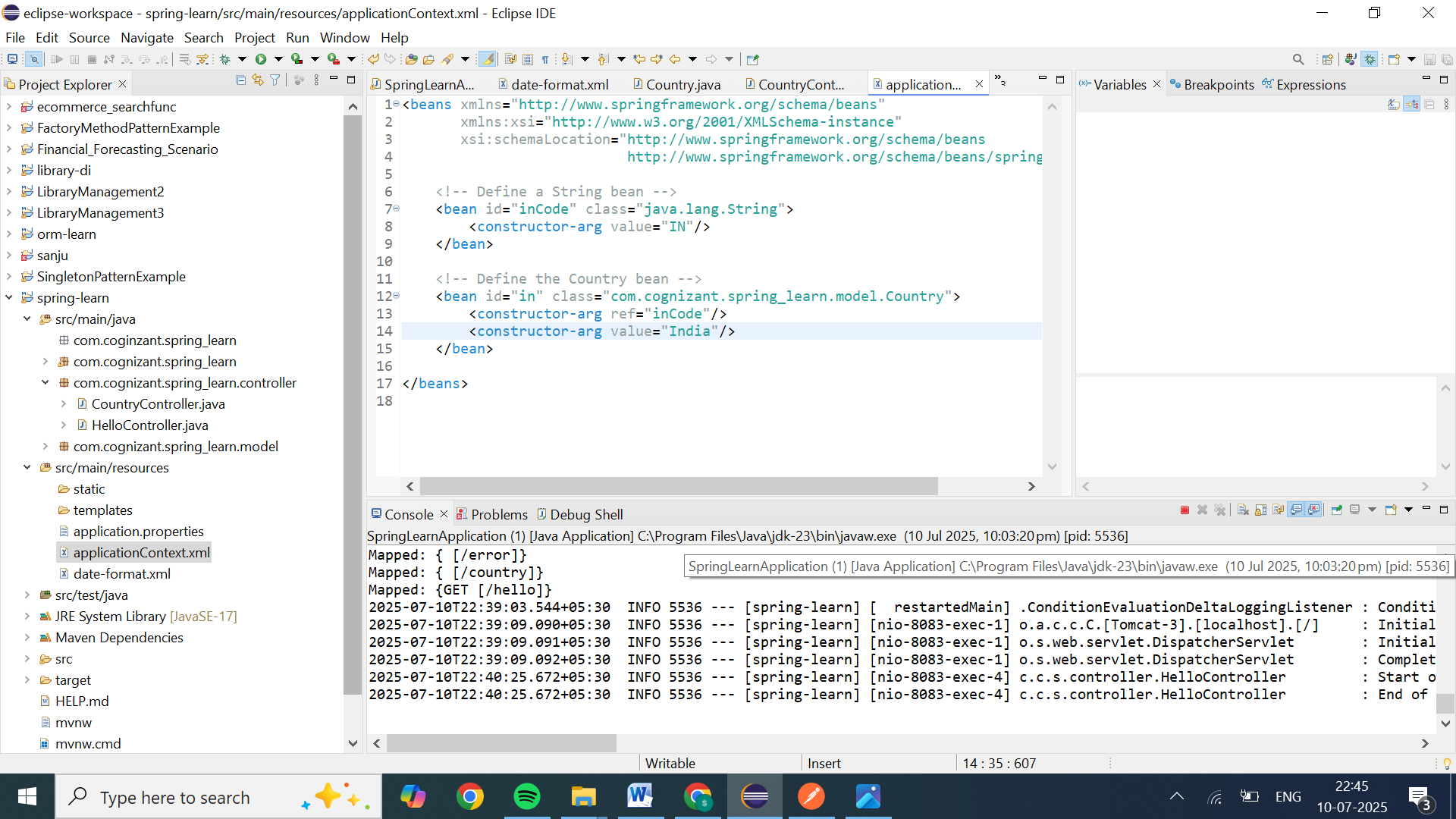
**public** Country getCountryIndia() {

**return** context.getBean("in", Country.**class**);

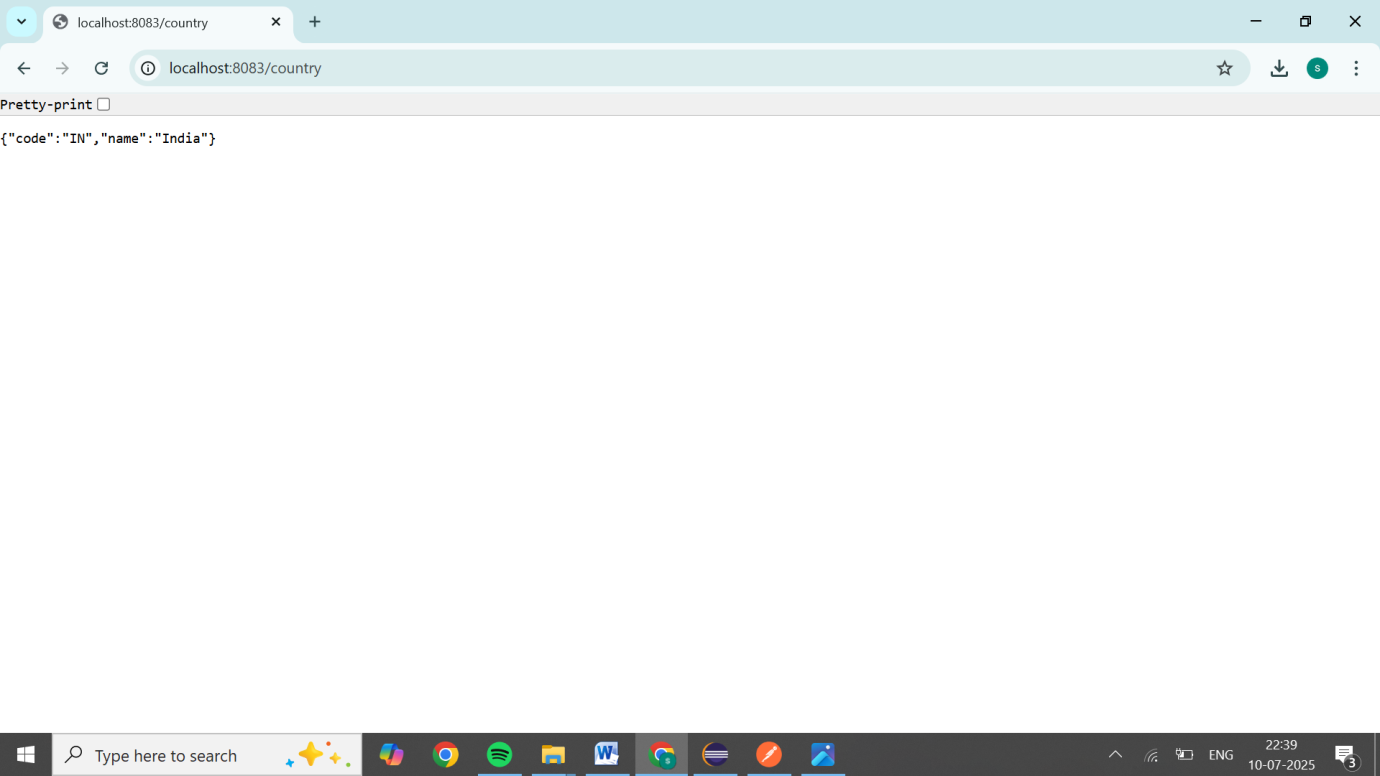
}

}

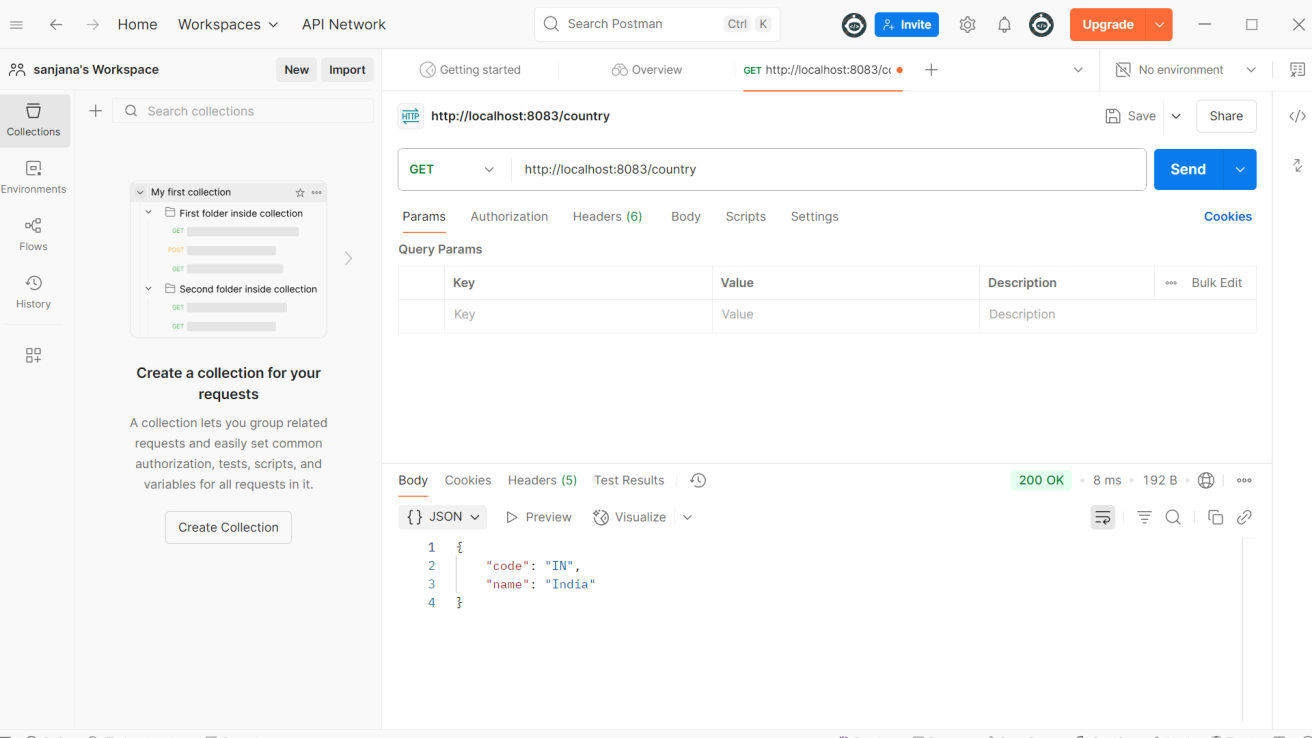
ECLIPSE OUTPUT:



CHROME OUTPUT:



POSTMAN OUTPUT:



**Hands on 5**

**REST - Get country based on country code**   
  
Write a REST service that returns a specific country based on country code. The country code should be case insensitive.  
**Controller**: com.cognizant.spring-learn.controller.CountryController  
**Method Annotation:** @GetMapping("/countries/{code}")  
**Method Name**: getCountry(String code)  
**Method Implemetation**: Invoke countryService.getCountry(code)   
**Service Method:**com.cognizant.spring-learn.service.CountryService.getCountry(String code)  
**Service Method Implementation**:

* Get the country code using @PathVariable
* Get country list from country.xml
* Iterate through the country list
* Make a case insensitive matching of country code and return the country.
* Lambda expression can also be used instead of iterating the country list

**Sample Request**: http://localhost:8083/country/in  
**Sample Response**:

{

  "code": "IN",

  "name": "India"

}

CountryController code:

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.model.Country;

import com.cognizant.spring\_learn.service.CountryService;

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

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

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

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

@RestController

public class CountryController {

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

return countryService.getCountry(code);

}

}

Country :

**package** com.cognizant.spring\_learn.model;

**public** **class** Country {

**private** String code;

**private** String name;

// Constructor

**public** Country(String code, String name) {

**this**.code = code;

**this**.name = name;

}

// Getters and setters

**public** String getCode() {

**return** code;

}

**public** **void** setCode(String code) {

**this**.code = code;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

Country.xml:

<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">

<!-- Define a String bean -->

<bean id="inCode" class="java.lang.String">

<constructor-arg value="IN"/>

</bean>

<!-- Define the Country bean -->

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

<constructor-arg ref="inCode"/>

<constructor-arg value="India"/>

</bean>

</beans>

Springlearn apllication:

**package** com.cognizant.spring\_learn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

@SpringBootApplication(exclude = {DataSourceAutoConfiguration.**class** })

**public** **class** SpringLearnApplication {

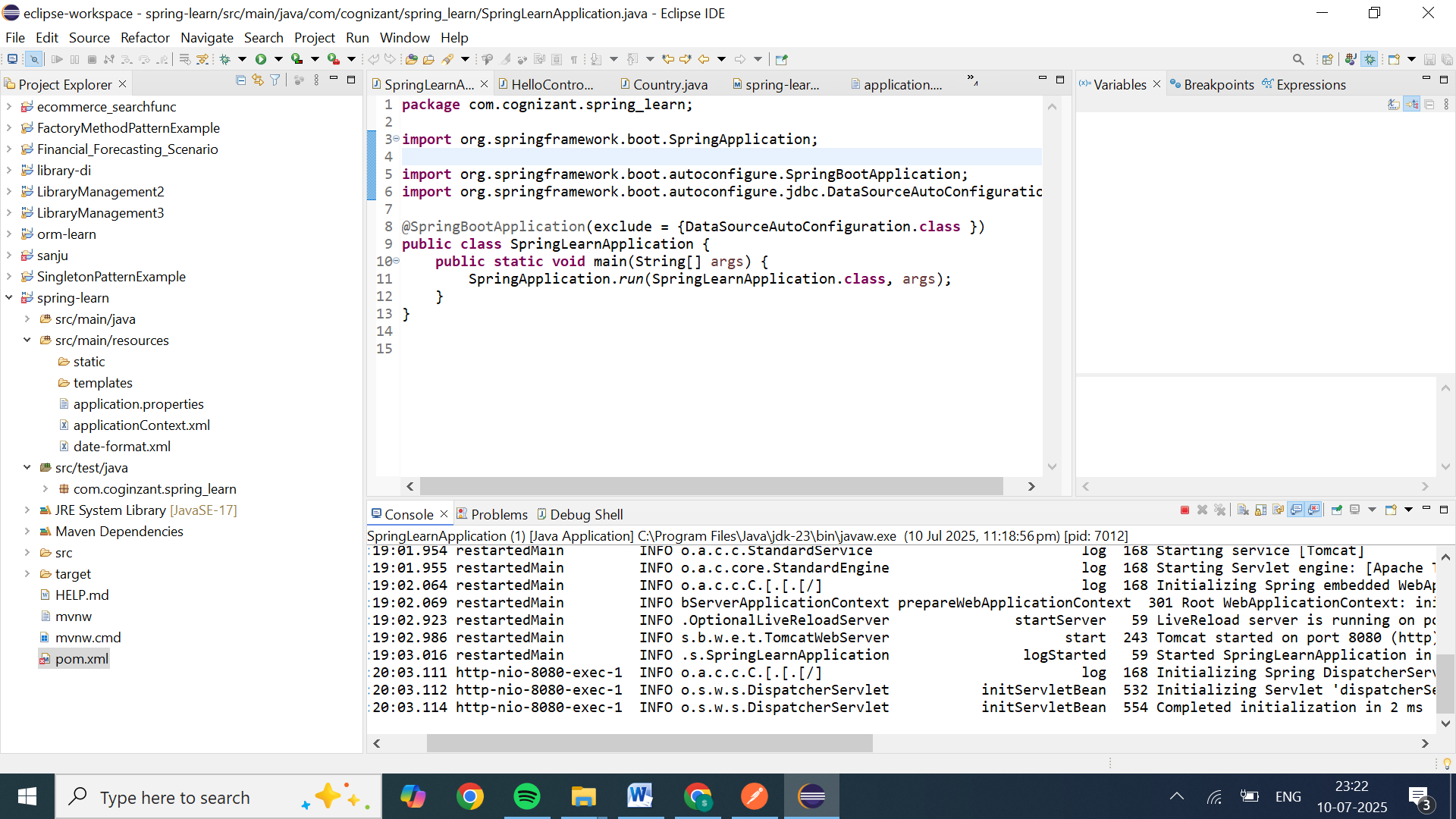
**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringLearnApplication.**class**, args);

}

}

JAVA OUTPUT:



CHROME OUTPUT

