**Mandatory Hands On**

**Spring REST using Spring Boot 3**

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

**Follow steps below to create a project:**

**Go to**[**https://start.spring.io/**](https://start.spring.io/)

1. **Change Group as “com.cognizant”**
2. **Change Artifact Id as “spring-learn”**
3. **Select Spring Boot DevTools and Spring Web**
4. **Create and download the project as zip**
5. **Extract the zip in root folder to Eclipse Workspace**
6. **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**
7. **Import the project in Eclipse "File > Import > Maven > Existing Maven Projects > Click Browse and select extracted folder > Finish"**
8. **Include logs to verify if main() method of SpringLearnApplication.**
9. **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.**

**Code:**

SpringLearnApplication.java(main) :-

package com.cognizant.spring\_learn;

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) {

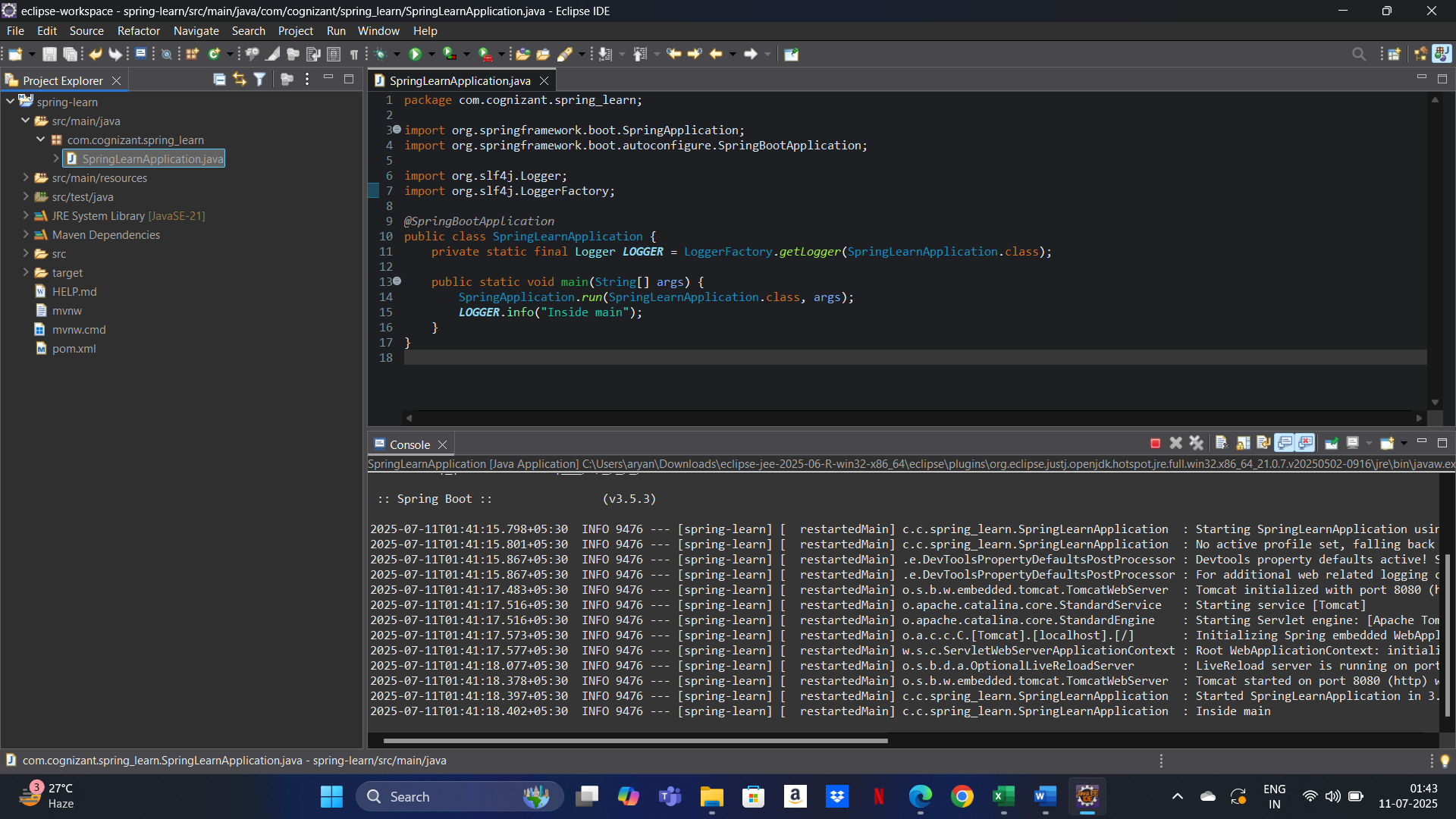
SpringApplication.run(SpringLearnApplication.class, args);

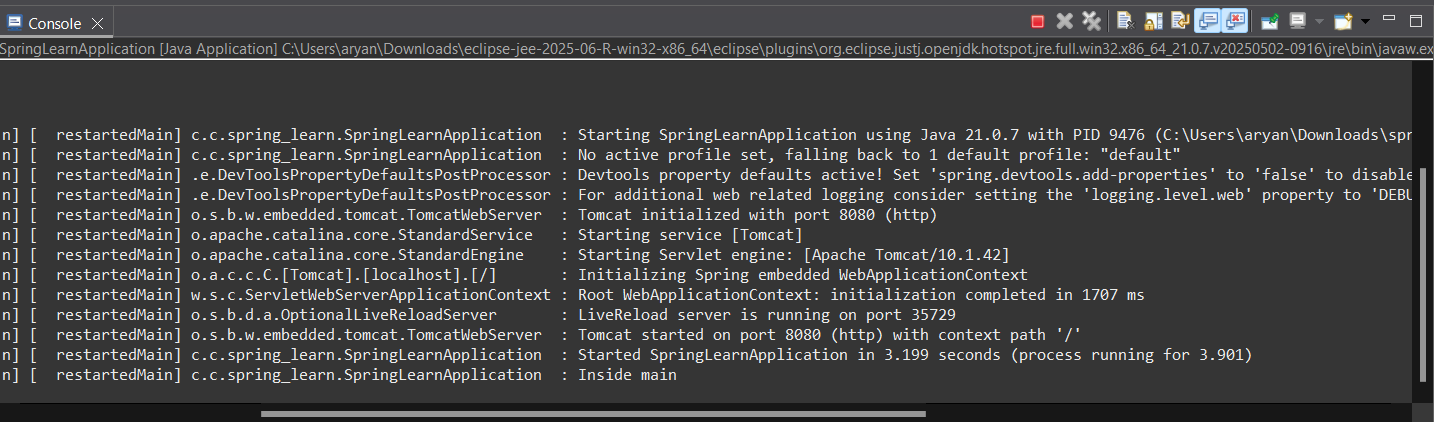
LOGGER.info("Inside main");

}

}

**Output:**

****

****

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

**An airlines website is going to support booking on four countries. There will be a drop down on the home page of this website to select the respective country. It is also important to store the two-character ISO code of each country.**

|  |  |
| --- | --- |
| **Code** | **Name** |
| **US** | **United States** |
| **DE** | **Germany** |
| **IN** | **India** |
| **JP** | **Japan** |

**Above data has to be stored in spring configuration file. Write a program to read this configuration file and display the details.  
  
Steps to implement**

* **Pick any one of your choice country to configure in Spring XML configuration named country.xml.**
* **Create a bean tag in spring configuration for country and set the property and values**

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

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

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

**</bean>**

* **Create Country class with following aspects:**
  + **Instance variables for code and name**
  + **Implement empty parameter constructor with inclusion of debug log within the constructor with log message as “Inside Country Constructor.”**
  + **Generate getters and setters with inclusion of debug with relevant message within each setter and getter method.**
  + **Generate toString() method**
* **Create a method displayCountry() in SpringLearnApplication.java, which will read the country bean from spring configuration file and display the country details. ClassPathXmlApplicationContext, ApplicationContext and context.getBean(“beanId”, Country.class). Refer sample code for displayCountry() method below.**

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

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

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

* **Invoke displayCountry() method in main() method of SpringLearnApplication.java.**
* **Execute main() method and check the logs to find out which constructors and methods were invoked.**

**SME to provide more detailing about the following aspects:**

* **bean tag, id attribute, class attribute, property tag, name attribute, value attribute**
* **ApplicationContext, ClassPathXmlApplicationContext**

**What exactly happens when context.getBean() is invoked**

**Code:**

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>

Country.java :-

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class Country {

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

private String code;

private String name;

public Country() {

LOGGER.debug("Inside Country Constructor.");

}

public String getCode() {

LOGGER.debug("Getting country code.");

return code;

}

public void setCode(String code) {

LOGGER.debug("Setting country code.");

this.code = code;

}

public String getName() {

LOGGER.debug("Getting country name.");

return name;

}

public void setName(String name) {

LOGGER.debug("Setting country name.");

this.name = name;

}

@Override

public String toString() {

return "Country{" +

"code='" + code + '\'' +

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

'}';

}

}

SpringLearnApplication.java(main) :-

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearnApplication {

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

public static void main(String[] args) {

displayCountry();

}

public static void displayCountry() {

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

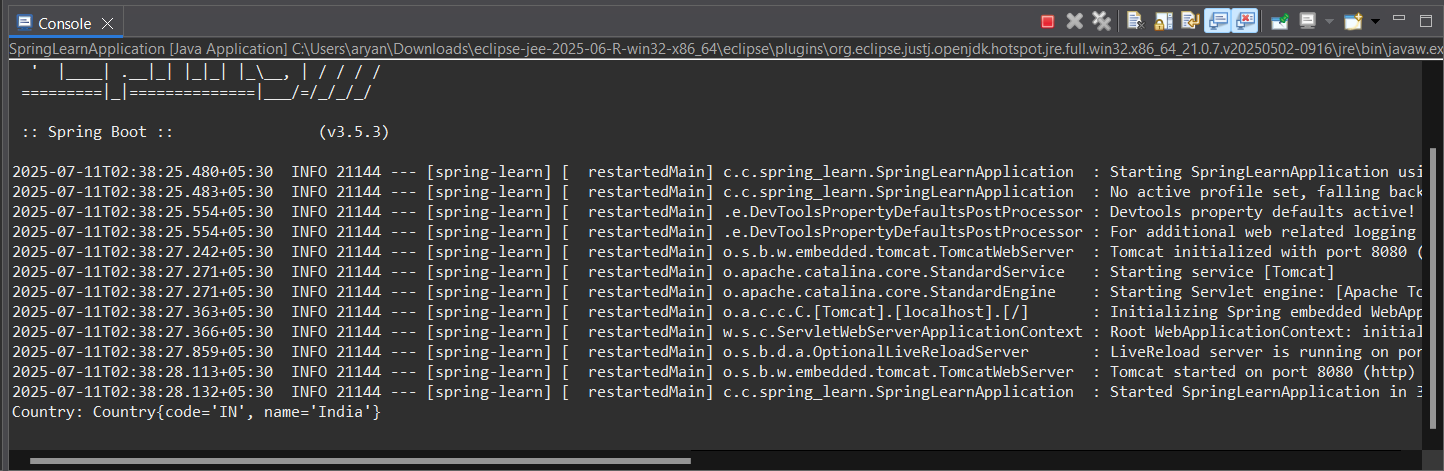
Country country = context.getBean("country", Country.class);

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

}

}

**Output:**

****

**Exercise 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**

**Code:**

HelloController.java :-

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class HelloController {

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

@GetMapping("/hello")

public String sayHello() {

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

String response = "Hello World!!";

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

return response;

}

}

SpringLearnApplication.java(main) :-

package com.cognizant.springlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

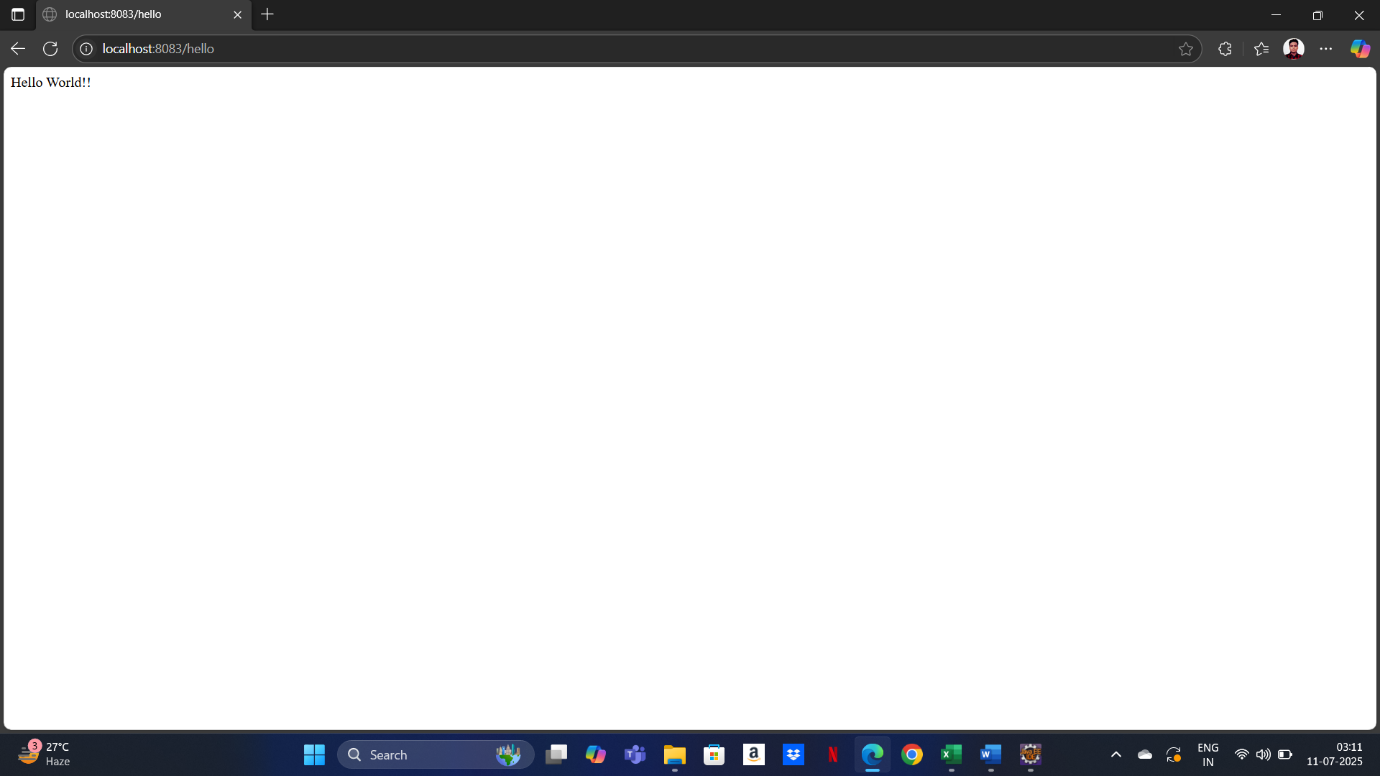
public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Output:**



**Exercise 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:**

Country.java :-

package com.cognizant.spring\_learn;

public class Country {

private String code;

private String name;

// Getters and setters

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

CountryConfig.java :-

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.Country;

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

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

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

*@RestController*

public class CountryController {

*@Autowired*

private Country country;

*@RequestMapping*("/country")

public Country getCountryIndia() {

return country;

}

}

SpringLearnApplication.java(main) :-

package com.cognizant.spring\_learn;

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="country" class="com.cognizant.spring\_learn.model.Country">

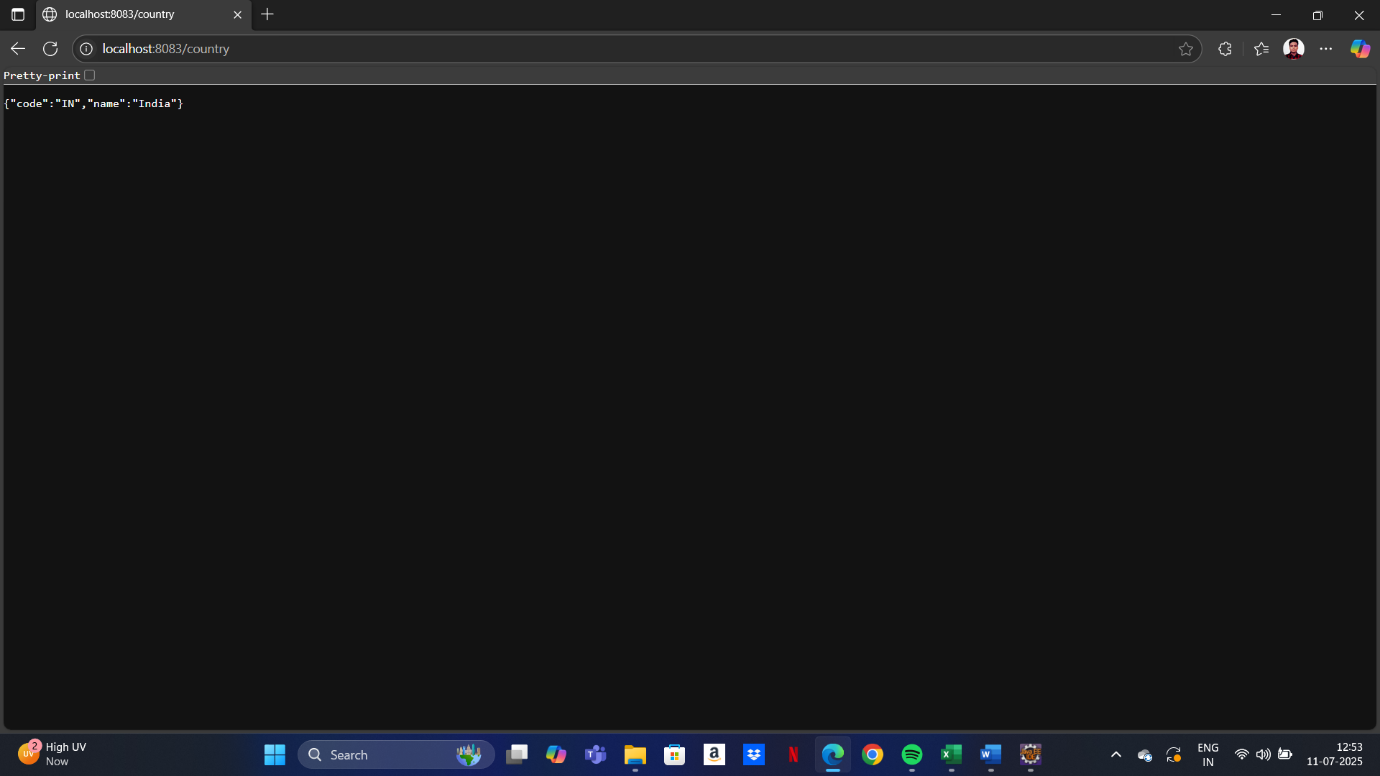
<property name="code" value="IN"/>

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

</bean>

</beans>

**Output:**

****

**Exercise 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"**

**}**

**Code:**

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="country" class="com.cognizant.spring\_learn.model.Country">

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

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

</bean>

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

<constructor-arg>

<list>

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

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

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

</bean>

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

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

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

</bean>

<!-- Add more countries if needed -->

</list>

</constructor-arg>

</bean>

</beans>

Country.java :-

package com.cognizant.spring\_learn;

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 String getName() {

return name;

}

public void setCode(String code) {

this.code = code;

}

public void setName(String name) {

this.name = name;

}

}

CountryService :-

package com.cognizant.spring\_learn.service;

import com.cognizant.spring\_learn.Country;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

*@Service*

public class CountryService {

private List<Country> countries;

public CountryService() {

// Simulated hard-coded list instead of loading from XML

countries = new ArrayList<>();

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

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

countries.add(new Country("CN", "China"));

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

countries.add(new Country("FR", "France"));

}

public Country getCountry(String code) {

return countries.stream()

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

.findFirst()

.orElse(null);

}

}

CountryController.java :-

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.Country;

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

*@RestController*

public class CountryController {

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

*@Autowired*

private CountryService countryService;

*@GetMapping*("/country/{code}")

public Country getCountry(*@PathVariable* String code) {

***LOGGER***.info("START getCountry with code: {}", code);

Country country = countryService.getCountry(code);

***LOGGER***.info("END getCountry");

return country;

}

}

SpringLearnApplication.java(main) :-

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class SpringLearnApplication {

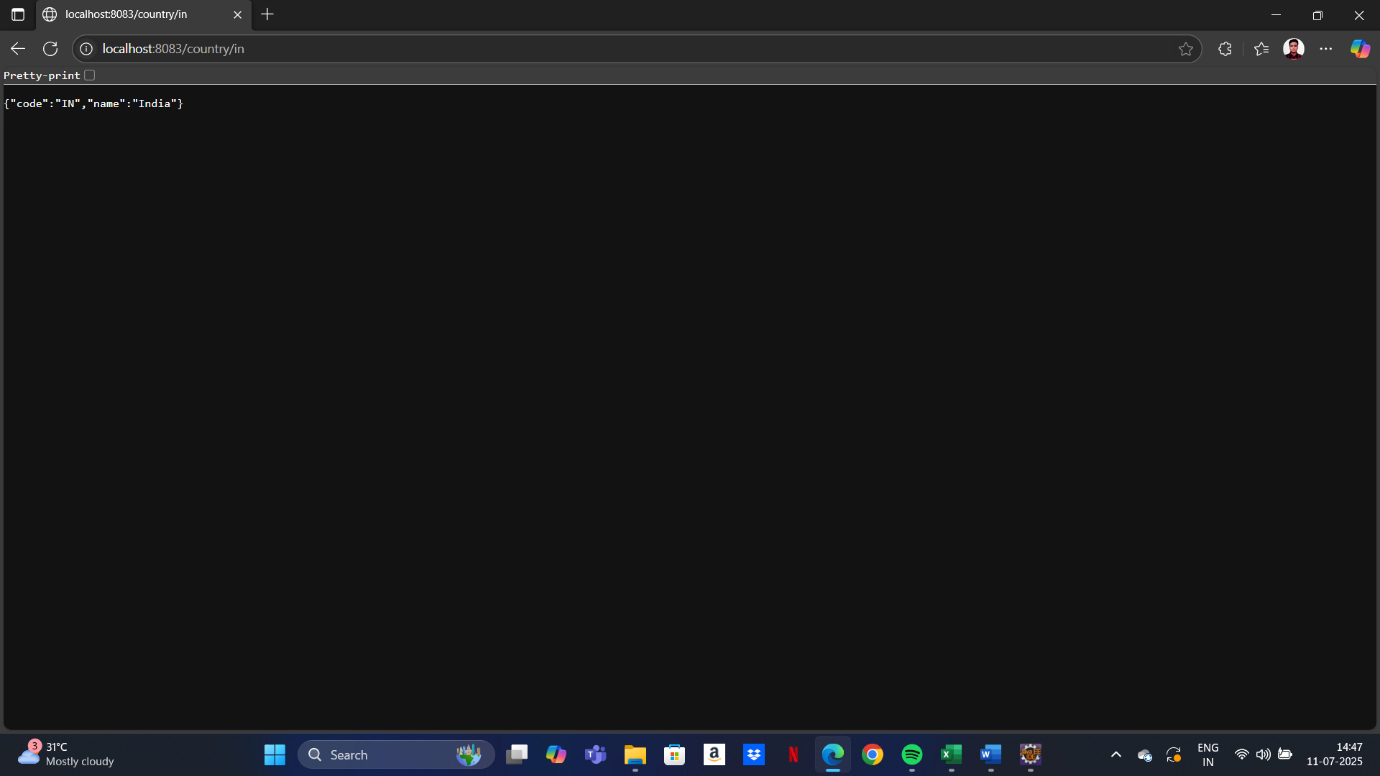
public static void main(String[] args) {

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

}

}

**Output:**



**Additional Important Hands On**

**Spring REST using Spring Boot 3**

**Problem Statement - Display Employee List and Edit Employee form using RESTful Web Service   
  
In the previous angular module, we developed a screen that lists employees and it was populated with hard coded values. Now this angular application has be changed to get the data from RESTful Web Service developed in Spring. The following are the high level activities that needs to be done to accomplish this:**

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

* **Create a REST Service that reads data from xml configuration and returns it**

* **Make changes in angular component to consume the created REST Service**

**Once above activities are completed, clicking on the Edit button against each employee should display Edit Employee form with values retrieved from RESTful Web Service. This will also involve activities similar to the one specified above.  
  
NOTE: There is no specific activity as part of this hands on, refer the next hands ons that covers above three activities in detail.**

**Create static employee list data using spring xml configuration   
  
Follow steps below to accomplish this activity:**

* **Incorporate the following in employee.xml:**
  + **Create one or two more departments**
  + **Create four more instances of Employee.  (use employee sample data from angular)**
  + **Reuse existing skills instead of creating new ones**
  + **Include all four employee instances in an ArrayList.**

* **In EmployeeDao, incorporate the following:**
  + **Create static variable with name EMPLOYEE\_LIST of type ArrayList<Employee>**
  + **Include constructor that reads employee list from xml config and set the EMPLOYEE\_LIST**
  + **Create method getAllEmployees() that returns the EMPLOYEE\_LIST**

**Create REST service to gets all employees   
  
Follow steps below to accomplish this activity:**

* **In EmployeeService, incorporate the following:**
  + **Change the annotation for this class from @Component to @Service**
  + **Create method getAllEmployees() that invokes employeeDao.getAllEmployees() and return the employee list**
  + **Define @Transactional annotation for this method.**

* **In EmployeeController, incorporate the following:**
  + **Include a new get method with name getAllEmployees() that returns the employee list**
  + **Mark this method as GetMapping annotation with the URL as '/employees'**
  + **Within this method invoke employeeService.getAllEmployees() and return the same.**

**​​​​​​**

* **Test ​the service using postman.**

**Create REST service for department**

**Create a new service to get all the departments.**

**Follow steps below to achieve this:**

* **Create a new REST Service, define below list of classes and respective methods:**
  + **DepartmentController**
    - **getAllDepartments() with URL "/departments", this method will return array of departments**
  + **DepartmentService**
    - **getAllDepartments()**
  + **DepartmentDao**
    - **getAllDepartments() - Create a static variable DEPARTMENT\_LIST, this should be populated from spring xml configuration**
* **Test ​the service using postman.**
* **Also verify if department REST service is called by looking into the logs.**

**Code:**

Employee.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

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

<!-- Skills -->

<bean id="skill1" class="com.cognizant.spring\_learn.model.Skill">

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

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

</bean>

<bean id="skill2" class="com.cognizant.spring\_learn.model.Skill">

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

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

</bean>

<!-- Departments -->

<bean id="dept1" class="com.cognizant.spring\_learn.model.Department">

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

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

</bean>

<bean id="dept2" class="com.cognizant.spring\_learn.model.Department">

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

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

</bean>

<!-- Employees -->

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

<constructor-arg>

<list>

<bean class="com.cognizant.spring\_learn.model.Employee">

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

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

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

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

<property name="department" ref="dept2" />

<property name="skills">

<list>

<ref bean="skill1" />

<ref bean="skill2" />

</list>

</property>

</bean>

<!-- Add 3 more employees similarly -->

</list>

</constructor-arg>

</bean>

<!-- Department List -->

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

<constructor-arg>

<list>

<ref bean="dept1" />

<ref bean="dept2" />

</list>

</constructor-arg>

</bean>

</beans>

EmployeeDao.java :-

@Repository

public class EmployeeDao {

private static List<Employee> EMPLOYEE\_LIST;

public EmployeeDao() {

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

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

}

public List<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

}

DepartmentDao.java :-

@Repository

public class DepartmentDao {

private static List<Department> DEPARTMENT\_LIST;

public DepartmentDao() {

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

DEPARTMENT\_LIST = (List<Department>) context.getBean("departmentList");

}

public List<Department> getAllDepartments() {

return DEPARTMENT\_LIST;

}

}

EmployeeService.java :-

@Service

public class EmployeeService {

@Autowired

private EmployeeDao employeeDao;

@Transactional

public List<Employee> getAllEmployees() {

return employeeDao.getAllEmployees();

}

}

DepartmentService.java :-

@Service

public class DepartmentService {

@Autowired

private DepartmentDao departmentDao;

@Transactional

public List<Department> getAllDepartments() {

return departmentDao.getAllDepartments();

}

}

EmployeeController.java :-

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@GetMapping

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

}

DepartmentController.java :-

@RestController

@RequestMapping("/departments")

public class DepartmentController {

@Autowired

private DepartmentService departmentService;

@GetMapping

public List<Department> getAllDepartments() {

return departmentService.getAllDepartments();

}

}

SpringLearnApplication.java(main) :-

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringLearnApplication {

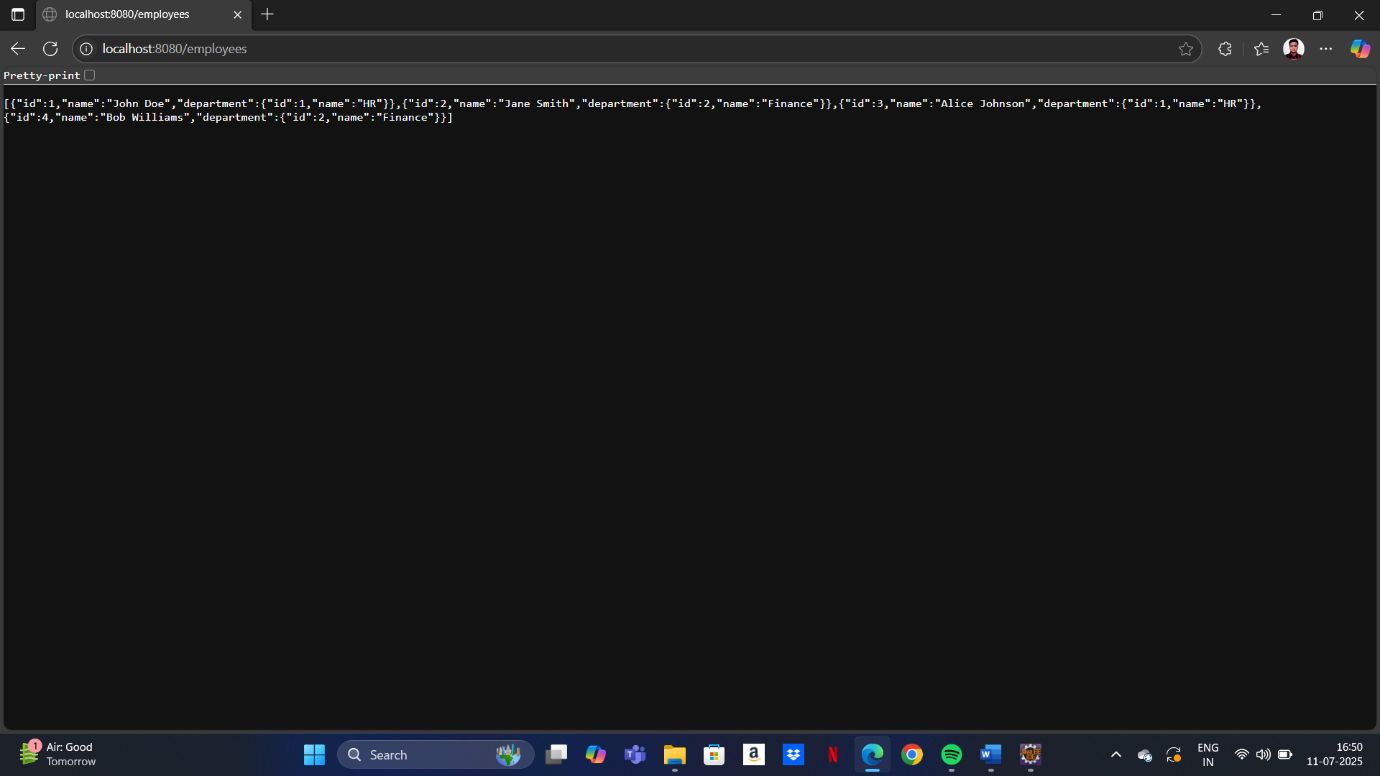
public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Output:**

****