**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

1. src/main/resources - Folder for application configuration
2. src/test/java - Folder with code for testing the application

4.SpringLearnApplication.java - Walkthrough the main() method.

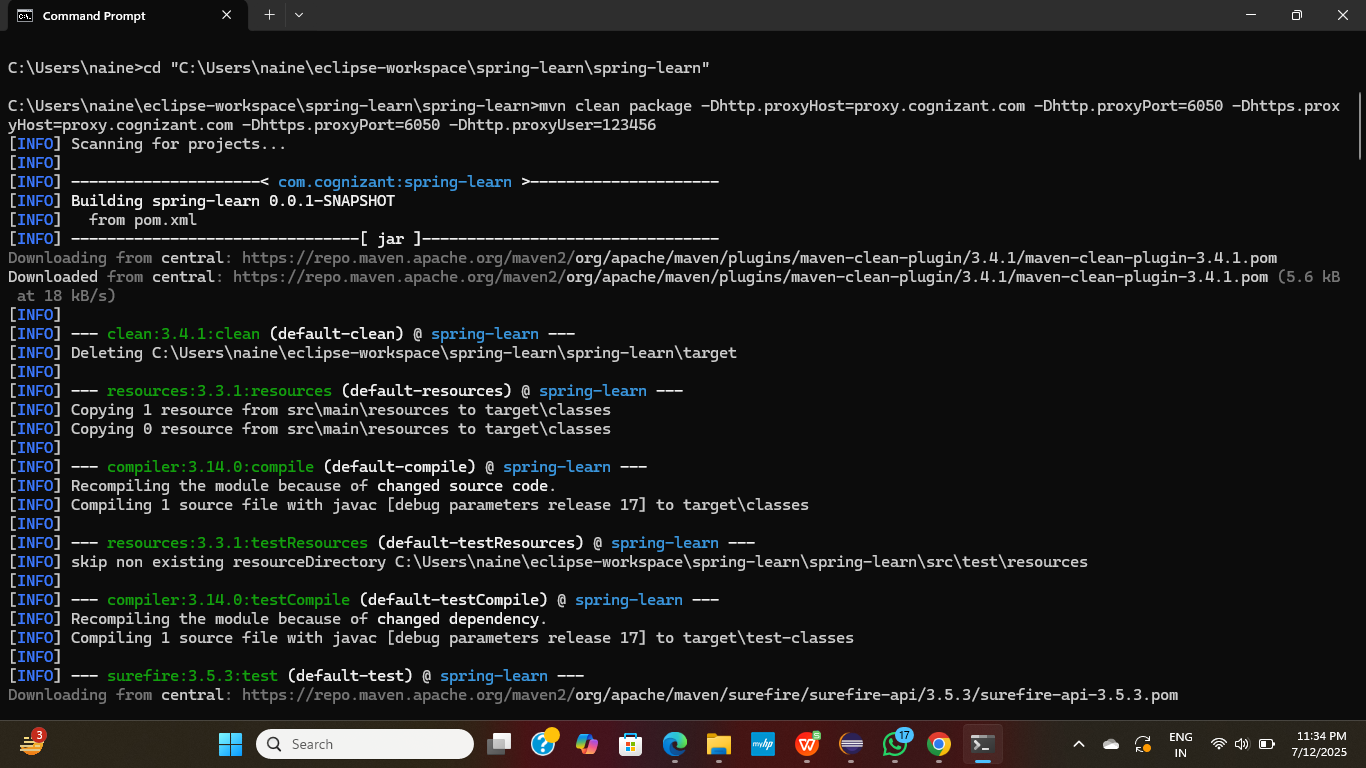
5.Purpose of @SpringBootApplication annotation

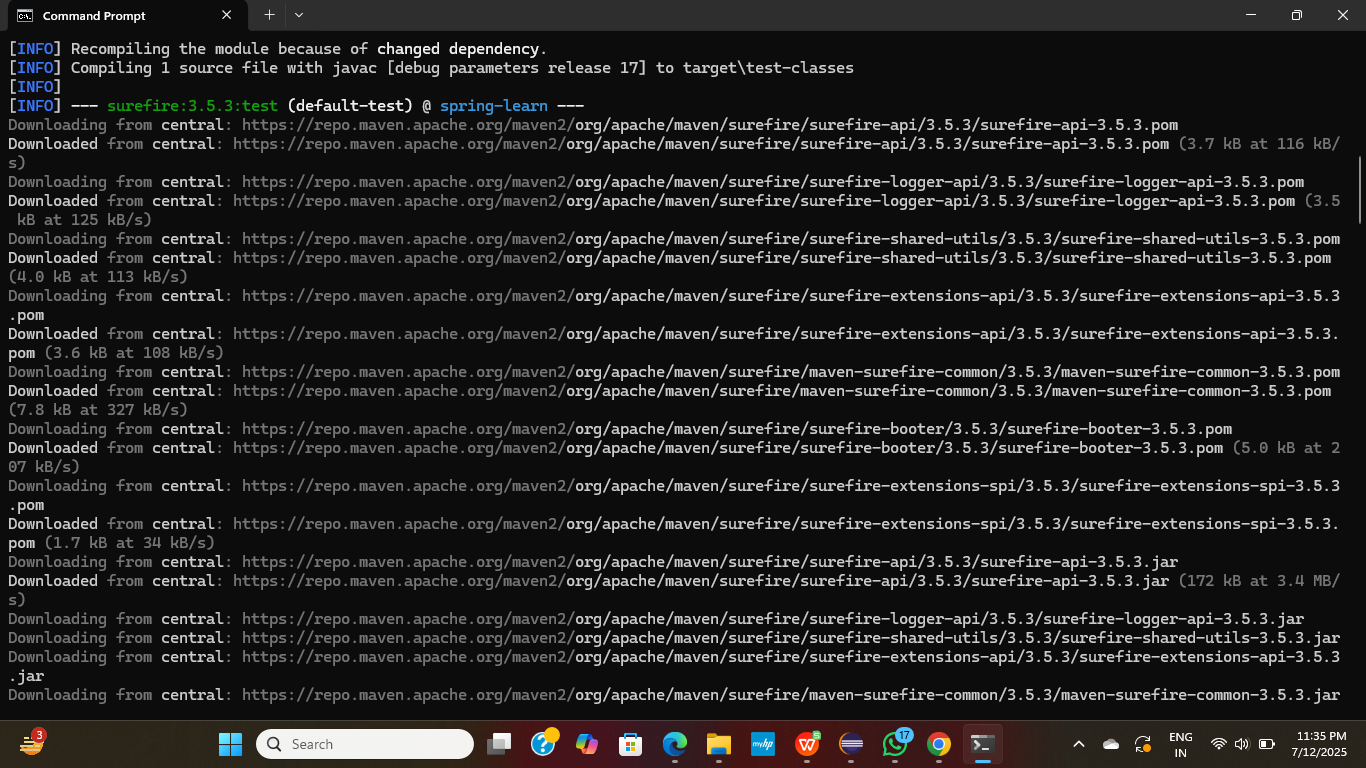
6. pom.xml

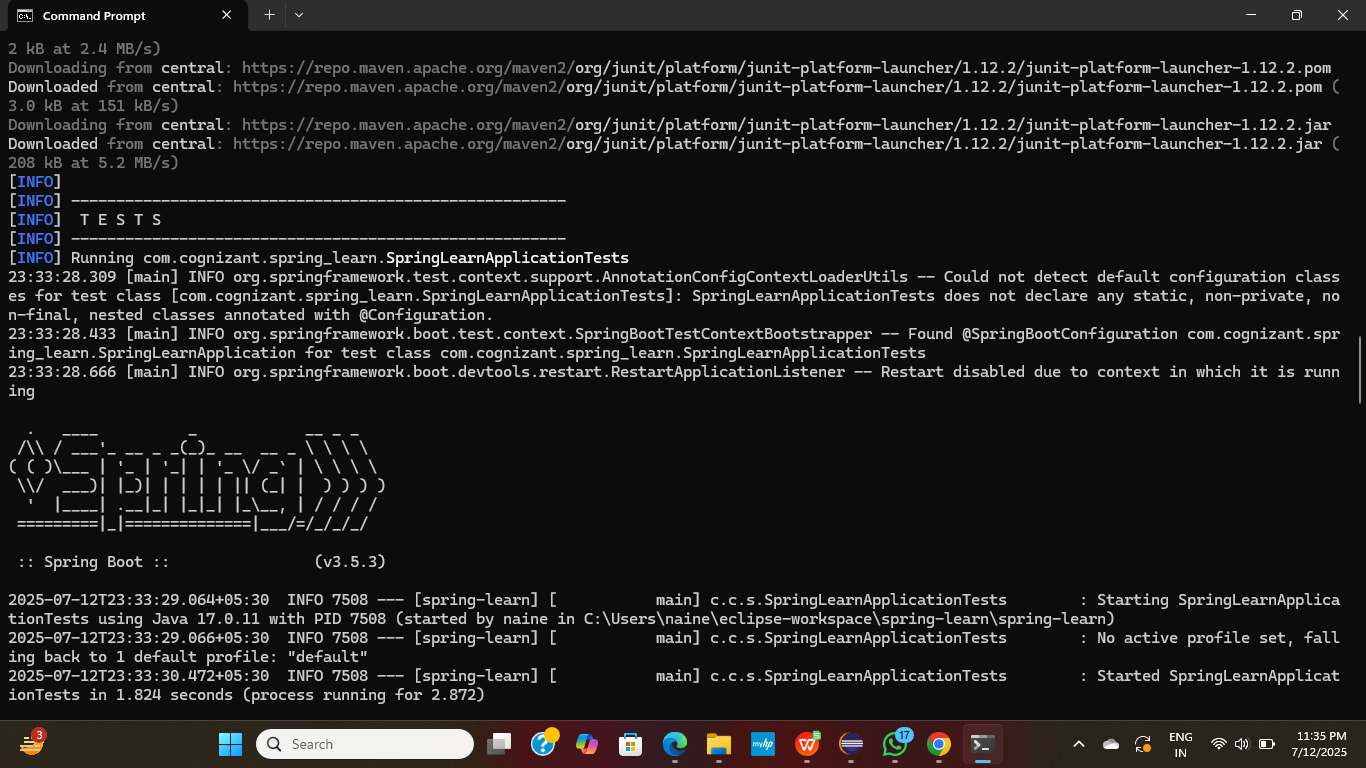
1. Walkthrough all the configuration defined in XML file

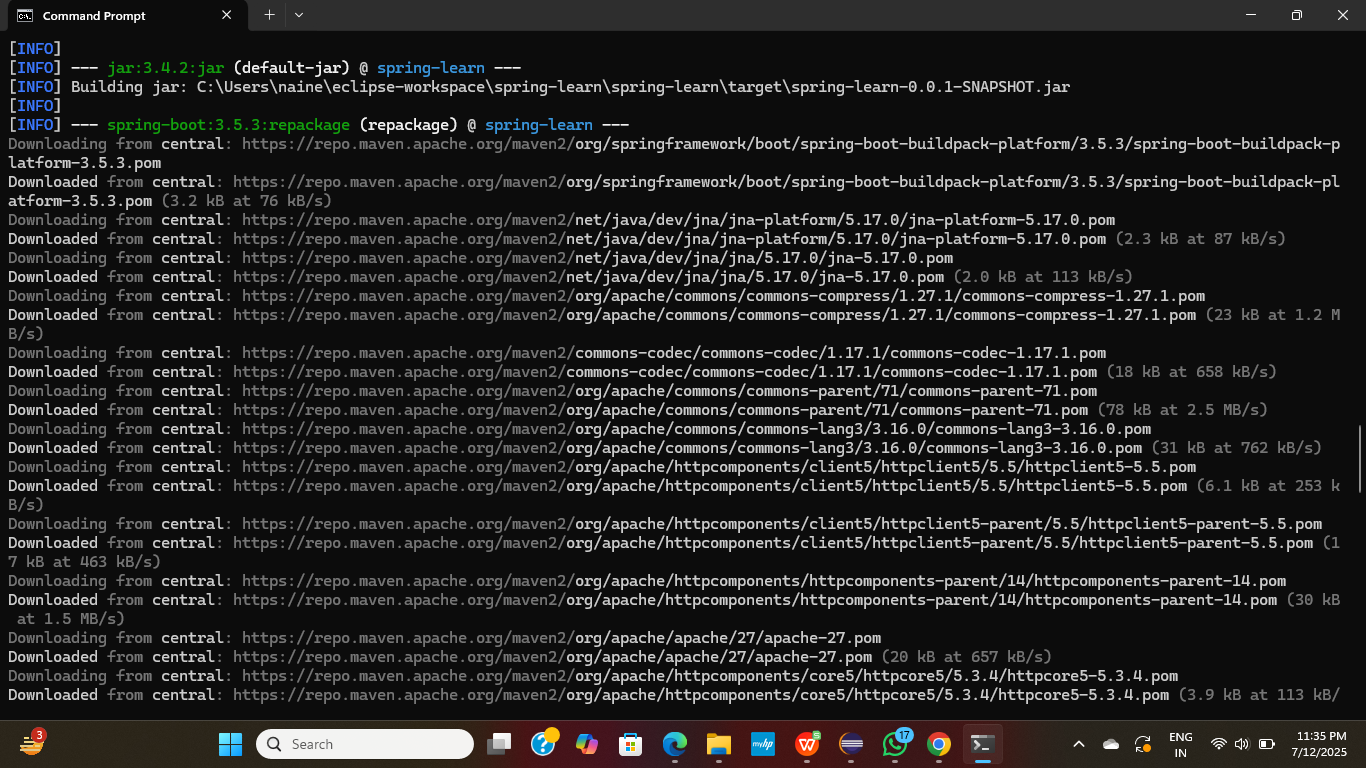
2. Open 'Dependency Hierarchy' and show the dependency tree.

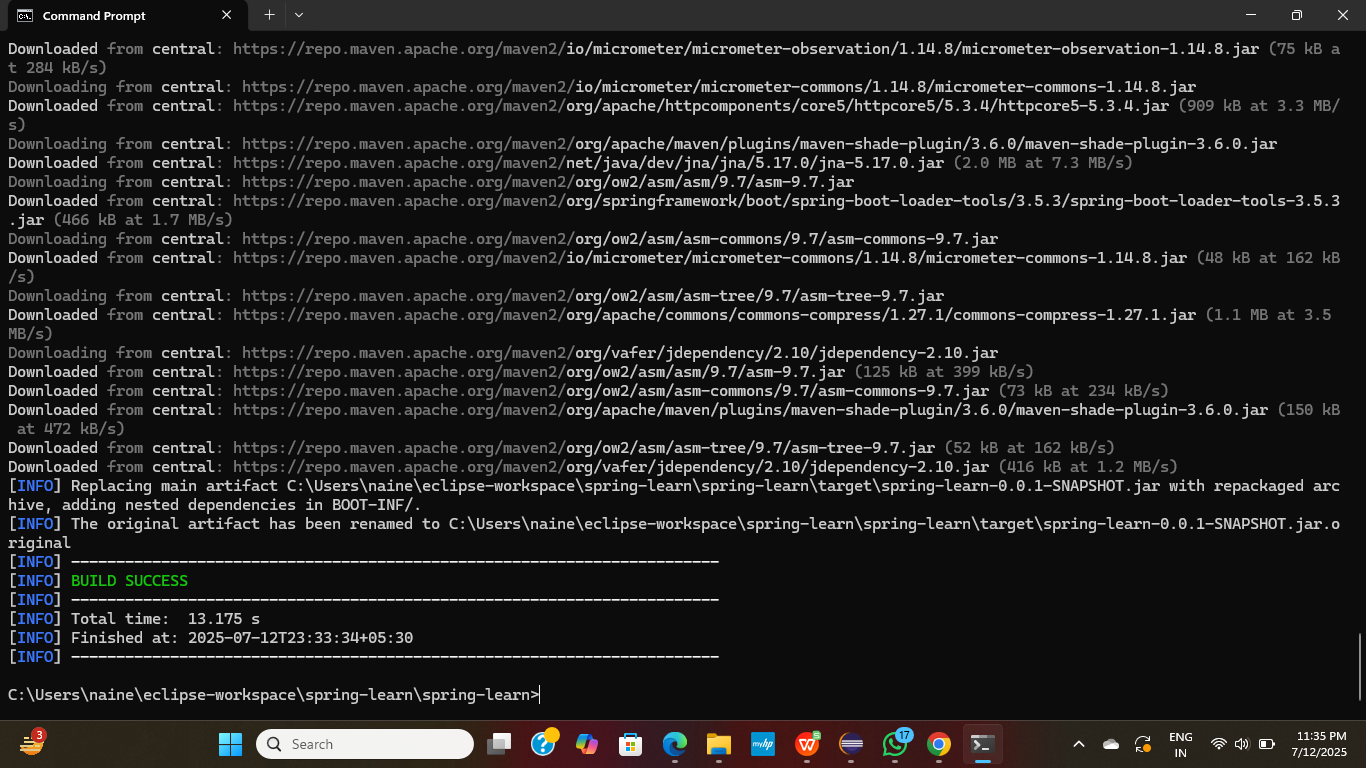
**OUTPUT:**

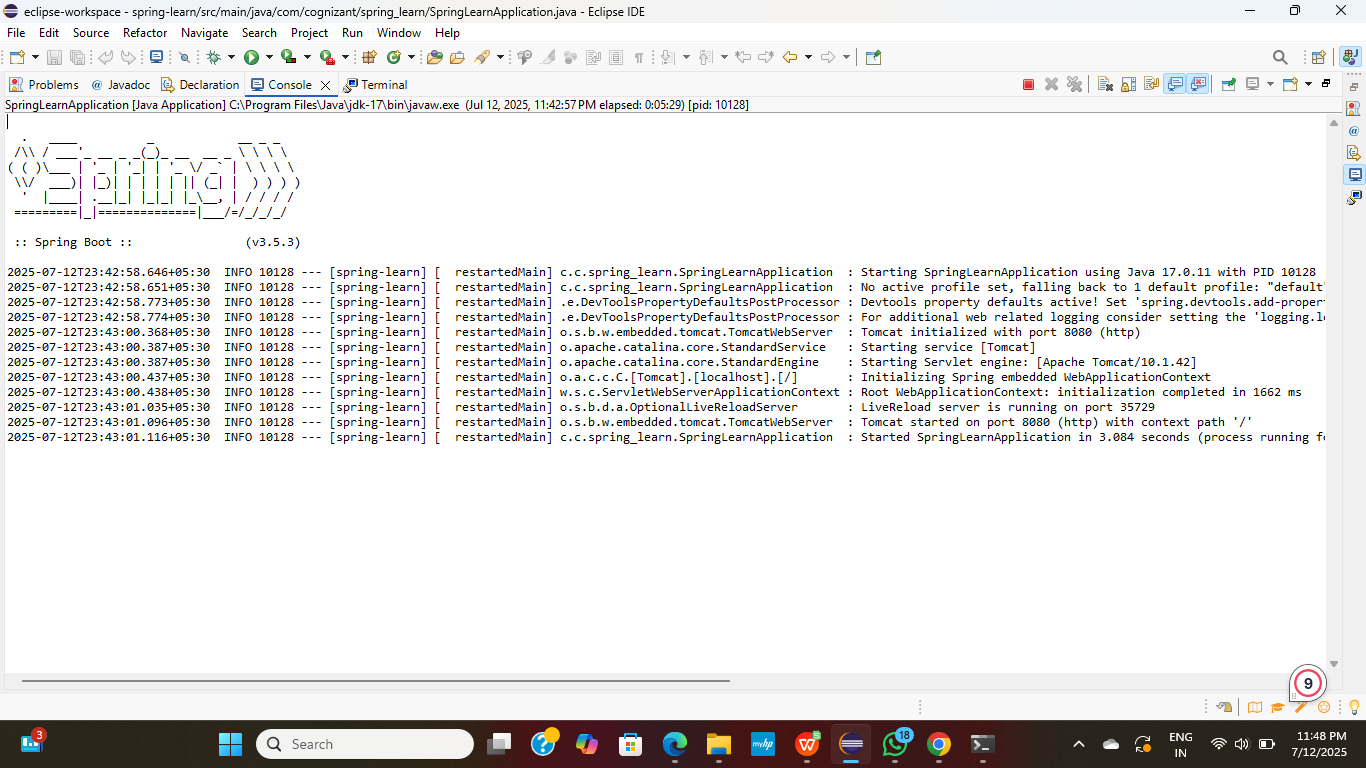
****

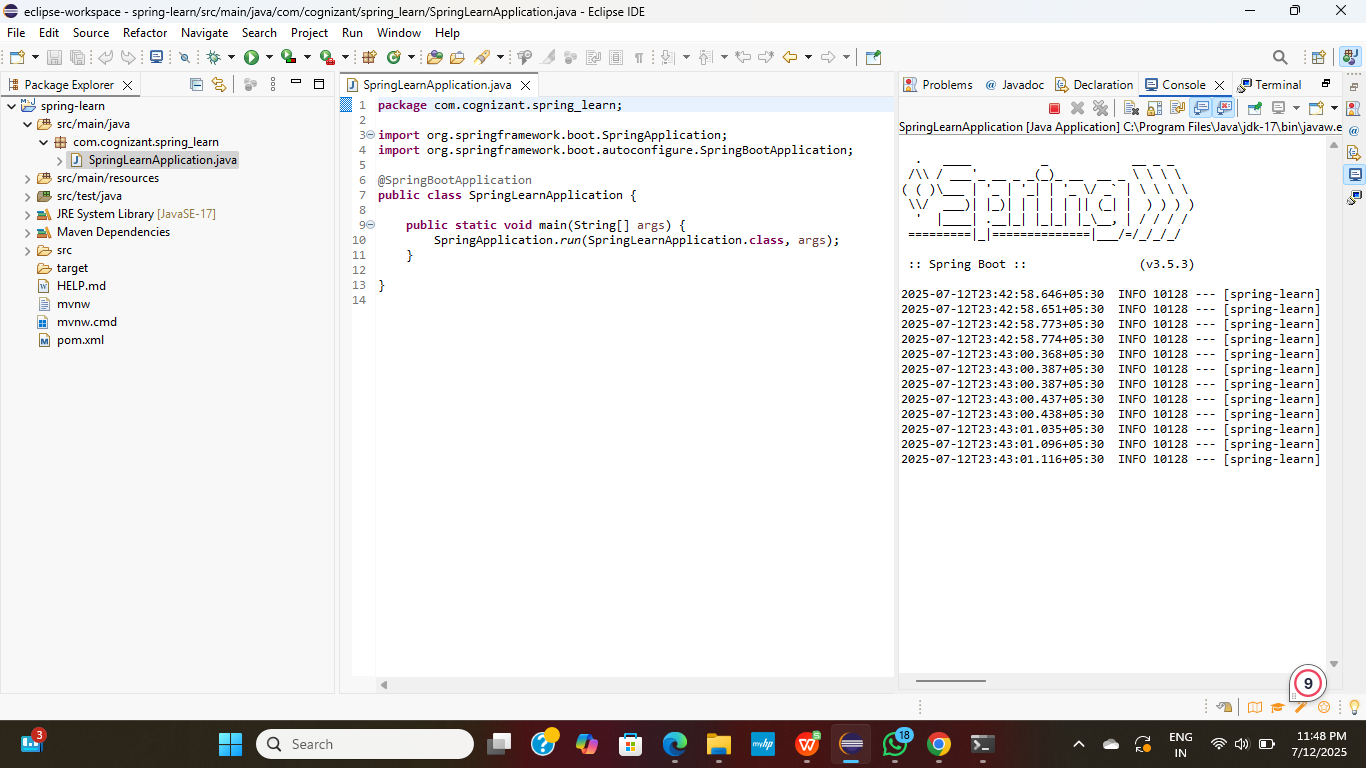
****

****

****

****

****

****

**Hands on 4**

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

### 1. pom.xml (Maven Project Object Model)

This file defines your project's dependencies and build configuration.

**Code:**

<?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 https://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>2.7.0</version> <!-- Use a recent stable version -->

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant.springlearn</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-learn</name>

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

<properties>

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

</properties>

<dependencies>

<!-- Spring Context for XML configuration -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.20</version> <!-- Ensure this version is compatible with your Spring Boot parent version -->

</dependency>

<!-- Spring Boot DevTools for automatic restarts (optional but helpful) -->

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<!-- Logging (e.g., Logback or Log4j2) -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.32</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.11</version>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Test (if you plan to write tests) -->

<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>

1. **Country.java**

**Code:**

package com.cognizant.springlearn;

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;

// Empty constructor

public Country() {

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

}

// Getters and Setters

public String getCode() {

LOGGER.debug("Inside getCode() method.");

return code;

}

public void setCode(String code) {

LOGGER.debug("Inside setCode() method.");

this.code = code;

}

public String getName() {

LOGGER.debug("Inside getName() method.");

return name;

}

public void setName(String name) {

LOGGER.debug("Inside setName() method.");

this.name = name;

}

// toString() method for easy printing

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

1. **country.xml**

**Code:**

<?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">

<!-- Define a bean for the Country class -->

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

<!-- Set properties for the Country object -->

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

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

</bean>

</beans>

1. **SpringLearnApplication.java**

**Code:**

package com.cognizant.springlearn;

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) {

// Log entry into the main method

LOGGER.debug("Inside main method.");

// Call the method to display country details

displayCountry();

// Log exit from the main method

LOGGER.debug("Exiting main method.");

}

private static void displayCountry() {

// Log entry into displayCountry method

LOGGER.debug("Inside displayCountry method.");

// Create a Spring ApplicationContext from the XML configuration file

// This loads the beans defined in country.xml

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

// Retrieve the "country" bean from the context

// The beanId "country" matches the id attribute in the <bean> tag in country.xml

// The second argument (Country.class) specifies the expected type of the bean

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

// Log the retrieved Country object

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

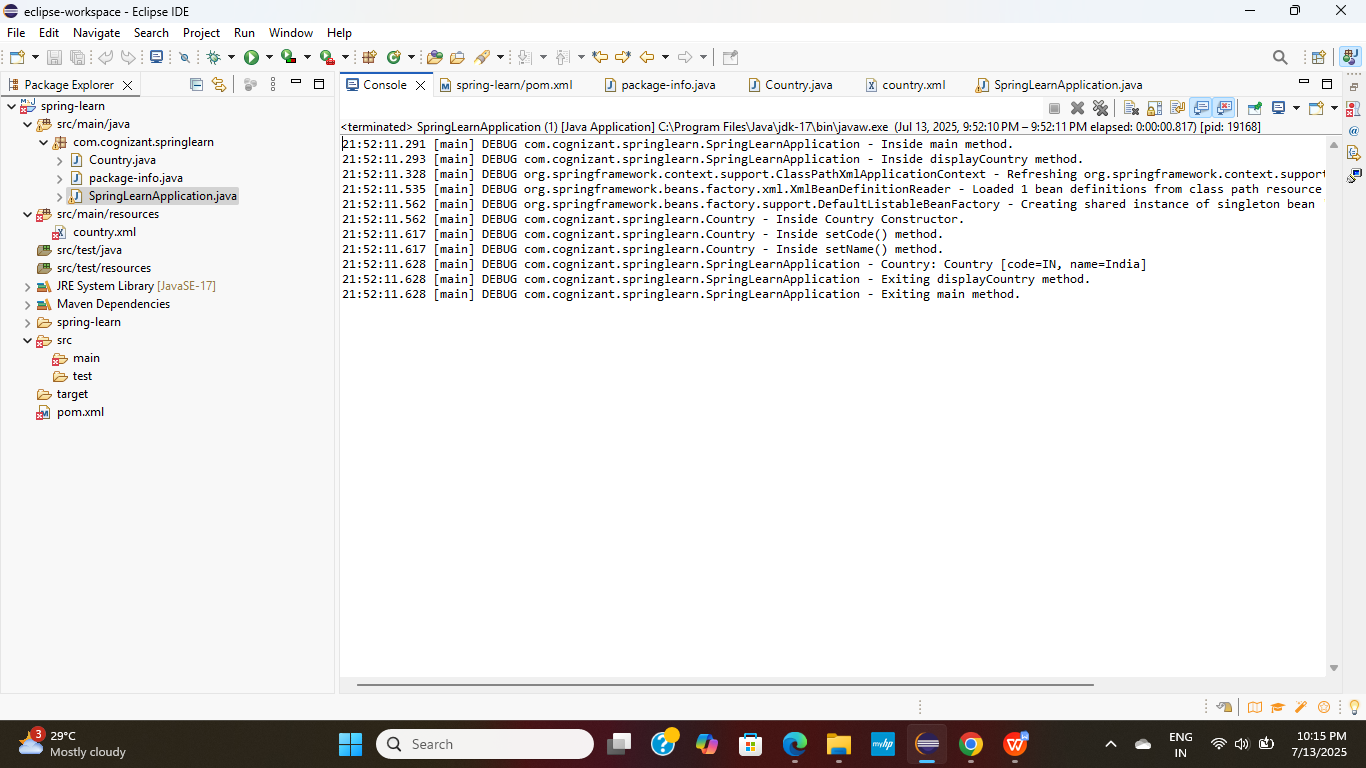
// Log exit from displayCountry method

LOGGER.debug("Exiting displayCountry method.");

}

}

**OUTPUT:**

****