**Hands on 1**

**Create a Spring Web Project using Maven**

**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 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>3.5.3</version>

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

</parent>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-learn</name>

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

<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>

**beans.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**>

**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 + "]";

}

}

**SpringLearnApplication.java**

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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 {

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

public static void main(String[] args) {

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

***LOGGER***.info("START");

// Load Spring XML configuration

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

// Get the bean

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

// Log and print the bean

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

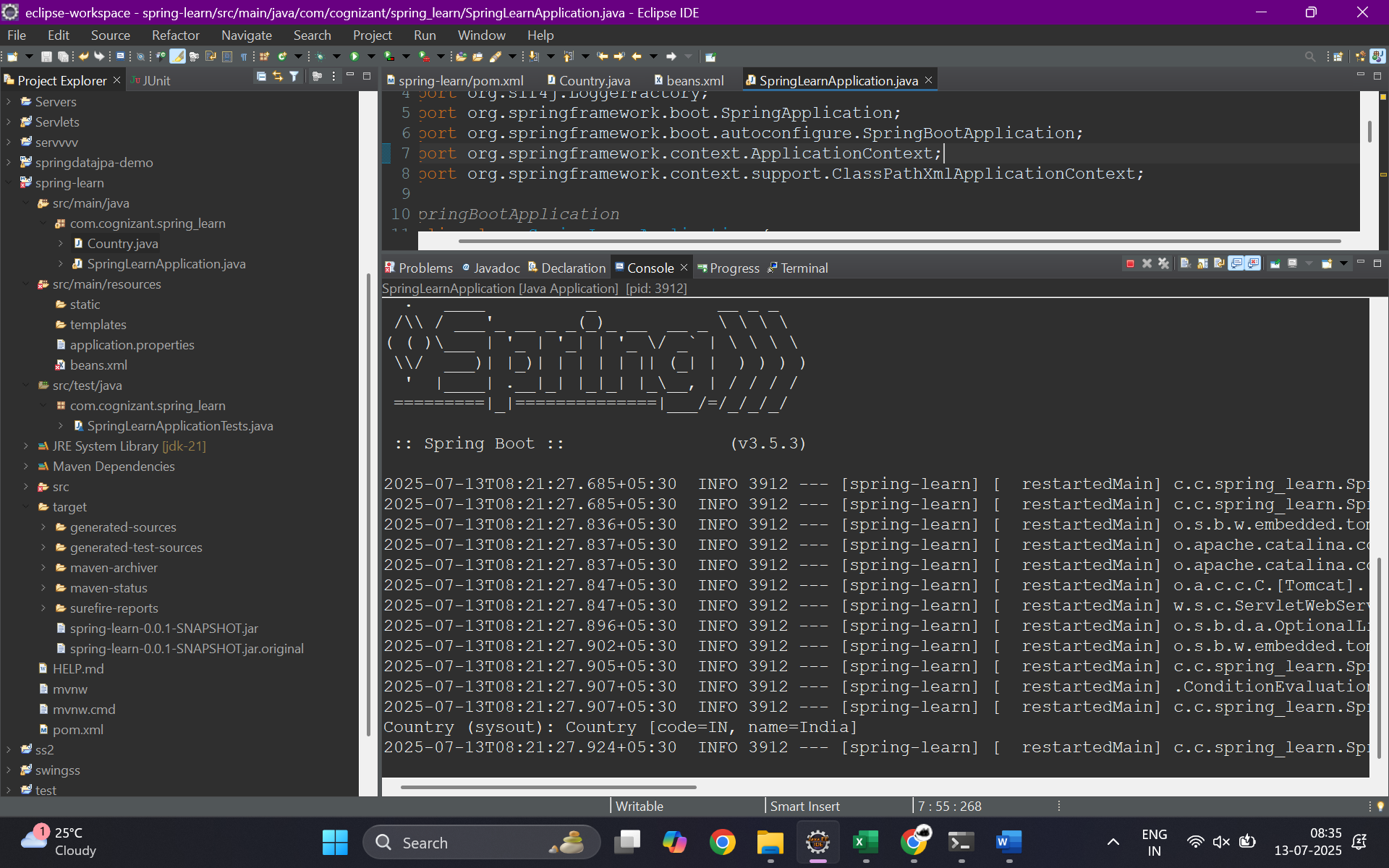
System.***out***.println("Country (sysout): " + country);

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

}

}

**Output:**



**SME**

In this hands-on exercise, we created a basic Spring Boot web application using Maven. The project follows the standard Maven directory structure. The src/main/java folder contains the main application source code including the entry point SpringLearnApplication.java. This is where the application's execution begins. The src/main/resources directory is used for storing configuration files such as application.properties or Spring XML configuration files (like beans.xml or country.xml used in later hands-ons). The src/test/java folder is reserved for unit and integration tests, allowing developers to write test cases separately from the main code.

The SpringLearnApplication.java file contains the main() method that boots up the application using SpringApplication.run(). This method initializes the Spring container, performs classpath scanning, and sets up all configurations and beans. The class is annotated with @SpringBootApplication, which is a meta-annotation that combines @Configuration, @EnableAutoConfiguration, and @ComponentScan. It tells Spring Boot to treat this as the main configuration class and to enable component scanning for the package.

The pom.xml file is the Maven configuration file. It manages dependencies, plugins, and build configurations for the project. Dependencies like Spring Boot Starter Web and Spring Boot DevTools are declared here, and Maven handles downloading and managing their transitive dependencies. The Dependency Hierarchy view in Eclipse shows how these libraries are interconnected, helping developers identify any version conflicts or redundant dependencies.

Additionally, if XML configuration is used (such as beans.xml), the SME should walk through the structure of these files—explaining how beans are declared, how their properties are injected, and how they are wired using Spring's dependency injection model. The Dependency Hierarchy tab in Eclipse is a useful tool to visually inspect the entire tree of libraries included in the project.