**WEEK-4**

**SPRING REST USING SPRING BOOT**

* **Create a Spring Web Project using Maven**

1. SpringLearnApplication.java

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);

}

}

1. 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-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

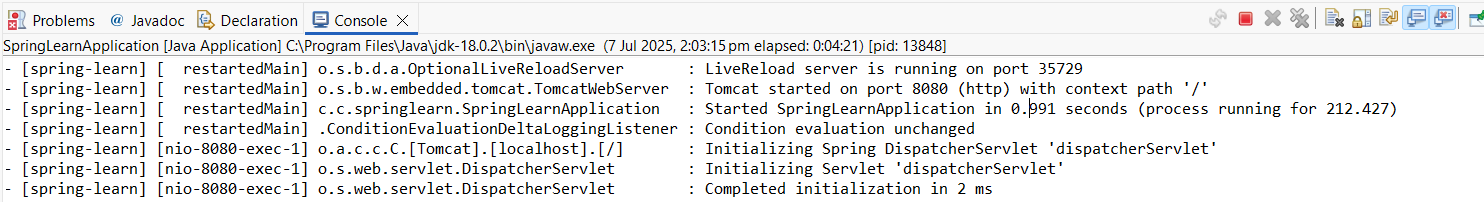
<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

**Output:**



**Explanation:**

* In order to construct a simple online application, I used Spring Initializr to create a Spring Boot project called spring-learn, choosing Java 17, Maven, and Spring Boot 3.5.3 with dependencies like Spring online and DevTools.
* I downloaded and extracted the project, then imported it as a Maven project into Eclipse, making sure the folder structure (src/main/java, src/main/resources, pom.xml, etc.) was correct.
* I built the project at the console by executing the command mvn clean package, which successfully compiled and packaged the code.
* To designate it as the entry point for Spring Boot, I made the main class SpringLearnApplication.java inside the com.cognizant.springlearn package and annotated it with @SpringBootApplication.
* When I ran the project, it started successfully, confirming that Spring Boot and Maven were correctly configured and the project was ready for developing REST APIs.
* **Spring Core – Load Country from Spring Configuration XML**

1. Src/main/resources -> New -> File

* 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.springlearn.Country">

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

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

</bean>

</beans>

1. Src/main/java -> com.xognizant.springlearn -> Country (class)

* Country.java

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;

public Country() {

LOGGER.debug("Inside Country Constructor");

}

public String getCode() {

LOGGER.debug("Getting code: {}", code);

return code;

}

public void setCode(String code) {

LOGGER.debug("Setting code: {}", code);

this.code = code;

}

public String getName() {

LOGGER.debug("Getting name: {}", name);

return name;

}

public void setName(String name) {

LOGGER.debug("Setting name: {}", name);

this.name = name;

}

@Override

public String toString() {

return "Country{" + "code='" + code + '\'' + ", name='" + name + '\'' + '}';

} }

1. Create new SpringLearnApplication3.java (we won’t get displayDate() method )

package com.cognizant.springlearn;

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 SpringLearnApplication3 {

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

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

displayCountry();

}

public static void displayCountry() {

LOGGER.info("START");

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

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

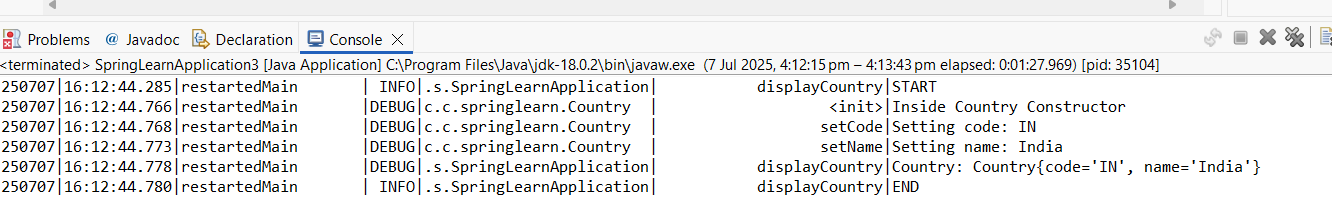
LOGGER.debug("Country: {}", country.toString());

LOGGER.info("END");

}

}

**Output:**



**Explanation:**

* I defined a country bean with code and name properties in an XML configuration file called country.xml that I made for this assignment and placed inside the src/main/resources folder.
* For improved traceability, I used SLF4J to add the proper getter, setter, and logging instructions to a basic Java class called Country.java that had fields for code and name.
* In order to test XML-based bean loading using ClassPathXmlApplicationContext—a fundamental idea in conventional Spring applications—I then developed a new main class, SpringLearnApplication3.java.
* To confirm that dependency injection via XML configuration functioned as intended, I pulled the nation bean from the XML and printed its data inside the displayCountry() method.
* When I ran the application, the logs confirmed that the bean was initialized and its properties were set and accessed successfully, demonstrating how Spring Core loads and manages beans from XML files.