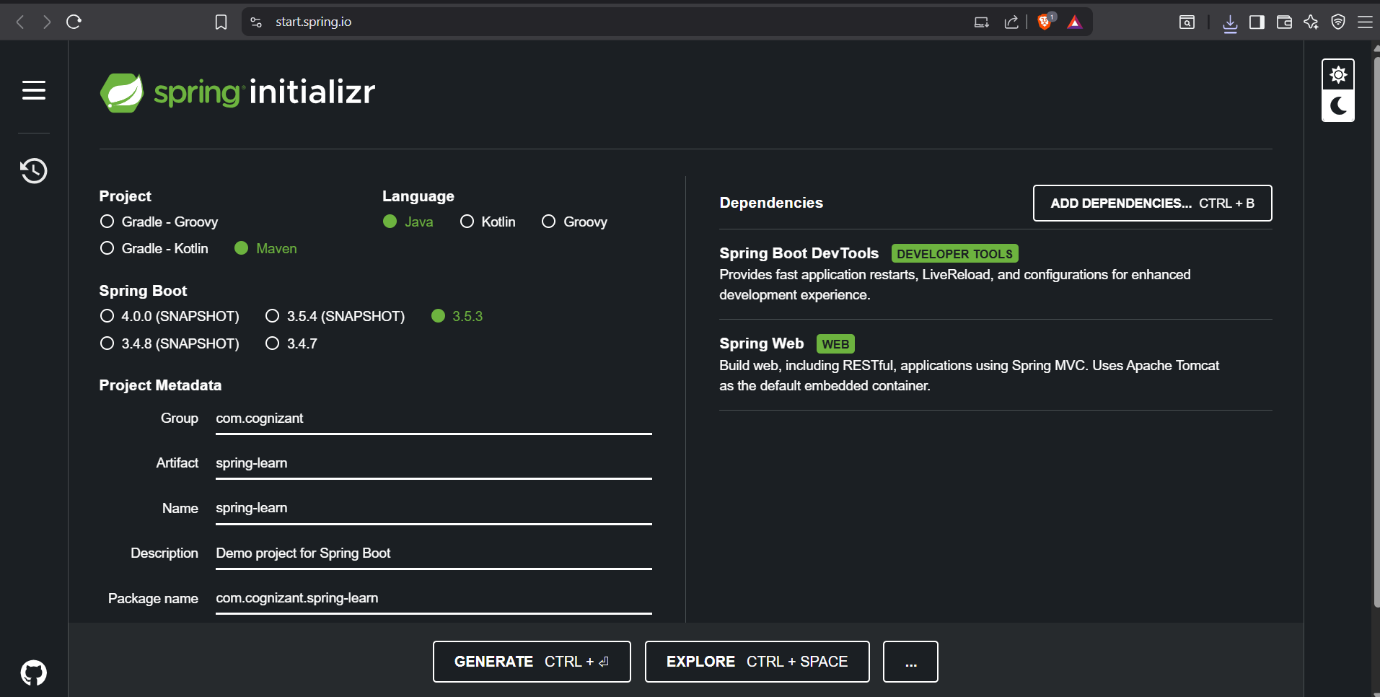
**Hands on 1: Create a Spring Web Project using Maven**

1. Go to Spring Initializr:



2. Extract and Import in IntelliJ:

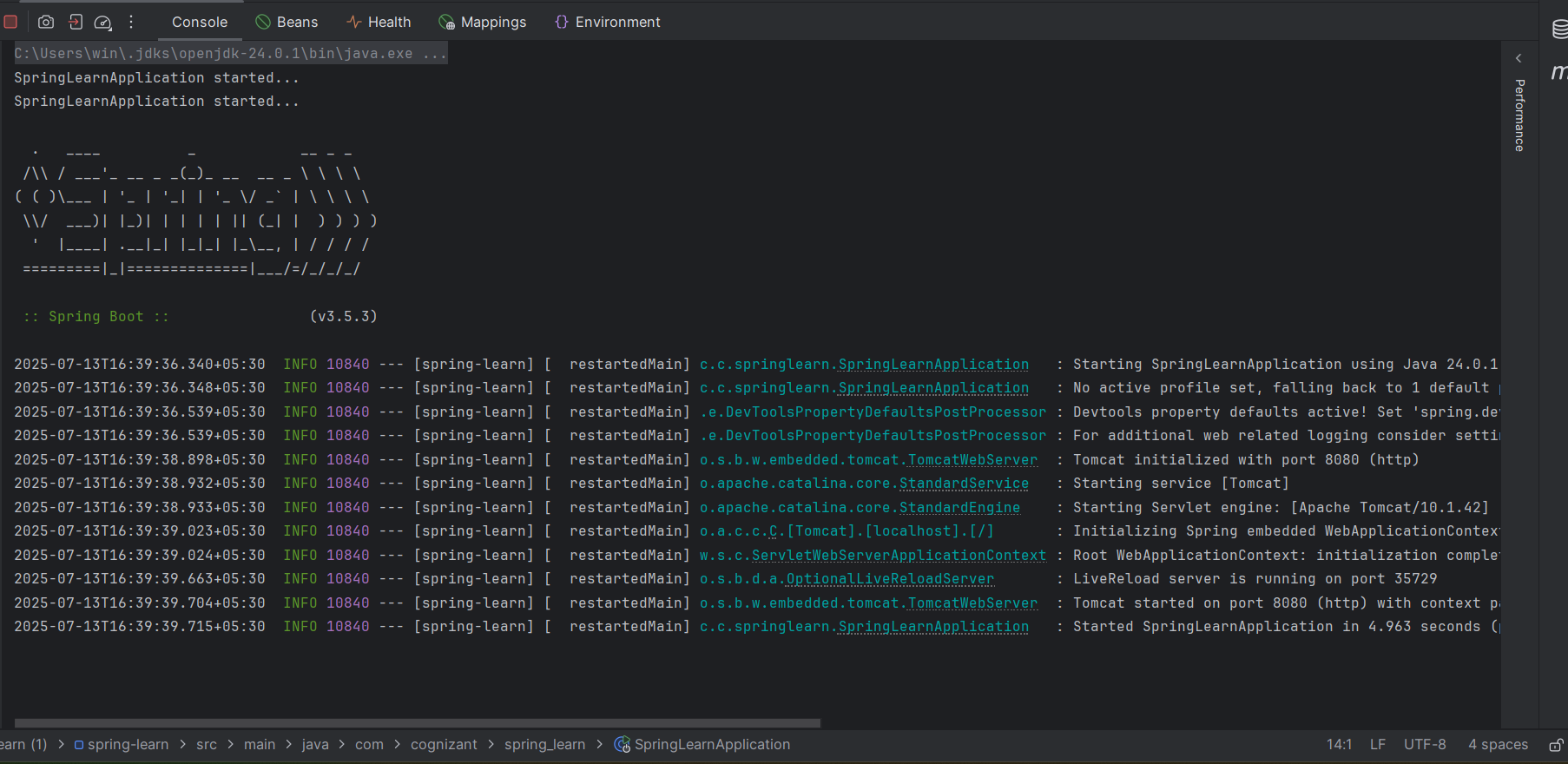
1. Extract the downloaded ZIP.
2. Open IntelliJ IDEA.
3. Select File > New > Project from Existing Sources.
4. Browse and select the extracted spring-learn folder.
5. Choose "Import project from external model" → Maven, then click Next until you reach Finish.

3. Verify Logs and Run Application:

**File:** 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) {  
 System.*out*.println("SpringLearnApplication started...");  
 SpringApplication.*run*(SpringLearnApplication.class, args);  
 }  
}

Run the application:



## 4. **IntelliJ Project Structure Walkthrough**

### a. src/main/java

Contains your main **application code**, including:

* Controllers
* Services
* SpringBoot main class

### b. src/main/resources

Contains **config files** like:

* application.properties
* static content (HTML, CSS, JS)
* templates (Thymeleaf/Freemarker)

### c. src/test/java

Used for **unit and integration testing**. Default test class is auto-generated.

### d. SpringLearnApplication.java

#### The main() method:

Acts as the **entry point** for the application. It bootstraps the Spring Boot application using:

SpringApplication.run(SpringLearnApplication.class, args);

#### @SpringBootApplication Annotation:

This is a **meta-annotation** combining:

* @Configuration - For Java-based configuration
* @EnableAutoConfiguration - Enables Spring Boot's auto-config
* @ComponentScan - Scans the current package and sub-packages

It makes your project **ready to run** with minimal setup.

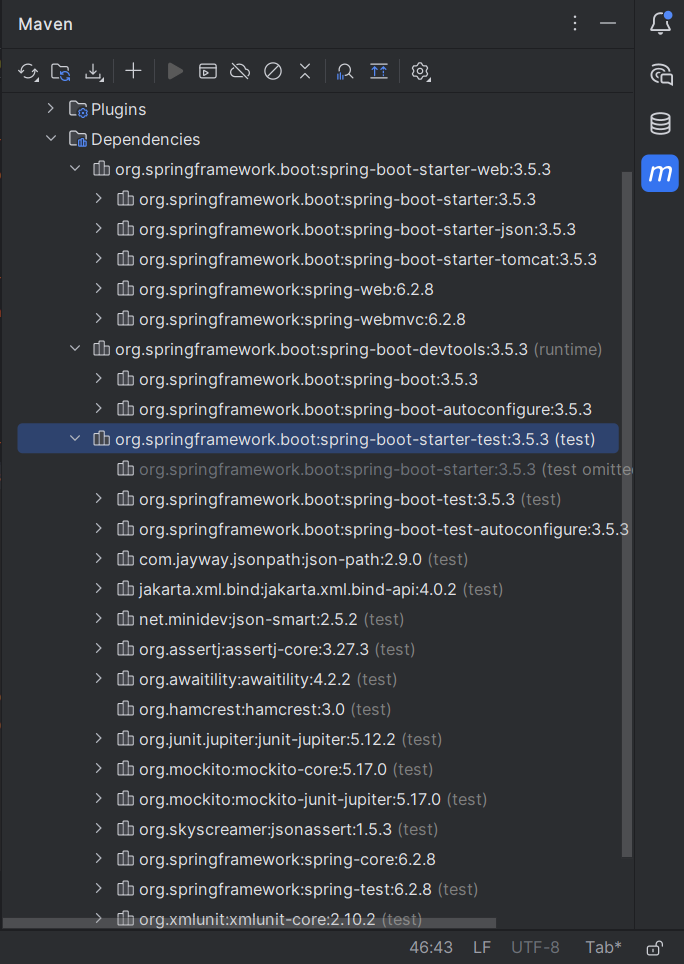
5. pom.xml Walkthrough

This file defines:

* Project coordinates (groupId, artifactId, version)
* Project dependencies
* Build plugins

<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>  
<url/>  
<licenses>  
 <license/>  
</licenses>  
<developers>  
 <developer/>  
</developers>  
<scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
</scm>  
<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>

6. Dependency Hierarchy in IntelliJ:



**Hands on 4 Spring Core – Load Country from Spring Configuration XML**

### **1. Create** Country.java

**Location:** src/main/java/com/cognizant/springlearn/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("Inside getCode()");  
 return code;  
 }  
  
 public void setCode(String code) {  
 *LOGGER*.debug("Inside setCode()");  
 this.code = code;  
 }  
  
 public String getName() {  
 *LOGGER*.debug("Inside getName()");  
 return name;  
 }  
  
 public void setName(String name) {  
 *LOGGER*.debug("Inside setName()");  
 this.name = name;  
 }  
  
 @Override  
 public String toString() {  
 return "Country [code=" + code + ", name=" + name + "]";  
 }  
}

### **2. Create** country.xml

**Location:** src/main/resources/country.xml

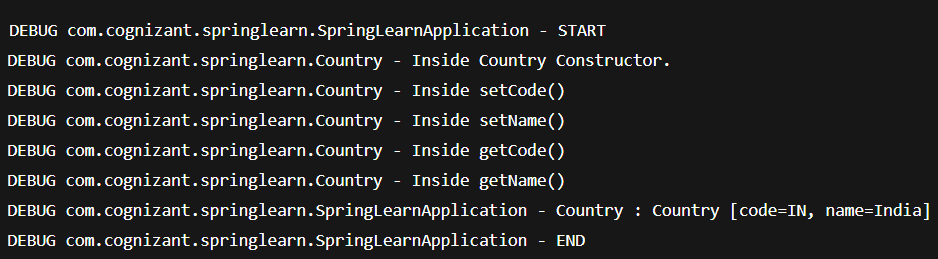
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans   
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <bean id="country" class="com.cognizant.springlearn.Country">  
 <property name="code" value="IN" />  
 <property name="name" value="India" />  
 </bean>  
  
</beans>

### **3. Modify** SpringLearnApplication.java

**Location:** src/main/java/com/cognizant/springlearn/SpringLearnApplication.java

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 displayCountry() {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 Country country = context.getBean("country", Country.class);  
 *LOGGER*.debug("Country : {}", country.toString());  
 }  
  
 public static void main(String[] args) {  
 *LOGGER*.debug("START");  
 *displayCountry*();  
 *LOGGER*.debug("END");  
 }  
}

4. Run the Application:

****

**CONCEPTS EXPLAINED**

**bean tag**

Defines a Spring bean (i.e., object managed by Spring container).

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

* id: unique name to reference the bean
* class: fully qualified name of the class

**property tag**

Used to set properties of the bean via setters.

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

* name="code" → calls setCode()
* value="IN" → passes "IN" as argument

**ApplicationContext**

* Central interface to access Spring beans
* Has methods to get beans and manage lifecycle

🔸 ClassPathXmlApplicationContext

Loads the XML file from src/main/resources (class path) and initializes the Spring container.

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

What Happens on context.getBean()?

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

Step-by-step:

1. Spring container reads country.xml
2. Finds bean with id "country"
3. Loads class com.cognizant.springlearn.Country
4. Calls default constructor
5. Sets code and name using setCode() and setName()
6. Returns the fully initialized bean
7. You call toString() → internally triggers getCode() and getName()