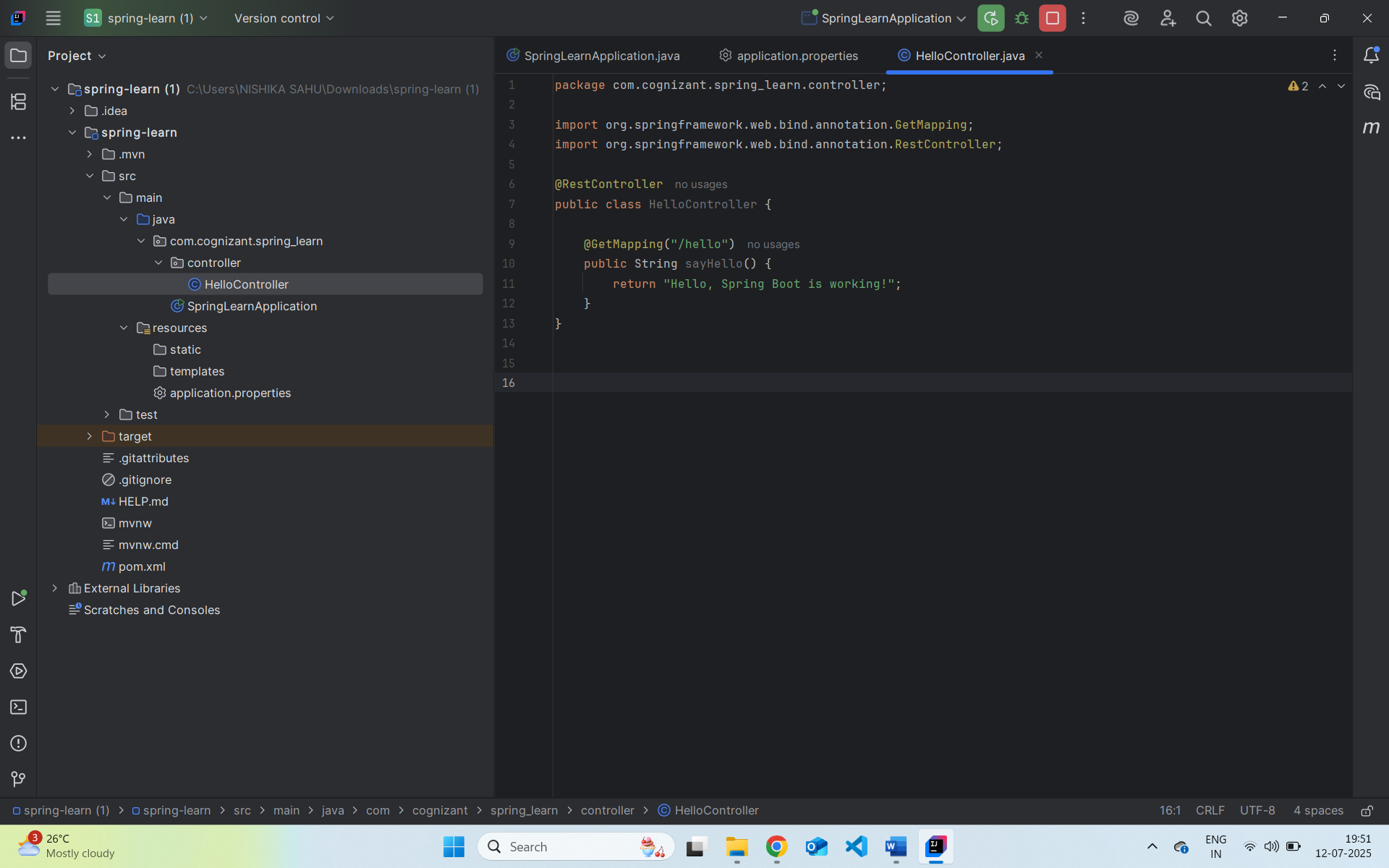
**Week 4**

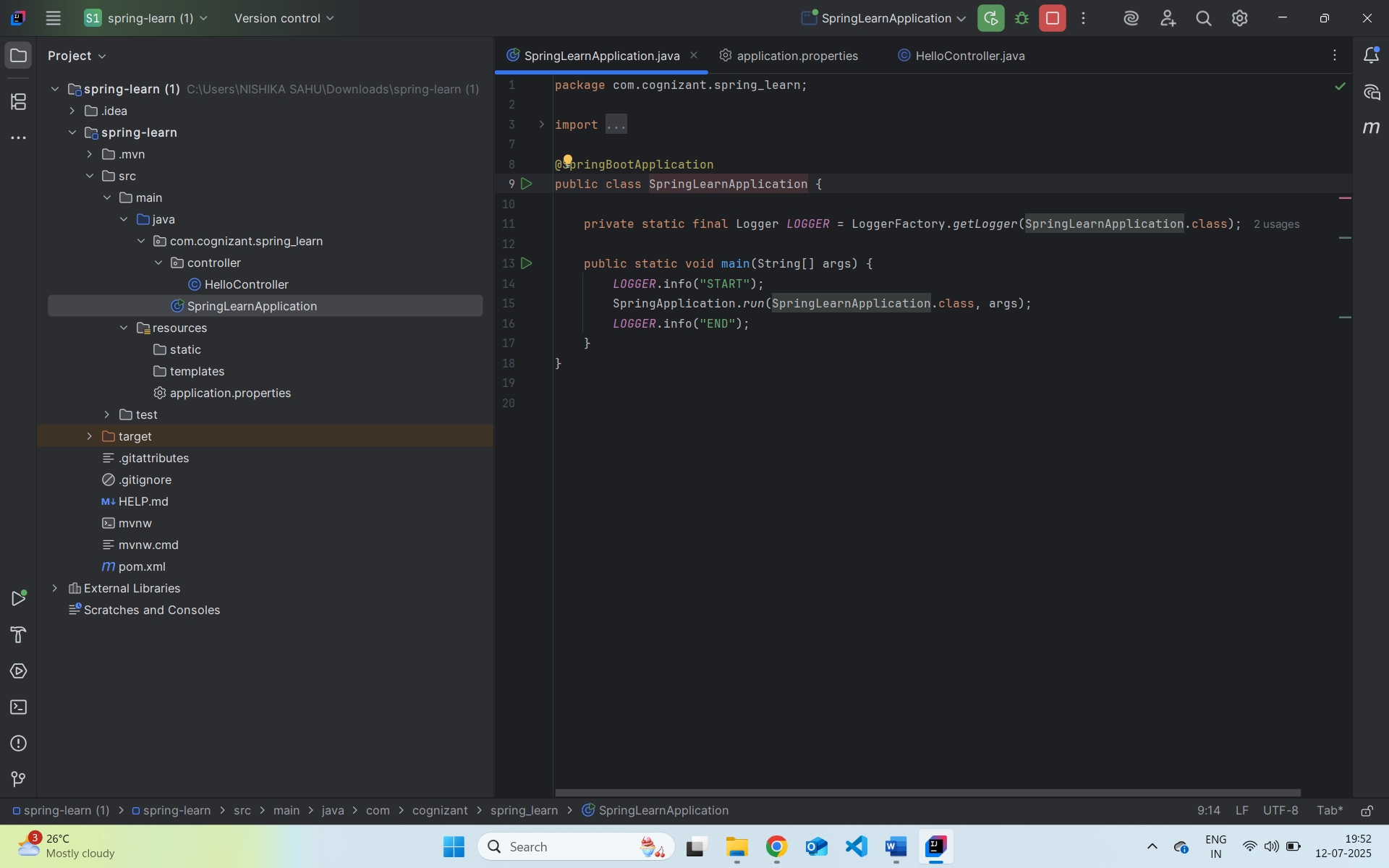
**1. spring-rest-hands-on**

**Create a Spring Web Project using Maven**

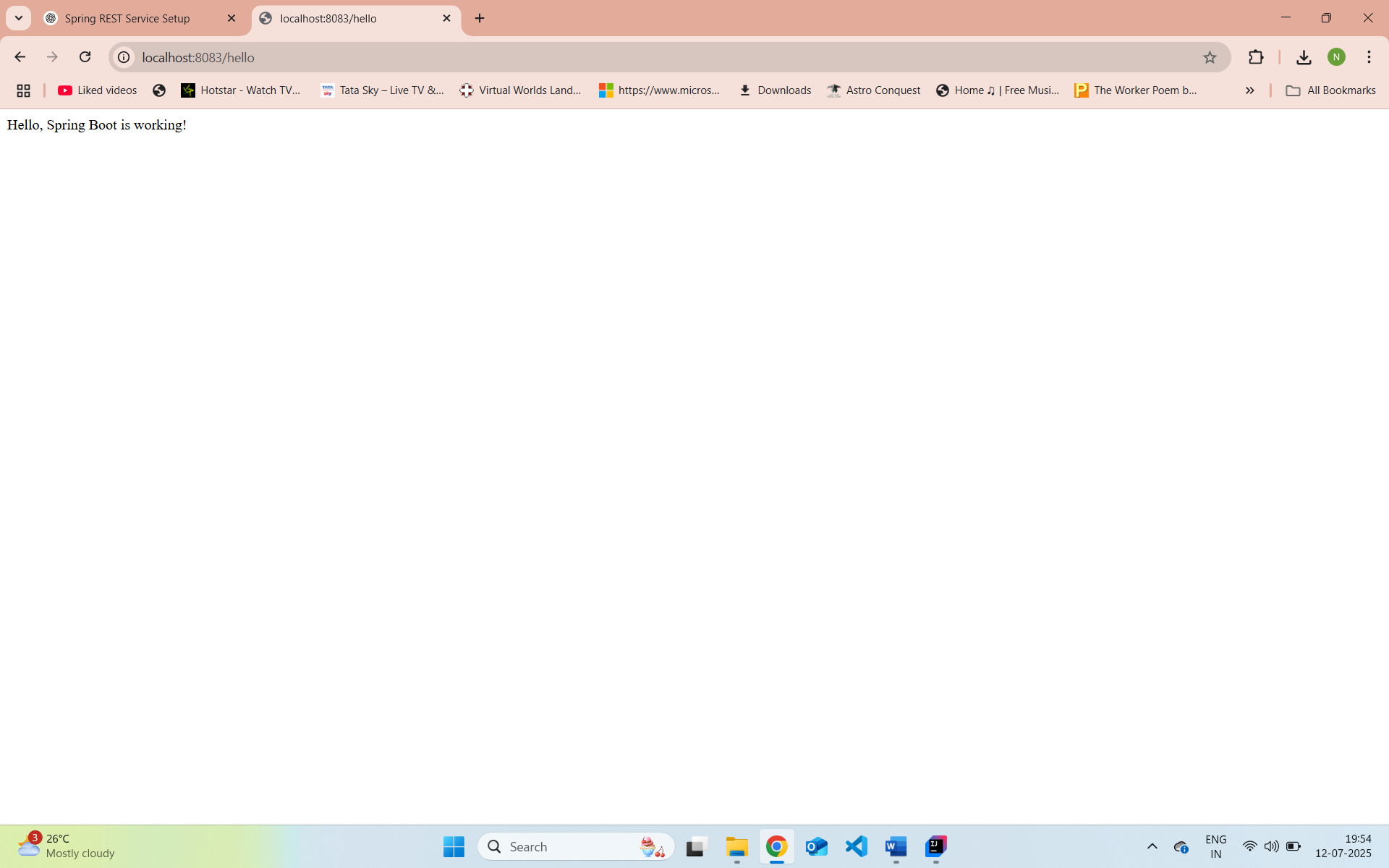
com.cognizant.spring\_learn.controller.HelloController



com.cognizant.spring\_learn.SpringLearnApplication:



Output :

[http://localhost:8083/hello](http://localhost:8083/hello)

**1. src/main/java – Application Code**

**This is the main source folder. Java packages and classes are written here.**

**2. src/main/resources – Configuration Files**

Contains:

* application.properties or application.yml: app-level config
* Static content: in /static folder
* Templates (for Thymeleaf/Freemarker): in /templates
* Logback or XML files (if needed)

**3. src/test/java – Test Code**

Same package structure as main. You’ll use it to:

* Write unit tests using **JUnit**
* Write integration tests with **Spring Test**

**4. SpringLearnApplication.java – Entry Point**

java

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

System.out.println("START");

}

}

* Starts the embedded **Tomcat** server
* Bootstraps **Spring context**
* Scans for beans (@Component, @Service, @RestController, etc.)
* **5. @SpringBootApplication – Key Annotation**
* This annotation does **3 things**:
* java
* CopyEdit
* @SpringBootApplication
* =
* @Configuration // Defines configuration class
* @EnableAutoConfiguration // Enables Spring Boot auto-config
* @ComponentScan // Enables component scanning in current package

**6. pom.xml – Maven Build Configuration**

Main responsibilities:

* Declares your project
* Sets parent (Spring Boot)
* Lists dependencies (Spring Web, DevTools, etc.)
* Configures Maven plugins

🔍 Example sections:

<parent>

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

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

<version>3.2.4</version>

</parent>

<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>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

**6.1 XML Configurations**

If using XML (e.g. for loading country.xml), place the file in src/main/resources, and use:

java

InputStream input = new ClassPathResource("country.xml").getInputStream();

// Parse it with JAXB/StAX/etc.

**6.2 View Dependency Hierarchy in Eclipse**

**Steps:**

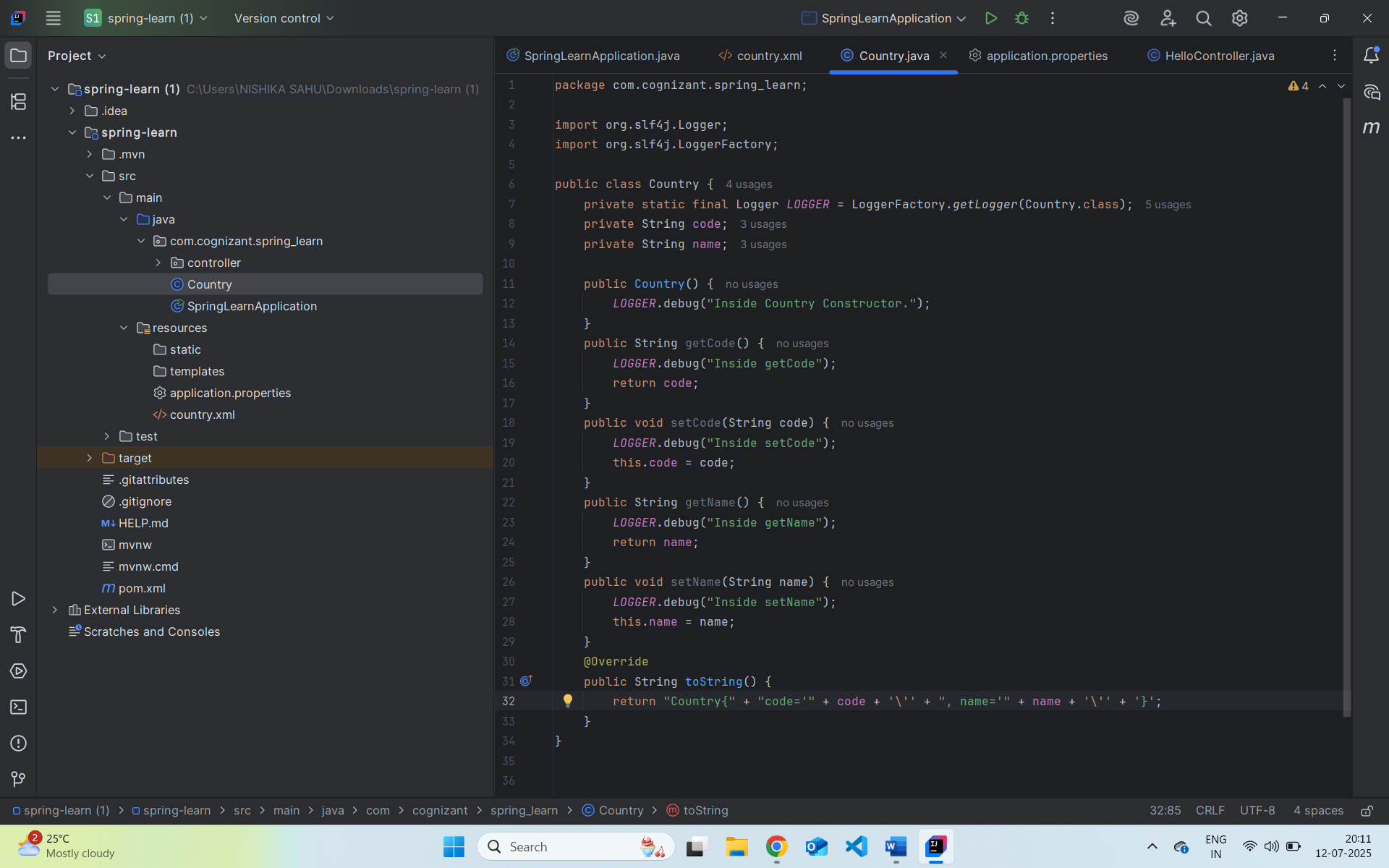
1. Right-click project → Maven → Show Dependency Hierarchy
2. You'll see:
   * Direct dependencies: spring-boot-starter-web, devtools
   * Transitive dependencies: spring-core, spring-context, jackson, tomcat, etc.

Useful for:

* Checking version conflicts
* Debugging build issues
* Verifying auto-config dependencies

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

com.cognizant.spring\_learn.Country.class

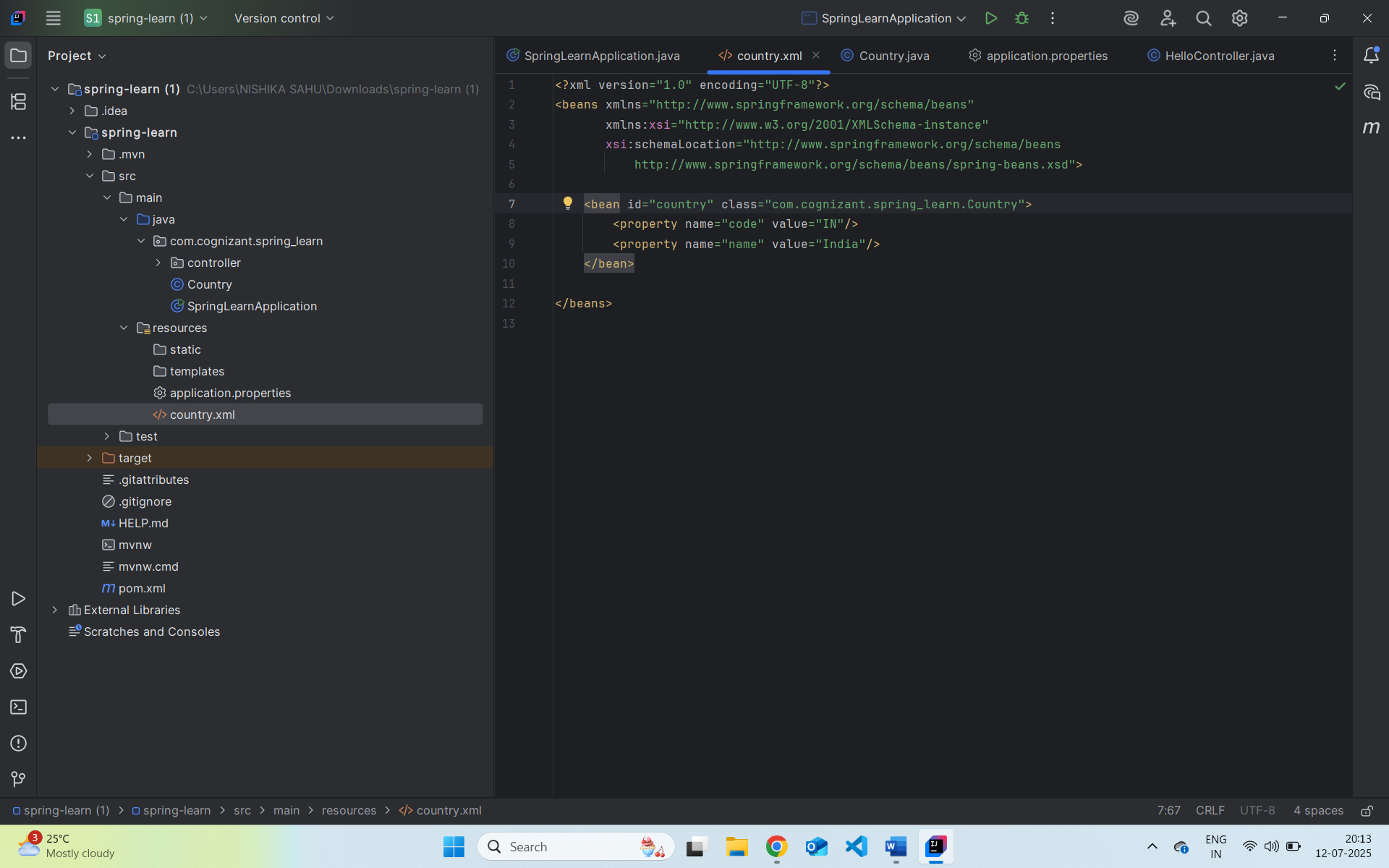


**com.cognizant.spring\_learn.SpringLearnApplication**

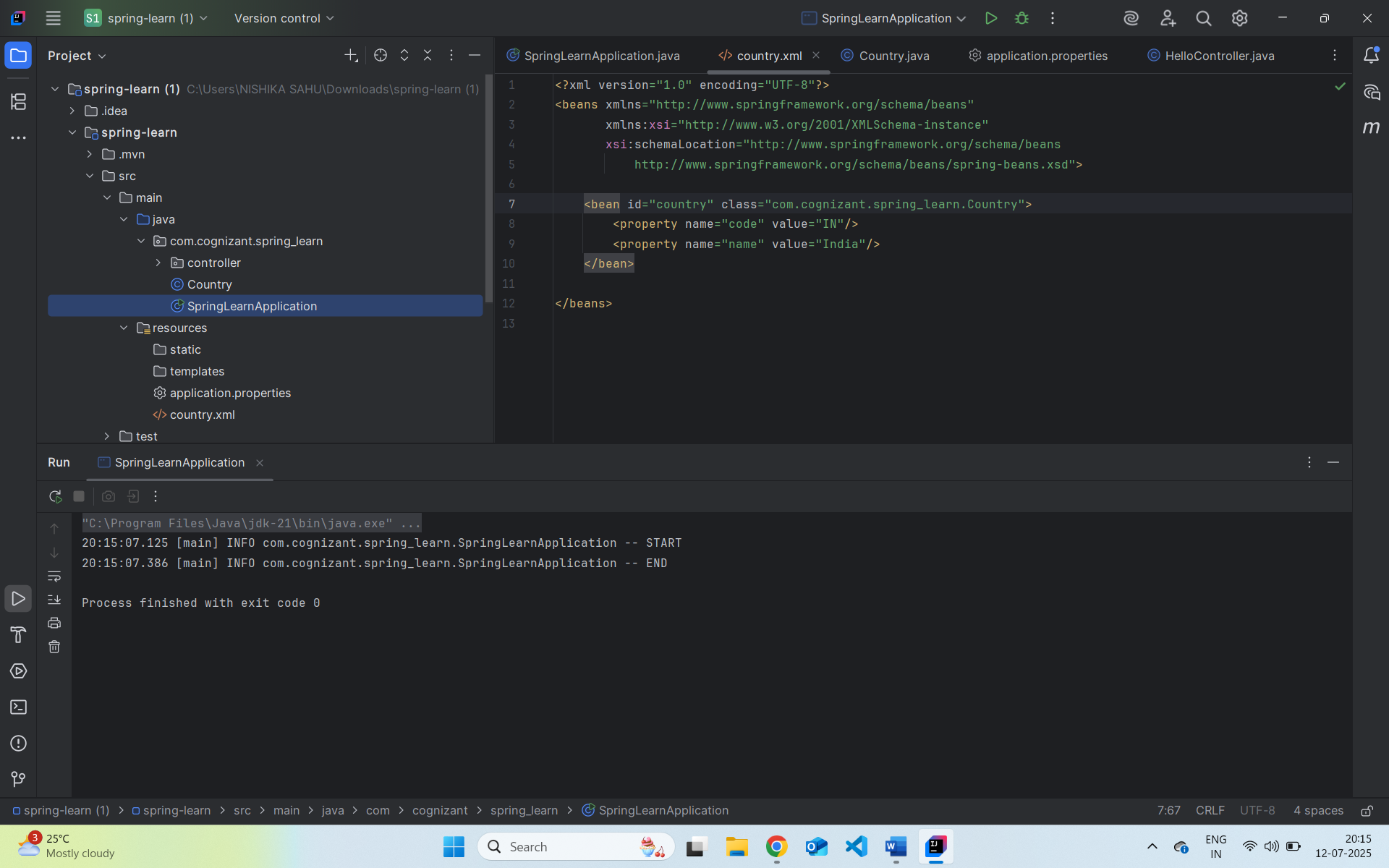
A computer screen shot of a program code

AI-generated content may be incorrect.

**Country.xml**



**Output**



**XML Configuration Tags:**

* <bean>: Defines a Spring bean.
* id: Unique name for the bean.
* class: Fully qualified class name to instantiate.
* <property>: Sets a property of the bean via a setter method.
* name: Corresponds to the property (e.g., setName()).
* value: The value to inject.

**ApplicationContext & ClassPathXmlApplicationContext**

* ApplicationContext: Central interface for Spring’s IoC container.
* ClassPathXmlApplicationContext: Loads the context from an XML file located in classpath.

**What happens during context.getBean():**

1. Spring container looks up the bean with the given ID.
2. Instantiates the bean if it hasn't already.
3. Injects dependencies (properties).
4. Returns the bean reference to the caller.