A screenshot of a computer

AI-generated content may be incorrect.

Files Structure

A screenshot of a computer program

AI-generated content may be incorrect.

**Country.java**

package com.cognizant.spring\_learns;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class Country {

private String code;

private String name;

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

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 + '\'' +

'}';

}

}

**Country.xml-configuration for bean**

<?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.spring\_learns.Country">

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

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

</bean>

</beans>

**Pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>1.0</version>

<dependencies>

<!-- Spring Core Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring Boot Starter -->

<dependency>

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

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

<version>2.7.18</version>

</dependency>

<!-- Spring Boot Test (required for @SpringBootTest) -->

<dependency>

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

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

<version>2.7.18</version>

<scope>test</scope>

</dependency>

<!-- SLF4J Logging Backend -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>1.7.36</version>

</dependency>

<!-- Spring Core (optional if already included) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Logging Dependency -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

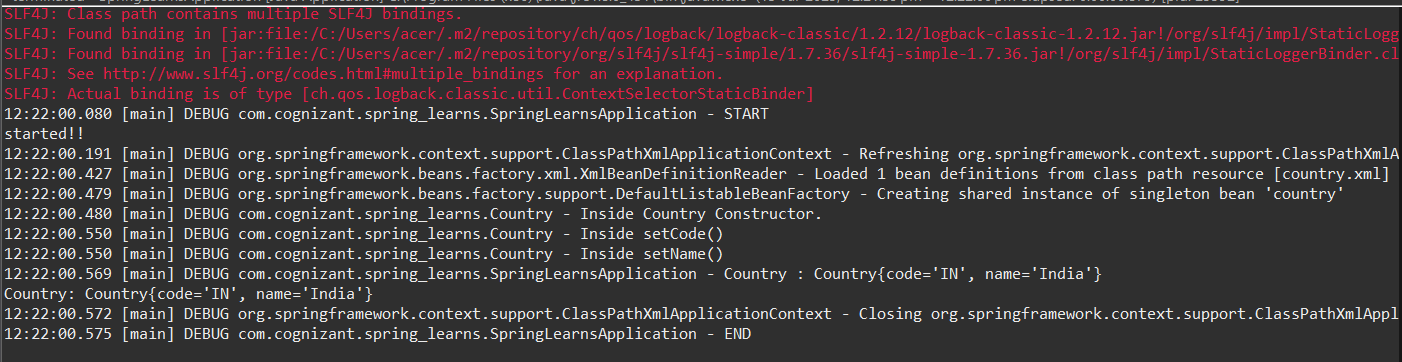
<version>1.7.36</version>

</dependency>

</dependencies>

</project>

**OUTPUT SCREENSHOT**



In Spring XML configuration, the <bean> tag is used to define objects that Spring manages. The id attribute specifies the bean name, and the class attribute specifies the fully qualified class name.

**Example:**

xml

<bean id="country" class="com.cognizant.spring\_learns.Country">

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

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

</bean>

* <property> tag is used to set values using setter methods.
* name refers to the property (like code), and value is the actual value passed ("IN").

To load this configuration, we use:

java

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

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

* ApplicationContext is the Spring container interface.
* ClassPathXmlApplicationContext loads the XML from src/main/resources.
* context.getBean() returns the fully initialized object with all property values set.