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

**Code –**

*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 + "]";

}

}

*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**>

*Logback.xml –*

<**configuration**>

<**appender** name=*"STDOUT"* class=*"ch.qos.logback.core.ConsoleAppender"*>

<**encoder**>

<**pattern**>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</**pattern**>

</**encoder**>

</**appender**>

<**root** level=*"debug"*>

<**appender-ref** ref=*"STDOUT"*/>

</**root**>

</**configuration**>

*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 main(String[] args) {

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

*displayCountry*();

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

}

public static void displayCountry() {

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

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

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

}

}

*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>springlearn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>springlearn</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>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

<!-- SLF4J + Logback for logging -->

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

</dependency>

<dependency>

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

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

</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>

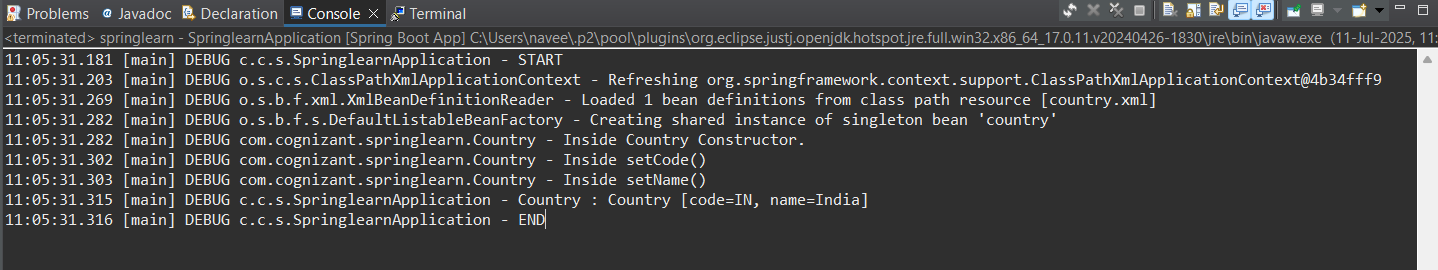
</plugin>

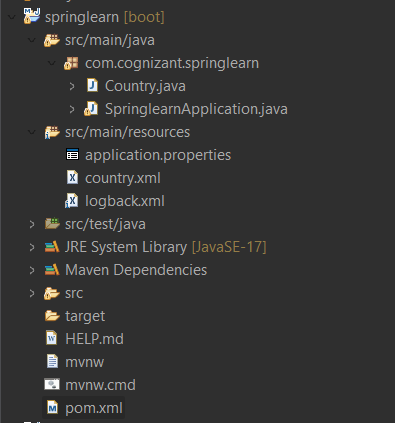
</plugins>

</build>

</project>

**Output – (Screenshots)**

****

****

**SME –**

* **bean tag, id attribute, class attribute, property tag, name attribute, value attribute**

**Answer -** The <bean> tag is used in Spring XML configuration to define a bean (an object) that will be managed by the Spring IoC (Inversion of Control) container.

**Attributes:**

* **id:** A unique identifier for the bean. You use this ID to retrieve the bean from the Spring container.
* **class:** Fully qualified name of the class that the Spring container will instantiate.

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

This tells Spring to create an object of Country class and name it as "country" within the container.

The <property> tag is used to inject values into the bean’s fields (using setter injection).

**Attributes:**

* **name**: Name of the Java property (must match the setter method in the class).
* **value**: Literal value to be set for that property.

**Ex: <property name="code" value="IN" />**

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

This will invoke the setCode("IN") and setName("India") methods of the Country class when the bean is initialized.

* **ApplicationContext, ClassPathXmlApplicationContext**

**Answer –**

* ApplicationContext is the central interface for accessing Spring’s IoC container.
* ClassPathXmlApplicationContext is a concrete implementation that loads the Spring configuration from an XML file in the classpath**.**

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

This line creates and initializes the Spring container by reading the country.xml file.

* **What exactly happens when context.getBean() is invoked**

**Answer –**

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

Here’s what happens:

1. Spring looks up the bean with ID "country" in the XML config.
2. It finds the class name com.cognizant.springlearn.Country.
3. It creates an instance of that class (using the default constructor).
4. It injects the properties using the setters (setCode(), setName()).
5. The fully initialized bean is returned.

This is the essence of Dependency Injection in Spring.