Assignment: Hands-on 4 – Spring Core – Load Country from Spring Configuration XML

# Objective

To create a Country bean with ISO code and name, configure it in a Spring XML file, and retrieve it in a Spring Boot application using ApplicationContext.

# Steps Followed

## Step 1: Created Spring XML Configuration (country.xml)

Location: src/main/resources/country.xml

Configuration content:

<?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  
 https://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>

## Step 2: Created Country.java Class

Package: com.cognizant.springlearn

Class content:

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 + "]";  
 }  
}

## Step 3: Modified SpringLearnApplication.java

Added displayCountry() method:

public static void displayCountry() {  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
  
 Country country = context.getBean("country", Country.class);  
 LOGGER.debug("Country : {}", country.toString());  
}

Called displayCountry() in main():

public static void main(String[] args) {  
 LOGGER.info("START");  
 SpringApplication.run(SpringLearnApplication.class, args);  
 displayCountry();  
 LOGGER.info("END");  
}

# Project Structure Overview

spring-learn/  
├── src/  
│ └── main/  
│ ├── java/  
│ │ └── com/cognizant/springlearn/  
│ │ ├── SpringLearnApplication.java  
│ │ └── Country.java  
│ └── resources/  
│ ├── application.properties  
│ └── country.xml

# Output (Console Logs)

INFO com.cognizant.springlearn.SpringLearnApplication - START  
DEBUG com.cognizant.springlearn.Country - Inside Country Constructor.  
DEBUG com.cognizant.springlearn.Country - Inside setCode()  
DEBUG com.cognizant.springlearn.Country - Inside setName()  
DEBUG com.cognizant.springlearn.Country - Inside getCode()  
DEBUG com.cognizant.springlearn.Country - Inside getName()  
DEBUG com.cognizant.springlearn.SpringLearnApplication - Country : Country [code=IN, name=India]  
INFO com.cognizant.springlearn.SpringLearnApplication - END

# Concepts Explained

## bean tag and attributes

• <bean>: Declares a Spring-managed object.

• id: Unique identifier to refer to this bean.

• class: Fully qualified name of the class.

• <property>: Used to inject values into bean's fields.

- name: Name of the field.

- value: Value to inject.

## ApplicationContext vs ClassPathXmlApplicationContext

• ApplicationContext: Spring’s central interface for managing beans.

• ClassPathXmlApplicationContext: Loads bean definitions from XML in the classpath.

## What happens in context.getBean()

1. Spring parses country.xml.

2. Finds the <bean> with ID 'country'.

3. Instantiates the Country class.

4. Calls constructor → Logs 'Inside Country Constructor.'

5. Injects values → Calls setter methods.

6. Returns the fully initialized bean.

# Conclusion

Successfully demonstrated how to define and load a custom Country bean using Spring XML configuration, retrieve it using ApplicationContext, and verify bean lifecycle behavior via logs.