**Hands-On 4: Spring Core – Load Country from Spring Configuration XML**

**Objective:**

The goal of this task is to **load a country’s data (like code and name)** from an **XML-based Spring configuration file** and display it. This simulates how a website, such as an airline booking portal, may need to load a list of countries for user selection (like in a dropdown).

**Scenario:**

An airline booking website supports bookings from multiple countries. It must display countries like:

* US – United States
* DE – Germany
* IN – India
* JP – Japan

Each country has a **two-character ISO code** and a **name**. In this exercise, we will:

* Store this data in an **XML file** using **Spring Bean configuration**
* Load that data in a **Java class**
* Display it using **Spring’s ApplicationContext**

**Steps to Implement:**

**Step 1: Create the Country XML configuration file**

**Create a file named country.xml in src/main/resources.**

<?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\_learn.Country">

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

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

</bean>

</beans>

**Step 2: Create the Country Java class**

package com.cognizant.spring\_learn;

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: Load the XML bean using ClassPathXmlApplicationContext**

Update your SpringLearnApplication class:

package com.cognizant.spring\_learn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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.info("START");

displayCountry();

LOGGER.info("END");

}

public static void displayCountry() {

try (ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("country.xml")) {

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

System.out.println(">>> Bean loaded: " + country.toString());

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

} catch (Exception e) {

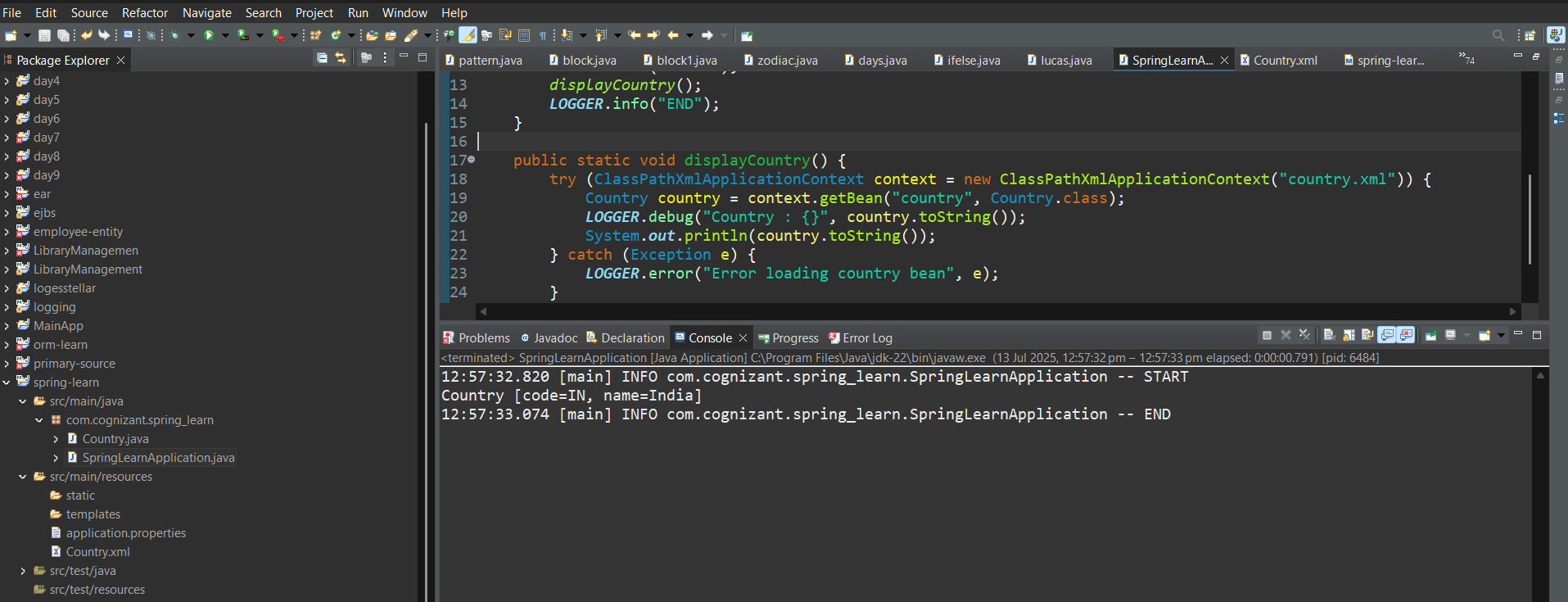
LOGGER.error("Error loading country bean", e);

}

}

}

**Output ScreenShot:**

****