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

# Problem Statement

An airline website supports bookings in 4 countries. The ISO code and name of each country must be configured using Spring XML. A program must be written to load this configuration and display it.

# Steps to Implement

1. Create an XML configuration file `country.xml`.  
2. Add a <bean> configuration for a country.  
3. Create a `Country` Java class with:  
 - Variables: `code`, `name`  
 - Constructor with debug log  
 - Getters and setters with debug logs  
 - `toString()` method  
4. Modify `SpringLearnApplication.java` to include a method `displayCountry()`:  
 - Load `country.xml` using `ClassPathXmlApplicationContext`  
 - Get the `country` bean and log the output.  
5. Invoke `displayCountry()` from `main()` and check logs.

# country.xml Configuration

```xml  
<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>  
```

# Country.java

```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 + "]";  
 }  
}  
```

# SpringLearnApplication.java - Add displayCountry()

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

# Invoke displayCountry() in main()

```java  
public static void main(String[] args) {  
 LOGGER.info("Start");  
 SpringApplication.run(SpringLearnApplication.class, args);  
 displayCountry();  
 LOGGER.info("End");  
}  
```

# Explanation of Spring Concepts

- `<bean>`: Defines a bean (object) to be managed by Spring.  
- `id`: Unique name for the bean.  
- `class`: Fully qualified class name of the bean.  
- `<property>`: Used to set values for the bean fields.  
- `name`: Field in the class to set.  
- `value`: Value assigned to the property.  
  
- `ApplicationContext`: Central interface to access Spring beans.  
- `ClassPathXmlApplicationContext`: Loads Spring config from classpath XML.  
  
- `context.getBean("country")`:   
 - Loads the bean by ID.  
 - Instantiates the class, injects dependencies, returns the object.