# Spring Boot Assignment - Hands-on 2 (Country Bean)

Spring Core – Load Country from Spring Configuration XML

## Steps to Implement

This task involves storing country details in a Spring XML configuration file and accessing them using a Spring Bean.

## Spring Configuration (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  
 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>

## 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("Getting Code: {}", code);  
 return code;  
 }  
  
 public void setCode(String code) {  
 LOGGER.debug("Setting Code: {}", code);  
 this.code = code;  
 }  
  
 public String getName() {  
 LOGGER.debug("Getting Name: {}", name);  
 return name;  
 }  
  
 public void setName(String name) {  
 LOGGER.debug("Setting Name: {}", name);  
 this.name = name;  
 }  
  
 @Override  
 public String toString() {  
 return "Country [code=" + code + ", name=" + name + "]";  
 }  
}

## Updated SpringLearnApplication.java

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

Call `displayCountry();` inside the `main()` method after `displayDate();`

## SME Explanation Points

• <bean>: Defines a bean in the Spring container.  
• id: Unique identifier for the bean.  
• class: Fully qualified class name for the bean.  
• <property>: Used to set values in the bean's fields.  
• name: Name of the setter property (matches setter method).  
• value: Value to inject into the property.  
  
• ApplicationContext: Central interface for Spring container, used to retrieve beans.  
• ClassPathXmlApplicationContext: Loads the Spring context from an XML file in classpath.  
• context.getBean(): Instantiates the bean (if not already created), sets properties, and returns the instance.