**Spring Core and Maven Assignment Problems**

### Exercise 1: Create a simple Spring Bean using XML configuration

**Problem:** Define a simple Country bean and display its properties using Spring’s ApplicationContext.

**Files:**

* 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.example.Country">  
 <property name="name" value="India" />  
 <property name="code" value="IN" />  
 </bean>  
</beans>

* Country.java:

package com.example;  
  
public class Country {  
 private String name;  
 private String code;  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
  
 @Override  
 public String toString() {  
 return "Country{name='" + name + "', code='" + code + "'}";  
 }  
}

* SpringLearnApplication.java:

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

**Output:**

Country{name='India', code='IN'}

### Exercise 2: Use Constructor Injection in Spring XML

**Problem:** Create a Department bean with a constructor that takes id and name.

**Files:**

* department.xml:

<bean id="department" class="com.example.Department">  
 <constructor-arg value="101" />  
 <constructor-arg value="HR" />  
</bean>

* Department.java:

package com.example;  
  
public class Department {  
 private int id;  
 private String name;  
  
 public Department(int id, String name) {  
 this.id = id;  
 this.name = name;  
 }  
  
 @Override  
 public String toString() {  
 return "Department{id=" + id + ", name='" + name + "'}";  
 }  
}

* Test in main class:

ApplicationContext context = new ClassPathXmlApplicationContext("department.xml");  
Department dept = context.getBean("department", Department.class);  
System.out.println(dept);

**Output:**

Department{id=101, name='HR'}

### Exercise 3: Dependency Injection Using Maven

**Problem:** Add Spring dependencies using Maven.

**pom.xml:**

<dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.20</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.7.36</version>  
 </dependency>  
 <dependency>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-classic</artifactId>  
 <version>1.2.11</version>  
 </dependency>  
</dependencies>

This ensures that Spring and SLF4J libraries are pulled automatically.

### Exercise 4: ApplicationContext and BeanFactory

**Problem:** Demonstrate both ApplicationContext and BeanFactory usage.

**Code:**

ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");  
MyBean myBean = context.getBean("myBean", MyBean.class);  
  
BeanFactory factory = new XmlBeanFactory(new ClassPathResource("beans.xml"));  
MyBean anotherBean = factory.getBean("myBean", MyBean.class);

**Note:** BeanFactory is a lower-level container; ApplicationContext is more commonly used in modern applications.