**Hands on 4**

**Spring Core – Load Country from Spring Configuration XML**

**SpringLearnApplication.java:**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SpringLearn1Application {

private static final Logger *LOGGER* = LoggerFactory.*getLogger*(SpringLearn1Application.class);

public static void main(String[] args) {

*displayCountry*();

}

public static void displayCountry() {

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

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

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

}

}

}

**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("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 + '\'' + '}';

}

}

**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.cognizant.springlearn.Country">

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

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

</bean>

</beans>

**logBack.xml:**

<configuration>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

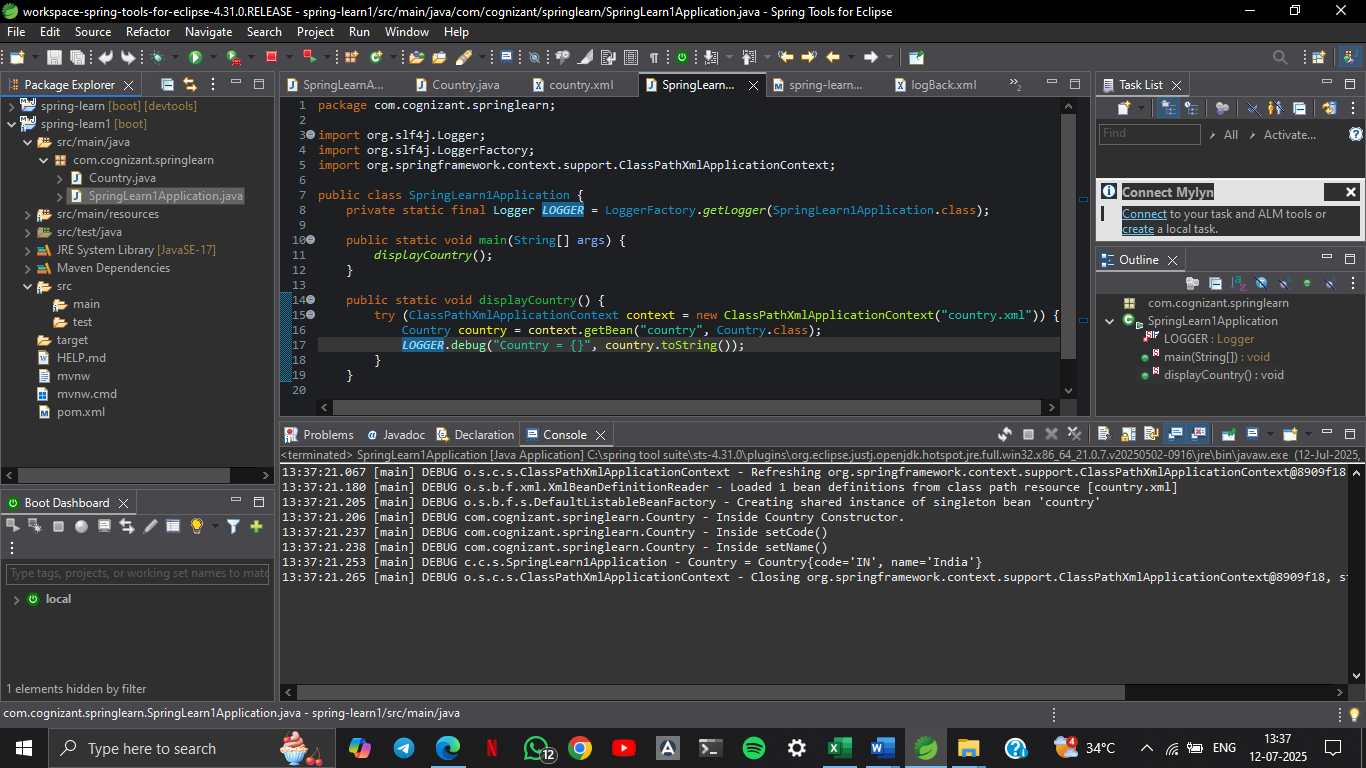
<root level="DEBUG">

<appender-ref ref="STDOUT" />

</root>

</configuration>

**OUTPUT:**



SME Explanation:

**1.<bean> tag:**

* Defines a Spring-managed object (bean).
* <bean id="country" class="com.cognizant.springlearn.Country">
* id: Unique identifier for the bean (country)
* class: Fully qualified class name to be instantiated

**2. <property> Tag:**

* Used to inject values into the bean’s fields via **setters**.
* <property name="code" value="IN"/>
* name: Name of the property (setCode)
* value: The actual value to assign ("IN")

**3. ApplicationContext**

* Interface in Spring for accessing the Spring container.
* Provides more enterprise features like internationalization, events, etc.

**4. ClassPathXmlApplicationContext**

* A concrete implementation of ApplicationContext.
* Loads the configuration XML file from the classpath.

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

**5. What exactly happens when context.getBean() is invoked**

* Spring parses country.xml and creates a Country object.
* Constructor is called → Inside Country Constructor.
* Setters are called to inject code and name.
* When you call context.getBean():
* You get the fully initialized bean.
* If debug logs are on, you'll see output of getter and toString().