WEEK – 4

Spring Core – Load Country from Spring Configuration XML

Nipuna

Superset id: 6432842

An airlines website is going to support booking on four countries. There will be a drop down on the home page of this website to select the respective country. It is also important to store the two-character ISO code of each country. 

|  |  |
| --- | --- |
| **Code** | **Name** |
| US | United States |
| DE | Germany |
| IN | India |
| JP | Japan |

Above data has to be stored in spring configuration file. Write a program to read this configuration file and display the details.  
  
Steps to implement

* Pick any one of your choice country to configure in Spring XML configuration named country.xml.
* Create a bean tag in spring configuration for country and set the property and values

    <bean id="country" class="com.cognizant.springlearn.Country">

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

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

    </bean>

* Create Country class with following aspects:
  + Instance variables for code and name
  + Implement empty parameter constructor with inclusion of debug log within the constructor with log message as “Inside Country Constructor.”
  + Generate getters and setters with inclusion of debug with relevant message within each setter and getter method.
  + Generate toString() method
* Create a method displayCountry() in SpringLearnApplication.java, which will read the country bean from spring configuration file and display the country details. ClassPathXmlApplicationContext, ApplicationContext and context.getBean(“beanId”, Country.class). Refer sample code for displayCountry() method below.

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

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

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

* Invoke displayCountry() method in main() method of SpringLearnApplication.java.
* Execute main() method and check the logs to find out which constructors and methods were invoked.

SME to provide more detailing about the following aspects:

* bean tag, id attribute, class attribute, property tag, name attribute, value attribute
* ApplicationContext, ClassPathXmlApplicationContext
* What exactly happens when context.getBean() is invoked

Solution:

1)SpringLearnApplication.java :

package com.cognizant.springlearn;

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(">>> Country Bean: " + country); // ← ADD THIS

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

} catch (Exception e) {

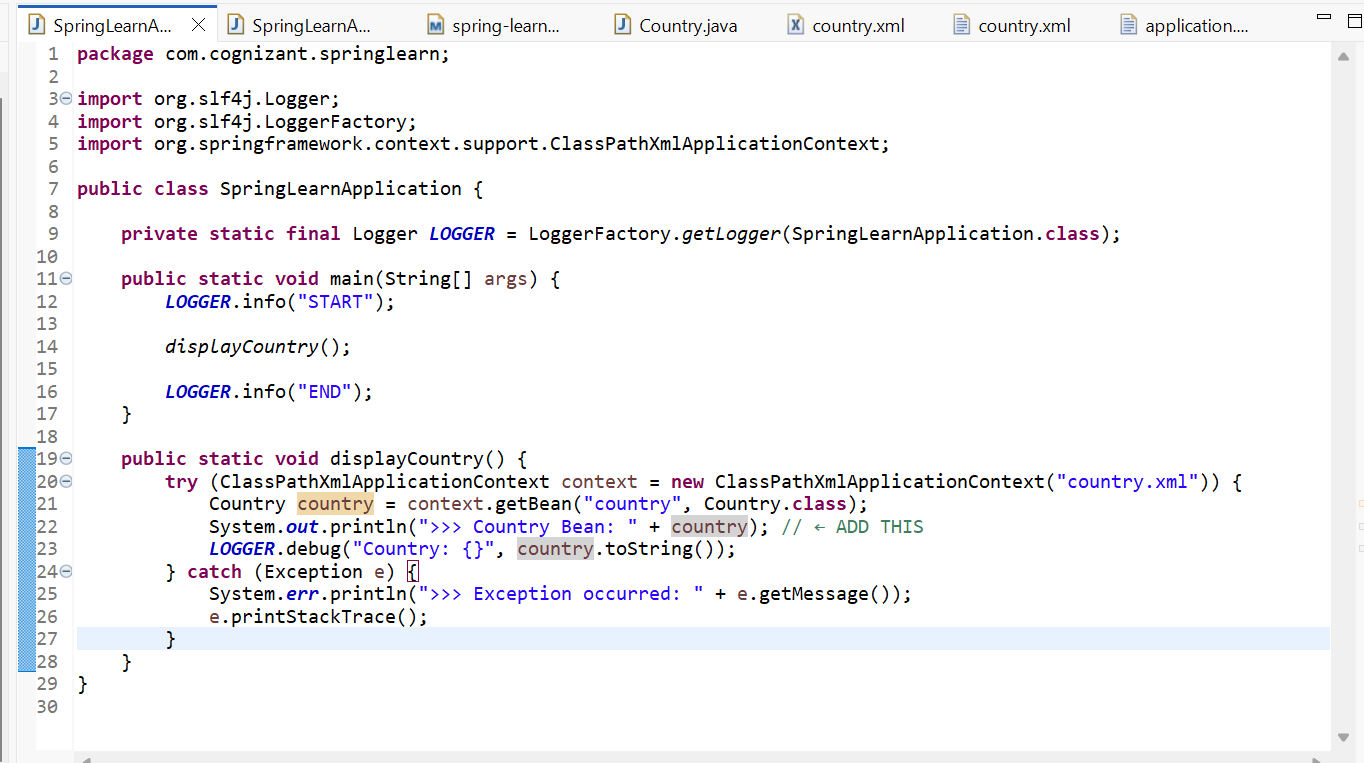
System.*err*.println(">>> Exception occurred: " + e.getMessage());

e.printStackTrace();

}

}

}



2) 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;

}

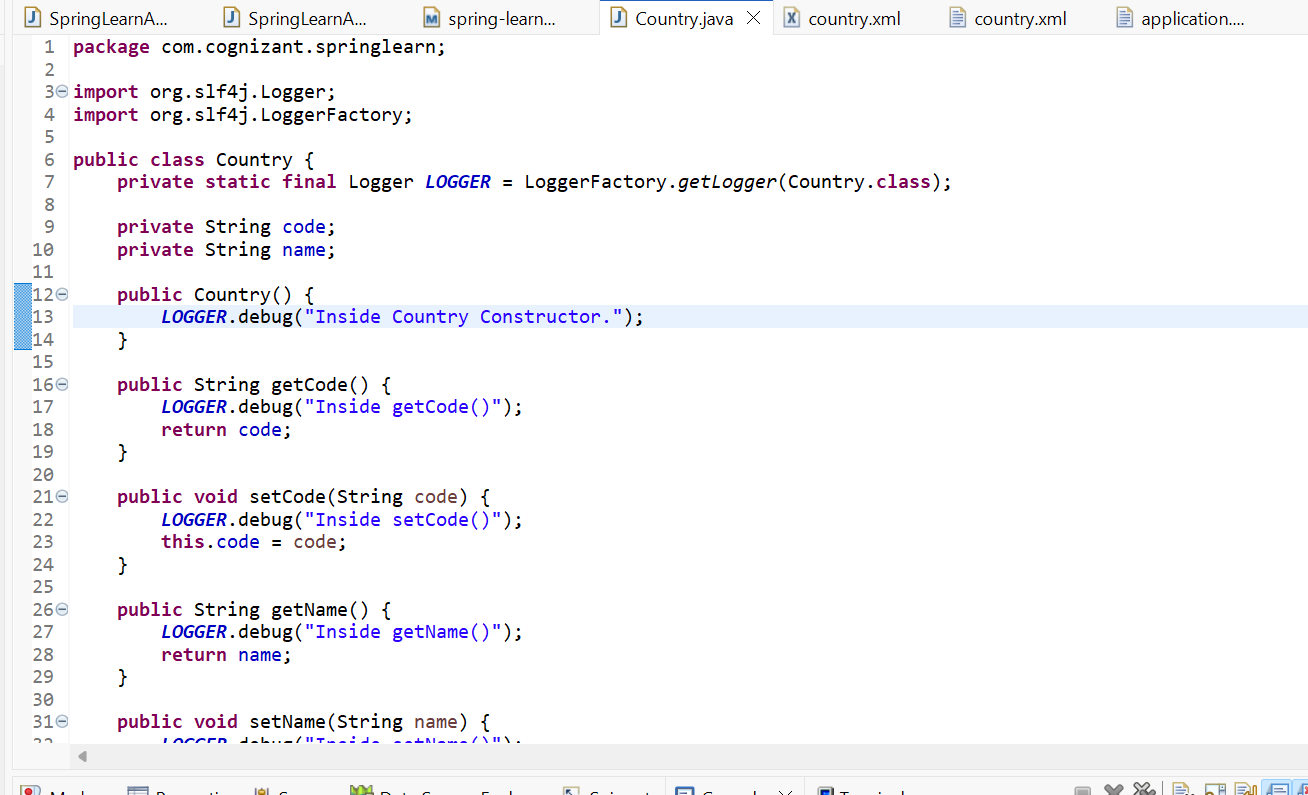
@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}



3) 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">

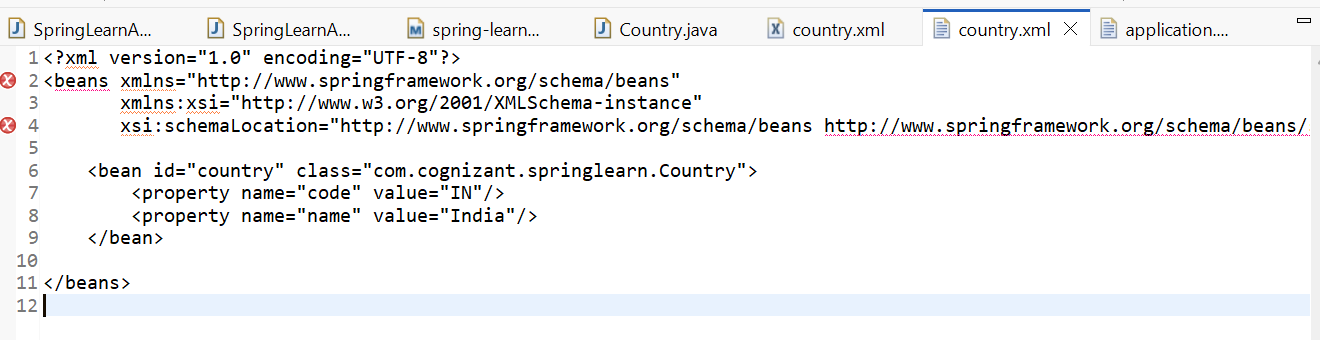
<bean id="country" class="com.cognizant.springlearn.Country">

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

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

</bean>

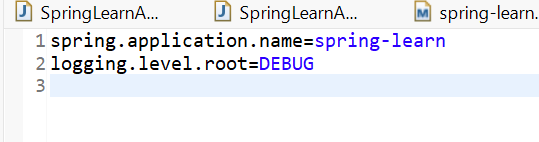
</beans>



4) application.properties :

spring.application.name=spring-learn

logging.level.root=DEBUG



5) pom.xml :

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>spring-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-learn</name>

<description>Spring Learn Project</description>

<properties>

<java.version>17</java.version>

<spring.version>5.3.34</spring.version>

</properties>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- SLF4J + Logback -->

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.11</version>

</dependency>

<!-- Lombok (optional) -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.28</version>

<scope>provided</scope>

</dependency>

<!-- JUnit for testing -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<version>2.7.18</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<resources>

<resource>

<directory>src/main/resources</directory>

<includes>

<include>\*\*/\*.xml</include>

<include>\*\*/\*.properties</include>

</includes>

</resource>

</resources>

</build>

</project>





Output:

