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

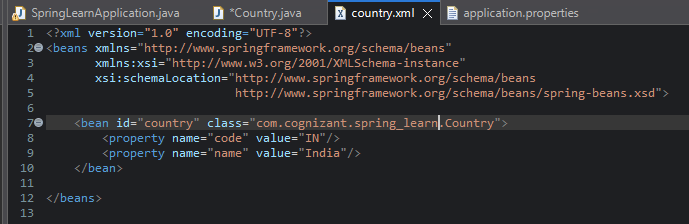
* Pick any one of your choice country to configure in Spring XML configuration named country.xml.
* Create a bean tag in spring configuration for country and set the property and values

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

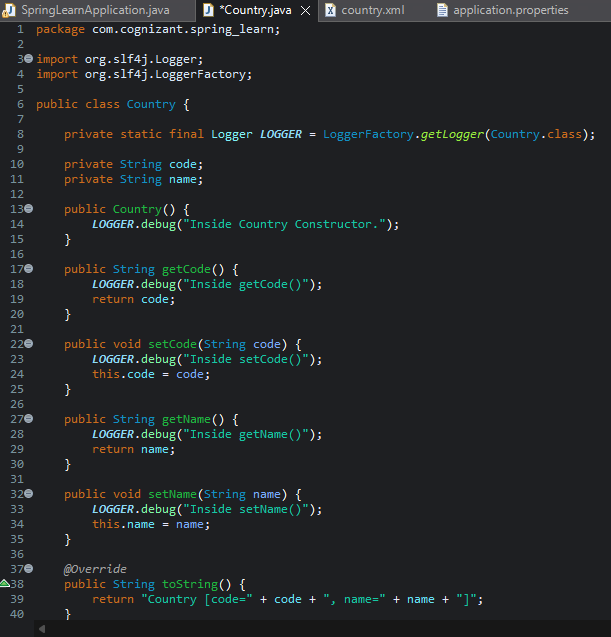
        <property name="code" value="IN" />

        <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



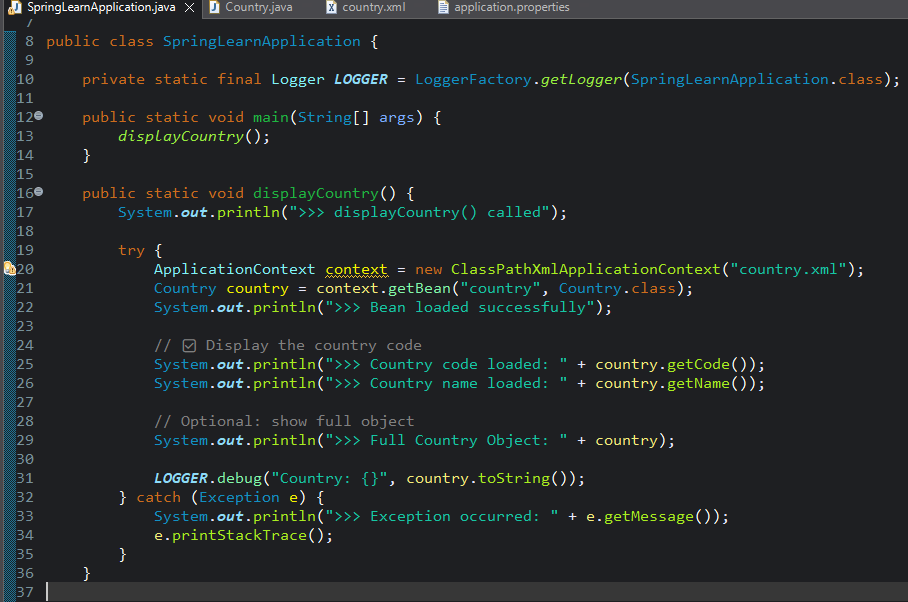
Create a method displayCountry() in SpringLearnApplication.java, which will read the country bean from spring configuration file and display the country details. ClassPathXmlApplicationContext, ApplicationContext and context.getBean(“beanId”, Country.class). Refer sample code for displayCountry() method below.

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

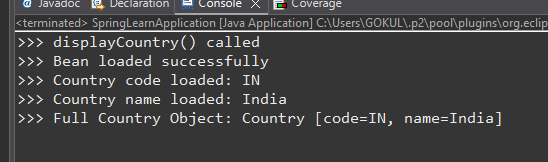
Country country = (Country) context.getBean("country", Country.class);

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.



Output:



SME to provide more detailing about the following aspects:

**1..bean tag, id attribute, class attribute, property tag, name attribute, value attribute**

**🔹 bean tag**

Defines a Spring bean. Format:

<bean id="beanId" class="fully.qualified.ClassName">

* id: Unique identifier of the bean in Spring context.
* class: Fully qualified class name of the object to be instantiated.

**🔹 property tag**

Used to inject values into JavaBean properties.

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

* name: Matches the setter method (e.g., setCode() for code)
* value: The value to set

**2..ApplicationContext, ClassPathXmlApplicationContext**

| Feature | ApplicationContext | ClassPathXmlApplicationContext |
| --- | --- | --- |
| Interface | ApplicationContext is an interface | ClassPathXmlApplicationContext is its implementation |
| Purpose | Provides access to beans, messages, env | Loads XML config files from classpath |
| Features | Supports internationalization, AOP, etc. | Basic Spring XML loading |

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

Spring looks up the bean definition by id (country).

It finds the class (Country) and creates an instance.

It injects the properties (code, name) using setters.

The bean is returned, fully initialized.

Any logs in constructor and setters will be triggered.