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

**<bean>**: This tag is used to declare a bean in the Spring configuration file. A bean is any Java object that Spring will manage — it will create, initialize, and inject it as needed.

**id**: The id attribute inside the <bean> tag is used to uniquely identify the bean within the Spring container. For example, id="country" means you can refer to this bean using the name country.

**class**: The class attribute specifies the fully qualified name of the Java class that the bean is an instance of. For example, class="com.cognizant.springlearn.Country" tells Spring to create an object of the Country class.

**<property>**: This tag is used within a bean declaration to set the properties of the bean using setter methods. Spring will call the appropriate setter to assign the value.

**name**: The name attribute of the <property> tag must match the name of the property in the Java class (i.e., it should match the corresponding setter method, like setCode or setName).

**value**: The value attribute provides the literal value to be assigned to the property. For example, value="IN" will be passed to the setCode() method.

**ApplicationContext & ClassPathXmlApplicationContext**

* **ApplicationContext** is the central Spring container interface for managing beans, life cycle, and configurations.
* **ClassPathXmlApplicationContext** loads Spring XML configuration from the classpath and initializes the beans defined in it.

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

This reads country.xml from src/main/resources.

**🔹 What happens in context.getBean("country", Country.class)?**

1. Spring finds the <bean id="country"> in country.xml.
2. It creates a new object of class com.cognizant.springlearn.Country.
3. It calls the **constructor** → debug logs "Inside Country Constructor."
4. It sets properties using **setCode()** and **setName()** → debug logs appear.
5. Returns the fully initialized bean.
6. Finally, your code logs the result using toString().