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

**Task:**

**<bean> :** We use this in a JSP tag library element for accessing or creating JavaBeans, and a Spring Framework annotation for defining beans within the application context.In our task we used it as :  
<bean id="country" class="com.cognizant.spring\_learn.Country">

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

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

</bean>

**<id> :** The <id> tag gives a name to the bean so that we can refer to it later.In our task, id="country" means we can get the object using context.getBean("country").It’s like giving the bean a nickname that we can use to get it later.

**<class> :** It is used to assign one or more CSS classes to an HTML element. By using classes, you can group elements together and apply consistent styles across them, streamlining both design and functionality.

**<property> :** It is a way to associate a value or attribute with an object or element, often for identification, organization, or display purposes.

**name attribute :** The name attribute specifies a name for an HTML element.This name attribute can be used to reference the element in a JavaScript.

**Value attribute :** The value attribute in HTML is used to specify the value of the element with which it is used. It has different meaning for different HTML elements. Usage: It can be used with the following elements: <input>, <button>, <meter>, <li>, <option>, <progress>, and <param>, <output>.

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

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

LOGGER.debug("Country : {}", country.toString());  
  
**What exactly happens when context.getBean() is invoked :**

We used it in our task here   
Country country = context.getBean("country", Country.class);  
When we call context.getBean() in a Spring application, Spring looks for the bean with the specified id in the XML configuration file. It then creates an object of the class using its no-argument constructor. After that, it sets the values for the properties using setter methods defined in the class. Finally, it returns the fully initialized bean so we can use it in our application. If the bean is already created , Spring just returns the same instance instead of creating a new one.

**Output:**  
 