**SME Explanation – Spring Core: Load Country from XML Configuration**

**1. Explanation of <bean> Tag, id, class, <property>, name, and value Attributes**

In a Spring XML configuration file, the <bean> tag is used to declare a Spring-managed object (bean). The id attribute uniquely identifies the bean so it can be retrieved using methods like getBean(). The class attribute defines the fully qualified class name (FQCN) of the bean to be created. Inside a <bean>, we use one or more <property> tags to inject values into the bean’s fields using their corresponding setter methods. The name attribute inside <property> specifies which Java property is being set, and the value attribute provides the literal value to inject. This is part of Spring's **Dependency Injection (DI)** mechanism, which simplifies configuration and reduces tight coupling in code.

----------------------------------------------------------------------------------

**2. Explanation of ApplicationContext and ClassPathXmlApplicationContext**

ApplicationContext is the primary interface of the Spring IoC container and is responsible for loading and managing beans. It handles bean lifecycle, dependency injection, internationalization, event propagation, and more. ClassPathXmlApplicationContext is one of its concrete implementations and is used to load XML configuration files from the classpath (commonly from src/main/resources). When an instance of ClassPathXmlApplicationContext is created with the XML filename, it reads the file, creates all defined beans, injects their dependencies, and prepares them for use in the application. It is typically used in simple standalone Java applications to bootstrap the Spring context.

-----------------------------------------------------------------------------

**🔸 3. What Happens When context.getBean() Is Invoked**

When the method context.getBean("beanId", ClassName.class) is called, the Spring container checks the configuration to locate the bean with the specified id. If found, Spring either returns the existing instance (if already created) or creates a new one by calling the no-argument constructor of the class. It then uses setter methods to inject values as defined in the <property> tags in the XML. This entire process follows the **Inversion of Control (IoC)** principle, where Spring, not the developer, controls object creation and dependency wiring. The method finally returns a fully-initialized bean that can be used directly in your application logic.