Spring XML Config

By creating a spring project without Spring Boot, we can gain a deeper understanding of how the spring framework works and how it manages objects behind the scenes.
Returning to the code, we have successfully created a basic spring project. In this project, the method "build" in the "dev" class is being called in the main class.

One might question the necessity of working with Spring, especially in XML-based configuration, when Spring Boot is available. However, it is important to note that many legacy projects in the enterprise market still rely on Spring and XML configuration. Therefore, familiarizing oneself with Spring and XML configuration can be highly advantageous and provide a clearer understanding of these projects.

We have generated an XML file called spring.xml, which serves as our configuration file. This file contains information about the available classes and their objects that are managed by the Spring framework within the container. These classes are referred to as beans.
XML, like HTML, utilizes tags, with certain default tags having their own definitions specified within the DTD file (Document Type Definition).

It also follows syntax; using the <beans> tag, write down the number of beans present inside the project.

```
<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-

    <bean id="dev" class="com.telusko.Dev">


    </bean>
```

We have written the fully qualified name inside the bean tag along with its id, also known as the alias name, instead of writing the qualified name every time. We have also added the DTD information inside the tag, so it recognizes the definition of it.
Now, let's run the project and see the results.

Output

```
/Library/Java/JavaVirtualMachines/amazon-corretto-21.jdk/Contents/Home/bin/java ...
working on Awesome Project

Process finished with exit code 0
```

In other words, we asked Spring to create an object inside the container using this xml configuration file.

.

Let's create another class, Laptop, with a method called compile that prints "compiling" and asks Spring to create an object for it.

```java
package com.telusko;

public class Laptop {

    public void compile(){
        System.out.println("Compiling");
    }

}
```

Adding another <bean> tag inside the xml file and the full qualified class name of the laptop with its id and for confirming how many objects are in the container, we have called the constructor for both classes, printing their respective names.

```xml
<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

    <bean id="dev" class="com.telusko.Dev">
    </bean>

    <bean id="dev1" class="com.telusko.Dev">
    </bean>

    <bean id="laptop" class="com.telusko.Laptop">
    </bean>
```

Output:

```
/Library/Java/JavaVirtualMachines/amazon-corretto-21.jdk/Contents/Home/bin/java ...
Dev Constructor
Dev Constructor
Laptop Constructor
```

Essentially, we understand how Spring is instructed to generate objects within the container using XML-based configuration.
The upcoming chapter will cover the concepts of setter and constructor injection.

.

Thanks for reading...

Happy Learning 😊