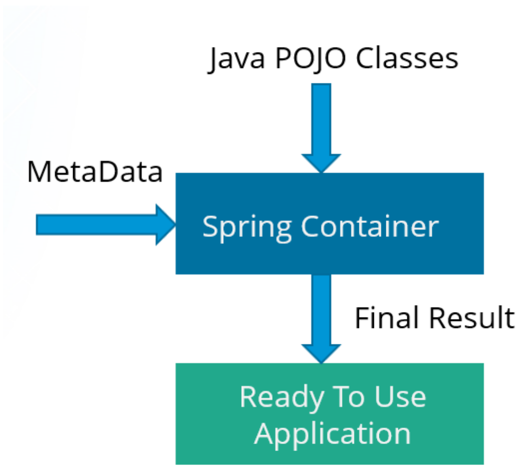
Spring IOC Container

20 Temmuz 2018 Cuma 11:41

In Spring Framework we have IoC container (Inversion of Control) which receives metadata from either an XML file, Java annotations, or Java code and works accordingly. IoC adds the flexibility and control of application, and provides a central place of configuration management for Plain Old Java Objects (POJO) of our application.



Above diagram shows how Spring makes use of Java POJO classes and configuration metadata to produce a fully configured and executable system or application. There are two types of IoC containers:

1. BeanFactory
2. ApplicationContext

Let me explain them in detail.

Bean Factory:

- It is an interface defined in org.springframework.beans.factory.BeanFactory.
- Bean Factory provides the basic support for Dependency Injection.
- It is based on factory design pattern which creates the beans of any type.
- BeanFactory follows lazy-initialization technique which means beans are loaded as soon as bean factory instance is created but the beans are created only when *getBean()* method is called.
- The XmlBeanFactory is the implementation class for the BeanFactory interface. To use the BeanFactory, you need to create the instance of XmlBeanFactory class as shown below:

```
BeanFactory beanFactory = new XmlBeanFactory(new ClassPathResource("beans.xml"));
```

ApplicationContext

- It is an interface defined in org.springframework.context.**ApplicationContext**.
- It is the advanced Spring container and is built on top of the BeanFactory interface.
- ApplicationContext supports the features supported by Bean Factory but also provides some additional functionalities.
- ApplicationContext follows eager-initialization technique which means instance of beans are created as soon as you create the instance of Application context.
- The ClassPathXmlApplicationContext class is the implementation class of ApplicationContext interface. You need to instantiate the ClassPathXmlApplicationContext class to use the ApplicationContext as shown below:

```
ApplicationContext context=new ClassPathXmlApplicationContext("beans.xml");
```

Hope this helps. In case you want to know more about it you can watch this tutorial video:

https://www.youtube.com/watch?v=rMLP-NEPgnM

Lets talk about a bit **lifecylce of Spring Beans**

There are three steps;
1- initialization step
2- Use step
3- Destruction step

In step 1
First of all, Application Context loads bean definition from meta data(XML, AppConfig or via annotation scanning) And then send bean definition to BeanFactoryPostProcessor. For example internalization or Loading constants from resource file is handled by BeanFactoryPostProcessor. Bean definition is updated with the value coming from resources.
In next, Instance is created and then dependencies are injected. In case if there is any init method(**@PostConstruct** annotated method) inside Bean, init method will called and then bean will be ready to use.

In Step 2
This is a use case. Getting bean from IOC Container via ApplicationContext.

In Step 3
Bean objects are removed from container and ready for GC.
@PreDestroy can be used for resource cleaning