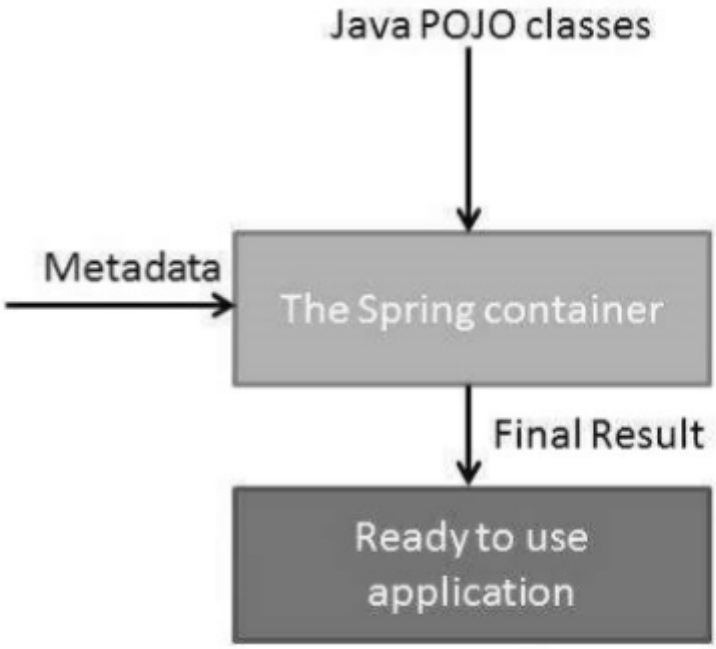
Spring – IoC Containers

The Spring container is at the core of the Spring Framework. The container will create the objects, wire them together, configure them, and manage their complete life cycle from creation till destruction. The Spring container uses DI to manage the components that make up an application. These objects are called Spring Beans.

The container gets its instructions on what objects to instantiate, configure, and assemble by reading the configuration metadata provided. The configuration metadata can be represented either by XML, Java annotations, or Java code. The following diagram represents a high-level view of how Spring works. The Spring IoC container makes use of Java POJO classes and configuration metadata to produce a fully configured and executable system or application.



Spring provides the following two distinct types of containers.

Spring BeanFactory Container
This is the simplest container providing the basic support for DI and is defined by the org.springframework.beans.factory.BeanFactory interface. The BeanFactory and related interfaces, such as BeanFactoryAware, InitializingBean, DisposableBean, are still present in Spring for the purpose of backward compatibility with a large number of third-party frameworks that integrate with Spring.

Spring ApplicationContext Container
This container adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners. This container is defined by the org.springframework.context.ApplicationContext interface.

The ApplicationContext container includes all functionality of the BeanFactory container, so it is generally recommended over BeanFactory. BeanFactory can still be used for lightweight applications like mobile devices or applet-based applications where data volume and speed is significant.

Spring BeanFactory container

This is the simplest container providing the basic support for DI and defined by the org.springframework.beans.factory.BeanFactory interface. The BeanFactory and related interfaces, such as BeanFactoryAware, InitializingBean, DisposableBean, are still present in Spring for the purpose of backward compatibility with a large number of third-party frameworks that integrate with Spring.

There are a number of implementations of the BeanFactory interface that are come straight out-of-the-box with Spring. The most commonly used BeanFactory implementation is the XmlBeanFactory class. This container reads the configuration metadata from an XML file and uses it to create a fully configured system or application.

The BeanFactory is usually preferred where the resources are limited like mobile devices or applet-based applications. Thus, use an ApplicationContext unless you have a good reason for not doing so.

Spring ApplicationContext container

The Application Context is Spring's advanced container. Similar to BeanFactory, it can load bean definitions, wire beans together, and dispense beans upon request. Additionally, it adds more enterprise-specific functionality such as the ability to resolve textual messages from a properties file and the ability to publish application events to interested event listeners. This container is defined by org.springframework.context.ApplicationContext interface.

The ApplicationContext includes all functionality of the BeanFactory, It is generally recommended over BeanFactory. BeanFactory can still be used for lightweight applications like mobile devices or applet-based applications.

The most commonly used ApplicationContext implementations are:

FileSystemXmlApplicationContext − This container loads the definitions of the beans from an XML file. Here you need to provide the full path of the XML bean configuration file to the constructor.

ClassPathXmlApplicationContext − This container loads the definitions of the beans from an XML file. Here you do not need to provide the full path of the XML file but you need to set CLASSPATH properly because this container will look like bean configuration XML file in CLASSPATH.

WebXmlApplicationContext − This container loads the XML file with definitions of all beans from within a web application.