**Spring Container**

Spring container is the heart of Spring. All the magic is done by this container. The container is responsible for creating Objects, Wire the Objects between themselves, manage Objects Life Cycle till creation to destruction. This container defined by ***org.springframework.beans.factory.BeanFactory***

Sometimes we called it IOC container as Dependency injection is done by Container.

To do the magic I mean creating Spring bean, wiring them Spring container needs instruction or Metadata. We can provide metadata in two ways

1. By XML (What we have seen in earlier).

2. By Annotation (We will see it).

**Following diagram will clear the Idea**,

Spring provides two types of Containers

1. Spring BeanFactory.

2. Spring ApplicationContext.

Spring BeanFactory :

The simplest container Spring provides. This container offers minimal support for Dependency Injection, Some Aware interfaces, and initialization and destroys hook, so we can do additional works upon bean initialization and destruction.

It is a very simple container and not suite for J2EE application but where we only need DI feature and some Aware feature we can use it so in Android application we can use it.

Let take an Example How to create our Same HelloWorld using BeanFactory

1. Beans file is same as previous

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<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-beans-3.0.xsd"*>

<bean id=*"helloWorld"* class=*"com.example.hello.HelloWorld"*>

<property name=*"greet"* value=*"Hello World! Welcome to Spring"*/>

</bean>

</beans>

Now we see the How to loads a bean using BeanFactory?

package com.example.hello;

import org.springframework.beans.factory.BeanFactory;

import org.springframework.beans.factory.xml.XmlBeanFactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.core.io.ClassPathResource;

public class HelloWorld {

private String greet;

public String getGreet() {

return greet;

}

public void setGreet(String greet) {

this.greet = greet;

}

public static void main(String[] args) {

BeanFactory ctx = new XmlBeanFactory(new ClassPathResource("configFiles/beans.xml"));

HelloWorld bean =(HelloWorld) ctx.getBean("helloWorld");

System.out.println(bean.getGreet());

}

}

Please note that Here we create a XmlBeanfactory class which is a subclass of BeanFactory and pass an Implementation of Resource interface, ClasspathResource will find the configuration file in the classpath. There is a number of implementations of Resource interface. So, a number of different ways you can search spring configuration file

**Some of main Resource Implementation**

|  |  |
| --- | --- |
| ClassPathResource | ClassPathResource, applying the given path relative to the path of the underlying resource of this descriptor |
| FileSystemResource | FileSystemResource, applying the given path relative to the path of the underlying file of this resource descriptor. |
| UrlResource | UrlResource, applying the given path relative to the path of the underlying URL of this resource descriptor. |

**ApplicationContext :**

The Application Context is an advanced container. This is built on top of Beanfacoty so it can provide Dependency Injection, AOP, Manage Bean life cycle etc. Moreover, it supports JAVA EE application specific functionality like parse and read messages from a properties file the, register spring bean for notifying application state event, publish custom events to event listeners. This container is defined by the ***org.springframework.context.ApplicationContext*** interface.

The ApplicationContext has all features of the BeanFactory, Always use ApplicationContext over the BeanFactory.

**We have seen an Example of Application in the previous Chapter. So I am not repeat it here.**

Some of the important **ApplicationContext** interface implementation

|  |  |
| --- | --- |
| **FileSystemXmlApplicationContext** | Container, Load XML from Your m/c filesystem. |
| **ClassPathXmlApplicationContext** | Container, Load XML from classpath. |
| **WebXmlApplicationContext** | Container, Load XML within web application use for Spring MVC. |