BY NAGOOR BABU

IOC Containers

The main intention of IOC Container is to read bean configurations from configuration file , creating Bean objects and Providing bean Objects to Spring applications.

There are two types of IOC Containers in SPring framework.

1. BeanFactory
2. ApplicationContext

1. BeanFactory

-->It is the fundamental or Base container provided by SPring Framework inorder to manage bean objects.
-->BeanFactory IOC container will provide basic functionalities to the spring framework by creating maintaining beans objects as per the beans configuration details which we provided in spring beans configuration file.

-->To represent BeanFactory IOC Container, SPring framework has provided an interface in the form of
"org.springframework.beans.factory.BeanFactory".

-->For BeanFactory interface, Spring Framework has provided an implementation class in the form of
org.springframework.beans.factory.xml.XmlBeanFactory

-->If we want to use BeanFactory IOC Container in Spring applications then we have to use the following steps.

1) Create Resource Object.
2) Create BeanFactory object
3) Get Bean and access Business Method.

Resource:

-->Resource is an object in SpringFramework, it able to represent all bean configuration details which we provided in beans configurations details.

-->To represent Resource object, Spring Framework has provided a predefined interface in the form of
"org.springframework.core.io.Resource"

-->For Resource interface, Spring Framework has provided the following implementation classes.

1. org.springframework.core.io.ByteArrayResource:

--> It able to represent all the beans configuration details which are available in the form of byte[].

2. org.springframework.core.io.FileSystemResource:

--> It able to get all the beans confguration details which are available in the form of a file in our system

hard disk.

3. org.springframework.core.io.ClassPathResource:

--> It able to get all the beans configuration details which are existed at "classpath" environment variable

refered locations.

4. org.springframework.core.io.InputStreamResource:

--> It able to get all the beans configuration which are existed in the form of InputStream.

5. **org.springframework.core.io.UrlResource:**

--> It able to get all the beans configuration details which are existed at a particular URL in the network.

6. **org.springframework.web.context.support.ServletContextResource:**

--> It able to get all the beans configuration details which are existed in ServletContext.
--> It will be used in spring web applications.

7. **org.apringframework.web.portlet.context.PortletContextResource:**

--> It able to get all the beans configuration details which are existed in PortletContext.
--> It will be used in spring web applications designed on the basis of portlets.

EX:
Resource res=new ClassPathResource("beans.xml");

2) Create BeanFactory Object:

To create XmlBeanFactory class object we have to use the following constructor.
public XmlBeanFactory(Resource res)

EX: BeanFactory factory=new XmlBeanFactrory(res);

3) Get Bean object from BeanFactory and access business method:

To get Bean object from BeanFactory we have to use the following method.
public Object getBean(String id_Name)

EX:
HelloBean bean=(HelloBean)factory.getBean("hello");

Note: BeanFactory is deprecated in Spring3.x version.

Example:

HelloBean.java

```
package com.durgasoft.beans;

public class HelloBean {
	static {
		System.out.println("Bean Loading.....");
	}
	public HelloBean() {
		System.out.println("Bean Created....");
	}
	public String sayHello() {
		return "Hello User";
	}
}
```

applicationContext.xml

```
<?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.xsd">
	<bean id="helloBean" class="com.durgasoft.beans.HelloBean"/>
 </beans>
```

Test.java

```
package com.durgasoft.test;
```

```
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.ClassPathResource;

import com.durgasoft.beans.HelloBean;

public class Test {

	public static void main(String[] args) throws Exception{

		XmlBeanFactory factory = new XmlBeanFactory(new
ClassPathResource("applicationContext.xml"));
		HelloBean bean = (HelloBean)factory.getBean("helloBean");
		System.out.println(bean.sayHello());
	}

}
```

In the above application, when we activate BeanFactory container then BeanFactory Container will be started and it will not create any Bean object immediately and it will perform th following actions.

1. **It will take bean configuration file name and location from Resource object.**
2. **It will search for the respective bean configuration file at the specified location.**
3. **If the respective bean configuration file is available then Beanfactory container will load that xml file to the memory.**
4. **After XML file loading, BeanFactory Container will parse that xml file, that is, it will check all the tags in XML file are provided properly or not, all attributes are available properly or not.**
5. **After the XML file parsing, BeanFactory Container will read data from Beans configuration file and stores in Resource object.**

After getting BeanFactory Object, when we access getBean(-) method then BeanFactory will perform the following actions.

1. BeanFactory will search for the respective Bean configuration in Resource object on the basis of the provided identity.
2. If any bean configuration is identified in Resource object on the basis of the provided identity then BeanFactory container will take the respective bean class name and its location.
3. BeanFactory Container will search for the respective bean class at the specified location, if it is available then BeanFactory Container will load all the bean class bytecode to the memory.
4. BeanFactry Container will create Object for the loaded bean class and its dependent bean objects.
5. BeanFactory Container will store the generated bean object and its dependent objects in Container object in the form of Key-Value pairs, where keys must be the "id" attribute values which we specified in beans configuration file and values are Bean Objects.

ApplicationContext:

ApplicationContext IOC Container is an extension of BeanFactory IOC Container , it able to provide some advanced features like Internationalization, Event Handling,..... along with fundamental functionalities what BeanFactory is providing.

In Spring, ApplicationContext IOC Container is represented in the form of the following predefined interface.
"org.springframework.context.ApplicationContext".

SpringFramework has provided ApplicationContext as a chaild interface to BeanFactory interface.

SpringFramework has provided the following three implementation classes for ApplicationContext.

1.ClassPathXmlApplicationContext:

--> It able to get all the beans configuration details from confguration file which is existed in application classpath.

2.FileSystemXmlApplicationContext:

--> It able to get all the beans configuration details from Configuration file which is existed at our system harddisk.

3.WebXmlApplicationContext:

--> It able to get all the beans configuration details from configuration file which is existed in web application.

Example:

HelloBean.java

```
--------------
package com.durgasoft.beans;

public class HelloBean {
	static {
		System.out.println("Bean Loading.....");
	}
	public HelloBean() {
		System.out.println("Bean Created....");
	}
	public String sayHello() {
		return "Hello User";
	}
}
```

applicationContext.xml

```
<?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.xsd">
         <bean id="helloBean" class="com.durgasoft.beans.HelloBean"/>
 </beans>
```

Test.java

```
package com.durgasoft.test;

import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.ClassPathResource;

import com.durgasoft.beans.HelloBean;

public class Test {

        public static void main(String[] args) throws Exception{

                ApplicationContext context = new
ClasspathXmlApplicationContext("applicationContext.xml");
                HelloBean bean = (HelloBean)context.getBean("helloBean");
                System.out.println(bean.sayHello());
        }

}
```

In the above application, when we activate BeanFactory container then BeanFactory Container will perform the following actions.

1. It will take bean configuration file name and location from Container class constructor.
2. It will search for the respective bean configuration file at the specified location.
3. If the respective bean configuration file is available then ApplicationCntext container will load that xml file to the memory.
4. After XML file loading, ApplicationContext Container will parse that xml file, that is, it will check all the tags in XML file are provided properly or not, all attributes are available properly or not.
5. After the XML file parsing, ApplicationContext Container will read data from Beans configuration file.
6. If any bean configuration is identified in beans configuration file the ApplicationContext container will take beans classes and their locations.
7. ApplicationContext Container will search for the respective beans at the specified locations, if they are available then IOC Container will load all the bean classes bytecode to the memory.
8. ApplicationContext Container will create Objects for the loaded bean classes and their dependent bean objects.
9. ApplicationContext Container will store all the bean objects and their dependent objects in Container object in the form of Key-Value pairs, where keys must be the "id" attribute values which we specified in beans configuration file.

In the above context, if we access getBean("--") method over Container reference then ApplicationContext Container will search for the Bean object on the basis of the provided bean identity , if it is available then AppliationContext Container will return Bean object.

Q) What are the differences between BeanFactory and ApplicationContext IOC Containers?
Ans:
1)BeanFactory is fundamental IOC Container , it able to provide fundamental functionalities to the spring applications like creating and maintaning bean objects.

ApplicationContext IOC Container is an extension of BeanFactory IOC Container , it able to provide some advanced features like Internationalization, Event Handling,..... along with fundamental functionalities what BeanFactory is providing.

2) BeanFactory is not supporting to integrate AOP services like Security, JTA,... to the spring applications.

ApplicationContext is supporting to integrate AOP services like Security, JTA,... to the spring applications.

3)BeanFactory is not suitable for web applications which we are going to prepare on the basis of Spring web module.

ApplicationContext is suitable for the web applications which we want to prepare on the basis of Spring web module.

4)BeanFactory is able to prpare Singleton objects when we send first request for bean, that is, Lazy Instantiation/Initialization.

ApplicationContext is able to prepare Singleton objects when we activate Container , that is , early Instantiation/Initialization.

5.BeanFactory is supporting only the scopes like Singleton and Prototype.

ApplicationContext is supporting almost all the Spring scopes like Singleton, Prototype, request, session, globalSession, webSocket,...

6.BeanFactory is mainly for Standalone Applications.

ApplicationContext is for all the types of Spring famework applications.

7.BeanFactory is an outdated Container in Spring applications.

ApplicationContext is not outdated Container.

BY NAGOOR BABU

CONTACT US:

Mobile: +91- **8885 25 26 27**

+91- **7207 21 24 27/28**

US NUM: **4433326786**

Mail ID: **durgasoftonlinetraining@gmail.com**

WEBSITE: **www.durgasoftonline.com**

FLAT NO: 202, HMDA MYTRIVANUM, AMEERPET, HYDERABAD.,