SPRING APPLICATIONCONTEXT CONTAINER

The Application Context is spring's more 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 the *org.springframework.context.ApplicationContext* interface.

The *ApplicationContext* includes all functionality of the *BeanFactory*, it is generally recommended over the *BeanFactory*. BeanFactory can still be used for light weight 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 bean configuration XML file in CLASSPATH.
- **WebXmlApplicationContext:** This container loads the XML file with definitions of all beans from within a web application.

We already have seen an example on ClassPathXmlApplicationContext container in *Spring Hello World Example*, and we will talk more about XmlWebApplicationContext in a separate chapter when we will discuss web based Spring applications. So let see one example on FileSystemXmlApplicationContext.

Example:

Let us have working Eclipse IDE in place and follow the following steps to create a Spring application:

Step	Description
1	Create a project with a name *SpringExample* and create a package *com.tutorialspoint* under the **src** folder in the created project.
2	Add required Spring libraries using *Add External JARs* option as explained in the *Spring Hello World Example* chapter.
3	Create Java classes *HelloWorld* and *MainApp* under the *com.tutorialspoint* package.
4	Create Beans configuration file *Beans.xml* under the **src** folder.
5	The final step is to create the content of all the Java files and Bean Configuration file and run the application as explained below.

Here is the content of **HelloWorld.java** file:

```
package com.tutorialspoint;

public class HelloWorld {
   private String message;

   public void setMessage(String message){
      this.message  = message;
   }
```

```
   public void getMessage(){
      System.out.println("Your Message : " + message);
   }
}
```

Following is the content of the second file **MainApp.java**:

```
package com.tutorialspoint;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.FileSystemXmlApplicationContext;

public class MainApp {
   public static void main(String[] args) {

      ApplicationContext context = new FileSystemXmlApplicationContext
            ("C:/Users/ZARA/workspace/HelloSpring/src/Beans.xml");

      HelloWorld obj = (HelloWorld) context.getBean("helloWorld");
      obj.getMessage();
   }
}
```

There are following two important points to note about the main program:

- First step is to create factory object where we used framework API **FileSystemXmlApplicationContext** to create the factory bean after loading the bean configuration file from the given path. The **FileSystemXmlApplicationContext** API takes care of creating and initializing all the objects ie. beans mentioned in the XML bean configuration file.
- Second step is used to get required bean using **getBean** method of the created context. This method uses bean ID to return a generic object which finally can be casted to actual object. Once you have object, you can use this object to call any class method.

Following is the content of the bean configuration file **Beans.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-3.0.xsd">

   <bean  >
       <property name="message" value="Hello World!"/>
   </bean>

</beans>
```

Once you are done with creating source and bean configuration files, let us run the application. If everything is fine with your application, this will print the following message:

```
Your Message : Hello World!
```