Spring Example

Here, we are going to learn the simple steps to create the first spring application. To run this application, we are not using any IDE. We are simply using the command prompt. Let's see the simple steps to create the spring application

- **create the class**
- **create the xml file to provide the values**
- **create the test class**
- **Load the spring jar files**
- **Run the test class**

Steps to create spring application

Let's see the 5 steps to create the first spring application.

1) Create Java class

This is the simple java bean class containing the name property only.

```
package com.;

public class Student {
private String name;

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public void displayInfo(){
    System.out.println("Hello: "+name);
}
}
```

This is simple bean class, containing only one property name with its getters and setters method. This class contains one extra method named displayInfo() that prints the student name by the hello message.

2) Create the xml file

In case of myeclipse IDE, you don't need to create the xml file as myeclipse does this for yourselves. Open the applicationContext.xml file, and write the following code:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
```

```
          http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<bean id="studentbean" class="com.javatpoint.Student">
<property name="name" value="Akshay Kumar"></
property>  </bean>

</beans>
```

The **bean** element is used to define the bean for the given class. The **property** subelement of bean specifies the property of the Student class named name. The value specified in the property element will be set in the Student class object by the IOC container.

3) Create the test class

Create the java class e.g. Test. Here we are getting the object of Student class from the IOC container using the getBean() method of BeanFactory. Let's see the code of test class.

```
package com.javatpoint;

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

public class Test {
public static void main(String[] args) {
    XmlBeanFactory factory =new XmlBeanFactory(new ClassPathResource("/applicationContext.xml", Test.class)); );
    BeanFactory factory=new XmlBeanFactory(resource);

    Student student=(Student)factory.getBean("studentbean");
    student.displayInfo();
}
}
```

The **Resource** object represents the information of applicationContext.xml file. The Resource is the interface and the **ClassPathResource** is the implementation class of the Reource interface. The **BeanFactory** is responsible to return the bean. The **XmlBeanFactory** is the implementation class of the BeanFactory. There are many methods in the BeanFactory interface. One method is **getBean()**, which returns the object of the associated class.

4) Load the jar files required for spring framework

There are mainly three jar files required to run this application.

- **org.springframework.core-3.0.1.RELEASE-A**
- **com.springsource.org.apache.commons.logging-1.1.1**
- **org.springframework.beans-3.0.1.RELEASE-A**

For the future use, You can download the required jar files for spring core application.

download the core jar files for spring

download the all jar files for spring including core, web, aop, mvc, j2ee, remoting, oxm, jdbc, orm etc.

To run this example, you need to load only spring core jar files.

5) Run the test class

Now run the Test class. You will get the output Hello: Akshay Kumar .