Creating spring application in Eclipse IDE

Here, we are going to create a simple application of spring framework using eclipse IDE. Let's see the simple steps to create the spring application in Eclipse IDE.

- **create the java project**
- **add spring jar files**
- **create the class**
- **create the xml file to provide the values**
- **create the test class**

Steps to create spring application in Eclipse IDE

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

1) Create the Java Project

Go to **File** menu - **New** - **project** - **Java Project**. Write the project name e.g. firstspring - **Finish**. Now the java project is created.

2) Add spring jar files

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 aop, mvc, j2ee, remoting, oxm, etc.

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

To load the jar files in eclipse IDE, **Right click on your project** - **Build Path** - **Add external archives** - **select all the required jar files** - **finish.**.

3) Create Java class

In such case, we are simply creating the Student class have name property. The name of the student will be provided by the xml file. It is just a simple example not the actual use of spring. We will see the actual use in Dependency Injection chapter. To create the java class, **Right click on src** - **New** - **class** - **Write the class name e.g. Student** - **finish**. Write the following code:

```
package com.javatpoint;

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.

4) Create the xml file

To create the xml file click on src - new - file - give the file name such as applicationContext.xml - finish. 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="Vimal Jaiswal"></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.

5) 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.

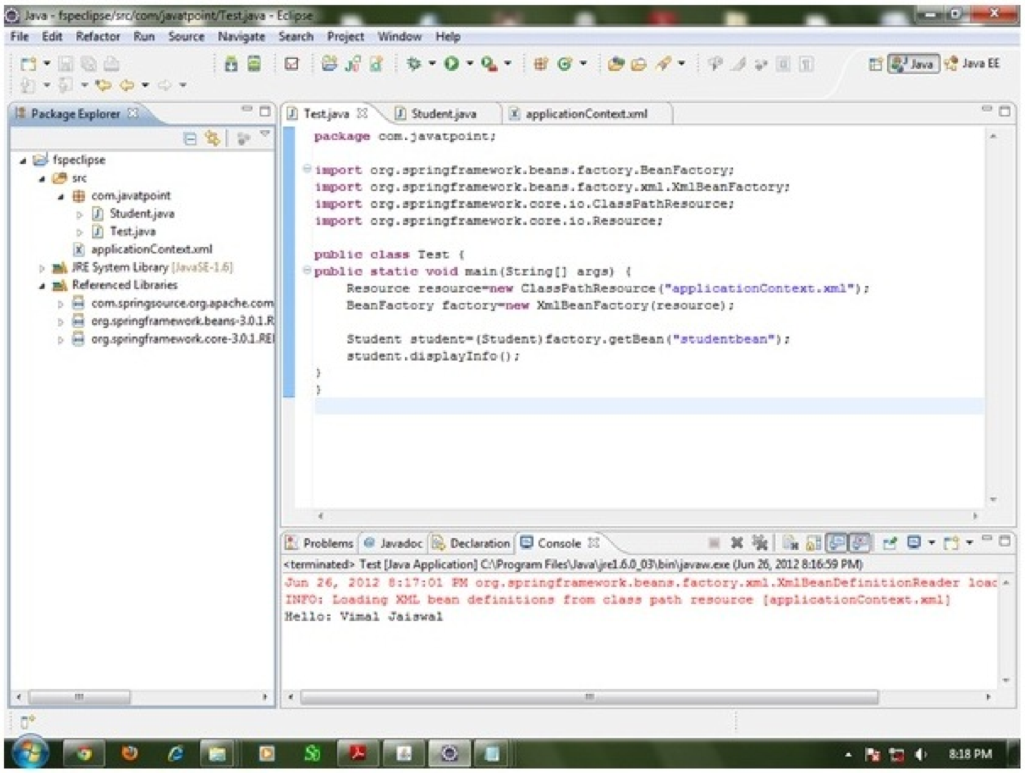
```
package com.javatpoint;

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) {
   Resource resource=new ClassPathResource("applicationContext.xml");
   BeanFactory factory=new XmlBeanFactory(resource);

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

Now run this class. You will get the output Hello: Vimal Jaiswal.



```
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) {
    Resource resource=new ClassPathResource("applicationContext.xml");
    BeanFactory factory=new XmlBeanFactory(resource);

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

```
<terminated> Test [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (Jun 26, 2012 8:16:59 PM)
Jun 26, 2012 8:17:01 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loac
INFO: Loading XML bean definitions from class path resource [applicationContext.xml]
Hello: Vimal Jaiswal
```

download this example

<<prev next>>

⇧

Please Share