VaanNila

Menu

Struts 1

Struts 2

Spring

Hibernate

Ant

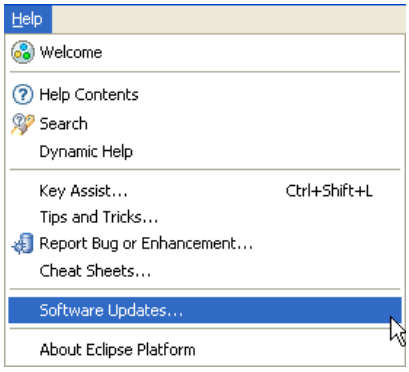
Log4j

Spring > Spring IDE

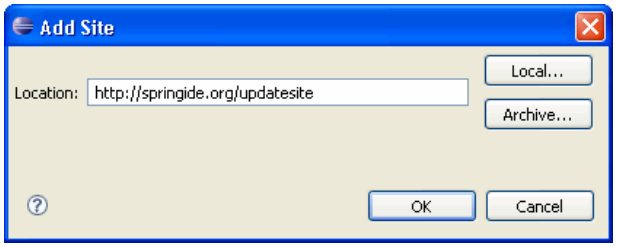
Spring IDE

Spring IDE is an eclipse plug-in that helps in developing Spring Application. First we will see how to install the Spring IDE and later we will create our first Spring project using it. I am using Eclipse 3.4.1 version to demonstrate this.

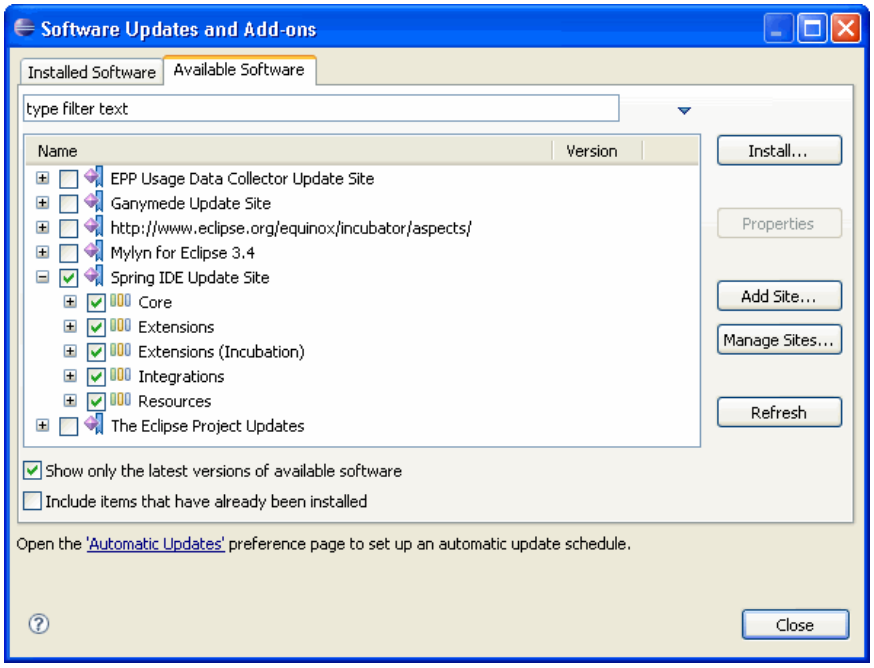
To install Spring IDE, Go to Help -> Software Updates.



Click the "Add Site" button and enter "http://springide.org/updatesite" in the Add Site popup.



Select all the Spring IDE features and click Install.



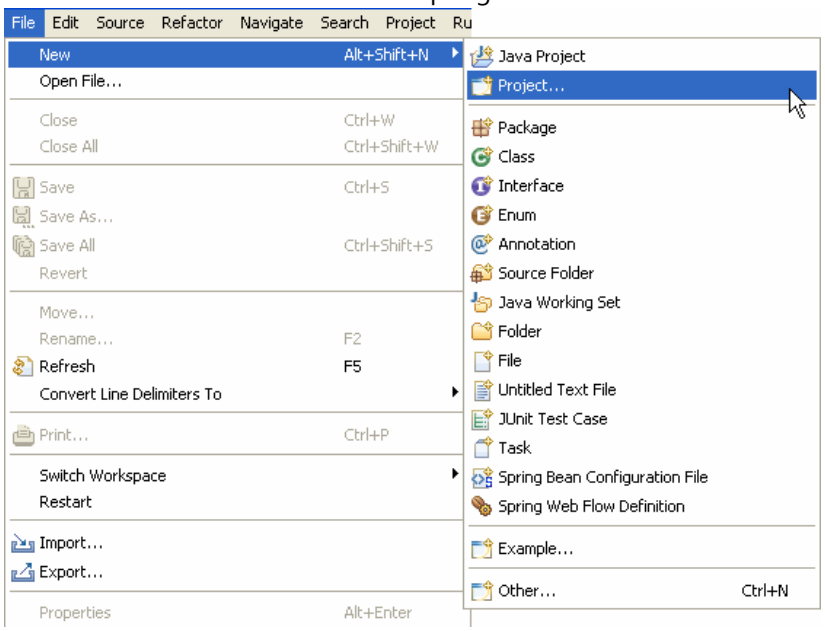
Once the installation is complete you are done. Now let's see how to create the hello world example using the Spring IDE.

First create a Spring project, go to File -> New -> Project.

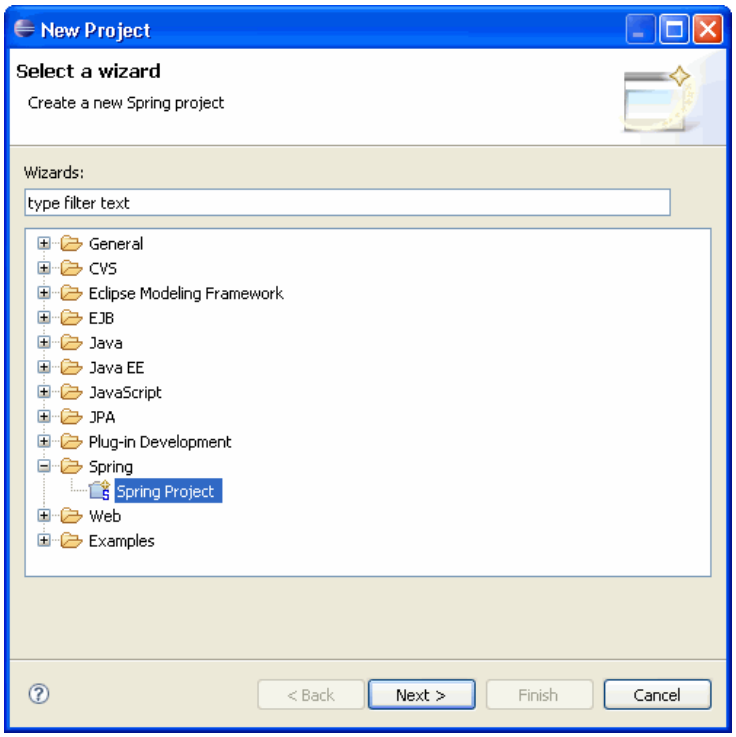
Subscribe

RSS

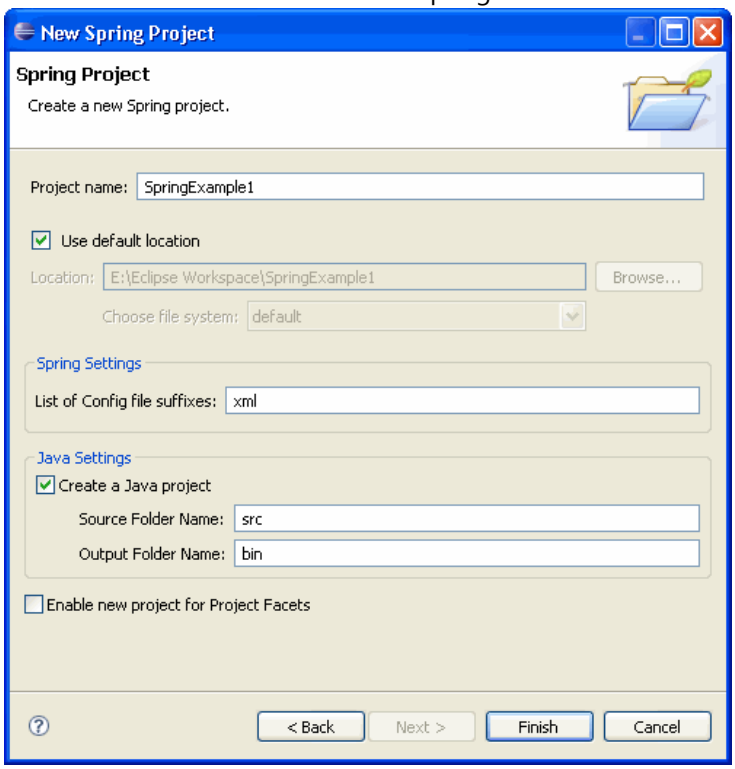
Email



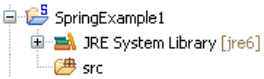
Select Spring Project and click Next.



Enter the project name and click Finish.



The "S" in the upper right corner indictes it is a Spring Project.

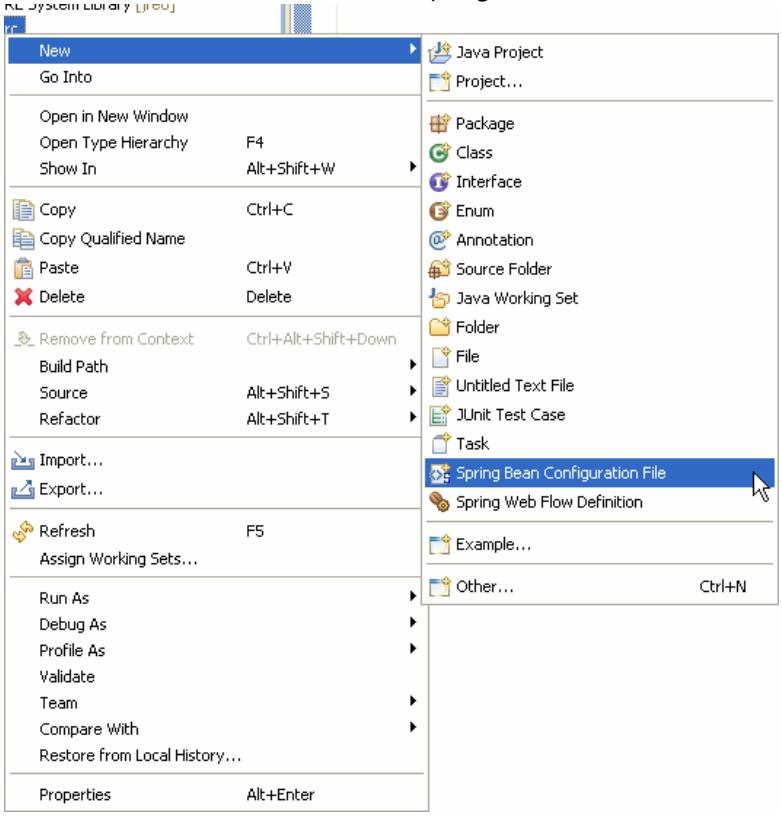


Right click the src package and create a new package "com.vaannila". Create the following HelloWorld class

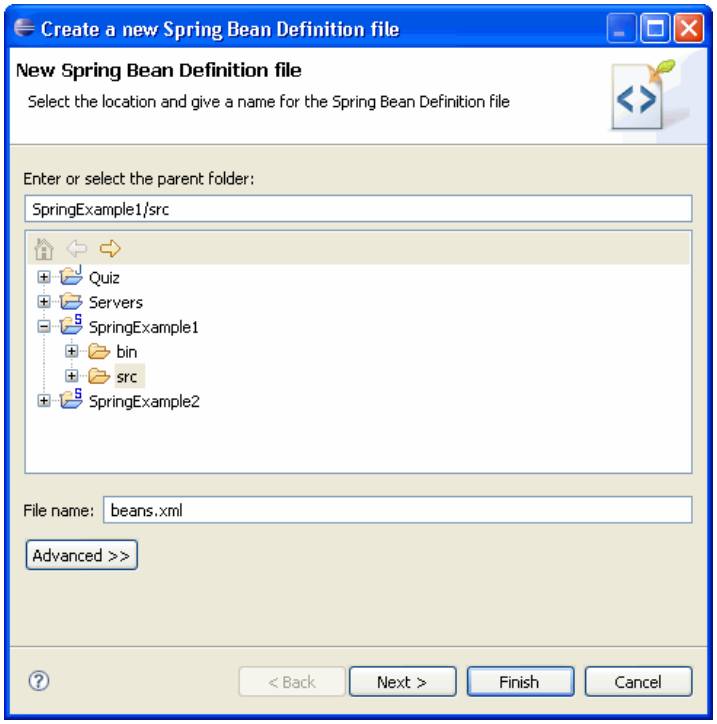
```
01. package com.vaannila;
02.
03. public class HelloWorld {
04.
05.     private String message;
06.
07.     public void setMessage(String message) {
08.         this.message = message;
09.     }
10.
11.     public void display()
12.     {
13.         System.out.println(message);
14.     }
15. }
```

The HelloWorld class has a message property and its value is set using the setMessage() method. This is called setter injection. Instead of directly hard coding the message, we inject it through an external configuration file. The design pattern used here is called Dependency Injection design pattern and it is explained in detail in the next example. (Spring Dependency Injection)The HelloWorld class also has a display() method to display the message.

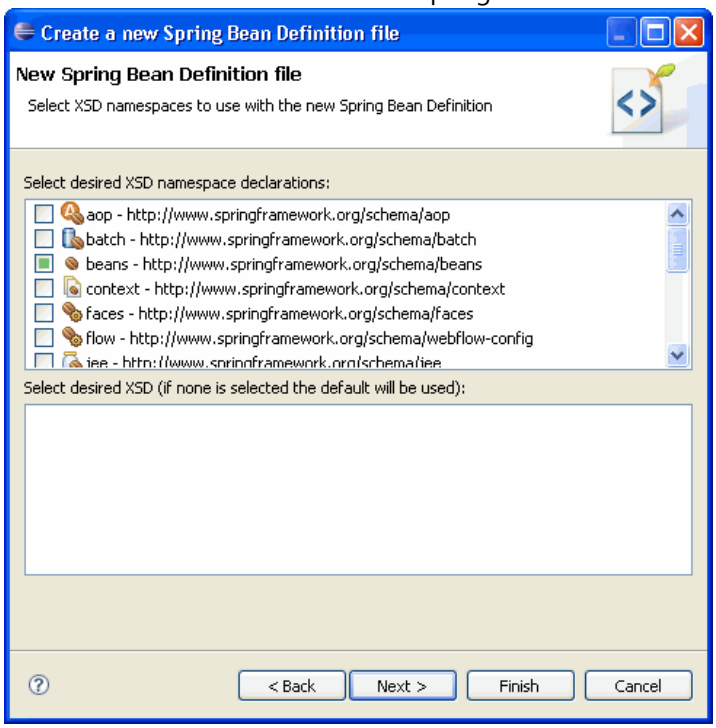
Now we have created the HelloWorld bean class, the next step is to add an entry for this in the bean configuration file. The bean configuration file is used to configure the beans in the Spring IoC container. To create a new bean configuration file right click the src folder and select New -> Spring Bean Configuration File.



Enter the bean name and click Next.



Select the beans option and click Finish.



Now the Spring configuration file is created. Add the following code to created an entry for the `HelloWorld` bean class.

```
01. <?xml version="1.0" encoding="UTF-8"?>
02. <beans xmlns="http://www.springframework.org/schema/beans"
03. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04. xsi:schemaLocation=" http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">
05.
06.     <bean id="helloWorld" class="com.vaannila.HelloWorld">
07.         <property name="message" value="Hello World!"></property>
08.     </bean>
09.
10. </beans>
```

The `id` attribute of the `bean` element is used to give a logical name to the bean and the `class` attribute specifies the fully qualified class name of the bean. The `property` element within the `bean` element is used to set the property value. Here we set the `message` property to "`Hello World!`".

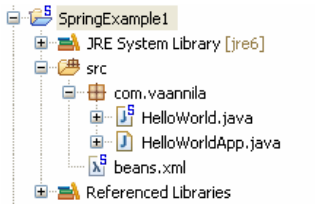
If you want to display a differnt message, the only change you need to is to change the value of the message in the bean configuration file. This is one of the main benefits of using the `Dependency Injection` design pattern, this makes the code loosely coupled.

To dispaly the message, create the following `HelloWorldApp` class.

```
01. package com.vaannila;
02.
03. import org.springframework.context.ApplicationContext;
04. import org.springframework.context.support. ClassPathXmlApplicationContext;
05.
06. public class HelloWorldApp {
07.
08.     public static void main(String[] args) {
09.         ApplicationContext context = new
              ClassPathXmlApplicationContext("beans.xml");
10.         HelloWorld helloWorld = (HelloWorld) context.getBean("helloWorld");
11.         helloWorld.display();
12.     }
13.
14. }
```

First we instantiate the Spring IoC container with the bean configuration file `beans.xml`. We use the `getBean()` method to retrive the `helloWorld` bean from the application context and call the `display()` method to display the message in the console.

The figure shows the final directory structure of the hello world example.



Add the following jar files to the classpath.

```
1. antlr-runtime-3.0
2. commons-logging-1.0.4
```

```
3. org.springframework.asm-3.0.0.M3
4. org.springframework.beans-3.0.0.M3
5. org.springframework.context-3.0.0.M3
6. org.springframework.context.support-3.0.0.M3
7. org.springframework.core-3.0.0.M3
8. org.springframework.expression-3.0.0.M3
```

To execute the example run the `HelloWorldApp` file. The `"Hello World!"` message gets printed on the console.

You can download and try the Spring hello world example by clicking the Download link below.

Source :**Download**

Source + Lib :**Download**