BY NAGOOR BABU

Steps to prepare First Spring Application

1. Download Spring Framework from Internet.
2. Provide Spring Setup in Eclipse IDE
3. Prepare Bean Class
4. Prepare Bean Configuration File
5. Prepare Test / Client Appl.

1. Download Spring Framework from Internet.

a) Download spring jar files in the form of "spring-framework-4.1.6.RELEASE-dist.zip" from the URL "https://repo.spring.io/release/org/springframework/spring/4.1.6.RELEASE/" and "commons-logging-1.2.jar" from apache website.

b) Unzip "spring-framework-4.1.6.RELEASE-dist.zip" at a particular location in our machine[E:\softwares\spring]

2. Provide Spring Setup in Eclipse IDE

a) Open Eclipse IDE.

1. Download eclipse-jee-neon-2-win32.zip file from www.eclipse.org website.
2. Extract ZIP file at "C" drive and get "eclipse" folder.
3. Click on "eclipse" icon at "C:\Eclipse" location
4. Provide Workspace "D:\spring3\core\eclipse".
5. Click on "OK" button.

b) Create Java Project

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.,

BY NAGOOR BABU

1. Right Click on "Project Explorer" Browser.
2. Select "New"
3. Select "Project".
4. Select "Java Project".
5. Click on "Next" button.
6. Provide project name "app1".
7. Click on "Next" button.
8. Click on "Next" button.
9. Click on "Yes" button.

c) Create User defined Library with all Spring jar files and add that User defined Library to "Java project".

1. Right Click on Project [app1].
2. Select "Properties".
3. Select "Java Build path".
4. Select "Libraries".
5. Click on "Add Library" button.
6. Select "User library".
7. Click on "Next" button.
8. Click on "User Libraries" button.
9. Click on "New" button.
10. Provide Library Name "Spring4.1.6_Lib_New".
11. Click on "OK" button.
12. Select "Spring4.1.6_Lib_New" and Click on "Add External Jars" button.
13. Select the required jar files
    commons-logging-1.2.jar
    spring-beans-4.1.6.RELEASE.jar
    spring-core-4.1.6.RELEASE.jar
    spring-context-4.1.6.RELEASE.jar
    spring-context-support-4.1.6.RELEASE.jar
    spring-expression-4.1.6.RELEASE.jar
14. Click on "Open" Button.

15. Click on "OK" button.
16. Click on "Finish" button.
17. Click on "Apply" button and "OK" button.

Note:

Prepare User Defined Library[Spring4.1.6_Lib_New] only one time and use the same for all Spring applications, not required to prepare for each and every application.

3. Prepare Bean Class

The main intention of Bean classes in Spring applications is to manage properties and their setXXX() and getXXX() methods and some other Business Methods.

To prepare Bean classes in Spring applications we have to use the following Guidlines.

a) Bean classes must be POJO classes, they must not extend or implement any predefined Library excepr java.io.Serializable marker interface.

b) Bean must be declared as "public" , "Non-abstract" and "non-final".

---->The main intention of declaring bean class as "public" is to make available bean class scope to IOC Container inorder to create objects.

---->The main intention to declare bean class as "Non-abstract" is to allow to create object .

---->The main intention to declare bean classes as "Non-final" is to extend one bean class to another bean class inorder to improve reusability.

c) In Bean classes, we have to declare all properties as "private" and all behaviours as "public", it will improve "Encapsulation".

d) If we want to provide any constructor in bean class then provide a constructor , it must be 0-arg constructor and "public" constructor, because, IOC Container will search and execute public and 0-arg constructor while instantiating bean.

To create bean class in Eclipse IDE, we have to use the following steps.

1. Right Click on "src" flder.

2. Select "New".
3. Select "Class".
4. Provide package name and class name.
package name: com.durgasoft.beans
class name: HelloBean
5. Click on "Finish" button.
6. As per the requirement provide properties and setXXX() and getXXX() with the following steps.
a) Declare variables manually.
b) Right Click and Select "Source".
c) Select "Generate Getters and Setters".
d) select "Select All" button.
e) Click on "OK" button.
7. Provide Business Methods as per the requirement.

EX: HelloBean.java

```
---------------------
package com.durgasoft.beans;
public class HelloBean {
 public String sayHello(){
         return "Hello User!";
 }
}
```

4 Prepare Bean Configuration File

The main intention of Bean COnfiguration File is to provide all the bean components configuration details like logical name, fully qualified names of the bean classes, bean classes properties and bean classes dependencies,..... to the IOCcontainer inorder to create bean objects and tjheir dependent objects.

Bean Configuration File is an XML file, it will use the following XML tags to configure bean classes.

<beans ----XSD----- >


```
 -----
 <bean id="--" class="---">
   -----
 </bean>
 -----
</beans>
```

Where "<beans>" tag is a root tag, it will include no of beans configurations.

Where "<bean>" tag can be used to configure single bean class.

Where "id" attribute will take identity name or logical name to the Bean component.

Where "name" attribute will take fully qualified name of the bean class.

To prepare bean confguration file in Eclipse IDE we have to use the following steps.

1) Create a package under "src" folder.

a) Right Click on "src".
b) Select "New".
c) select "package".
d) Provide package name "com.durgasoft.cfgs".
e) Click on "Finish" button.

2) Create XML file under package.

a) Right Click on "com.durgasoft.cfgs" package.
b) Select "New".
c) Select "Others".
d) Select XML and "XML File".
e) Click on "Next" button.
f) Provide file name "spring_beans_config.xml".
g) Click on "Next" button.

h) Click on "Next" button.
i) Click on "Finish" button.

3)Provide spring XSD in configuration File.

Open bean.html file available at the location "E:\softwares\spring\spring-framework-4.1.6.RELEASE\docs\spring-framework-reference\html" copy XSD from any example and past into XML file.
Provide beans configurations as per the requirement.
EX:spring_beans_config.xml

```
----------------------------
<?xml version="1.0" encoding="UTF-8"?>
<beans --- XSD ---->
  <bean id="hello" class="com.durgasoft.beans.HelloBean"/>
</beans>
```

5. Prepare Test / Client Appl:

The main intention of Test / Client application is to activate IOC Container , to create Bean Components and to access business methods.

To prepare Test application in Eclipse IDE we have to use the following steps.
a) Right Click on "SRC".
b) Select "New".
c) Select "Class".
d) Provide the following details.
  package name: com.durgasoft.test
  Class Name : Test
  Select "public static void main(String[] args)
e) Click on "Finish" button.

Provide Application logic in main() method with the following steps.
1) Activate ApplicationContext IOC Container.
2) Get Bean Object from ApplicationContext.

3) Access Business Method.

EX: Test.java

```
package com.durgasoft.test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.durgasoft.beans.HelloBean;
public class Test {
public static void main(String[] args)throws Exception {
ApplicationContext context=new ClassPathXmlApplicationContext
("/com/durgasoft/cfgs/spring_beans_config.xml");
HelloBean bean=(HelloBean)   context.getBean("hello");
System.out.println(bean.sayHello());
}
}
```

If we provide properties and their respective setXXX() and getXXX() methods in bean classes then we have to send values to bean properties in the following two ways.

1) From Test Program
2) From Beans configuration File.

1) From Test Program

In this approach, we have to set value to bean properties programatically from Test application.

Example:
HelloBean.java


```
package com.durgasoft.beans;

public class HelloBean {
	private String name;

	public String getName() {
		return name;
	}

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

	public String sayHello() {
		return "Hello "+name;
	}
}
```

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 name="helloBean" class="com.durgasoft.beans.HelloBean">
        </bean>
</beans>
```

Test.java

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

import org.springframework.context.ApplicationContext;
```


```
import org.springframework.context.support.ClassPathXmlApplicationContext;

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");
                bean.setName("Durga");
                System.out.println(bean.sayHello());
        }
}
```

OP: Hello Durga!

In the above approach, we have to recompile the Test application when we change messages, it is not suggestible in application development. To overcome this problem we have to use beans configuration file to prepare messages inorder to send to Bean object.

2. Beans Configuration File:

To provide messages to the bean properties through their setXXX() methods from bean configuration file we have to use the following tag in beans configuration file.

```
<beans ---XSD----- >
 ----
 <bean id="--" class="--">
   <property name="--" value="--"/>
   ----
 </bean>
 ----
</beans>
```

Where <property> tag is able to represent single bean property.
Where "name" attribute in <property> tag is able to take property name to which we want to send data.
Where "value" attribute in <property> tag is able to take the value which we want to send to the bean property.

app2:

HelloBean.java

```
package com.durgasoft.beans;

public class HelloBean {
	private String name;

	public String getName() {
		return name;
	}

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

	public String sayHello() {
		return "Hello "+name;
	}
}
```

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 name="helloBean" class="com.durgasoft.beans.HelloBean">
                <property name="name" value="Durga"/>
        </bean>
</beans>

Test.java
----------
package com.durgasoft.test;

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

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());

        }

}

OP: Hello Durga

Steps to prepare Spring Application[Core Module] in Netbeans IDE:

 1. Install Netbeans IDE
 2. Create Java Project:
 3. Add Spring library to Project Library
```

4. Prepare Packages under "Source Packages"
5. Prepare Bean classes as per the requirement
6. Prepare COnfiguration File
7. Create Test class.
8. Run Test class

1.Installation Process:

1. Download netbeans-8.2-windows.exe file from internet.
2. Double click on netbeans-8.2-windows.exe
3. Click on "Yes" button.
4. Click on "Next" button.
5. Select "Checkbox" like "I Accept the terms in licence aggrement".
6. Click on "Next" button.
7. Change Netbeans installation location from C:\Program Files\NetBeans 8.2 to C:\NetBeans 8.2
8. Click on "Next" button.
9. Change Glassfish server installation location from "C:\Program Files \glassfish-4.1.1" to "C:\glassfish-4.1.1" .
10. Click on "Next" button.
11. Click on "Install" button.
12. Click on "Finish".
13. Double Click on "Netbeans" IDE icon on our desktop.

2.Create Java Project:

a) Right Click on "Projects" Browser.
b) Select "New Project".
c) Select "Java" under "Catagories".
d) Select "Java Application" under projects.
e) Click on "Next" button.
f) Change application[app21]
g) Change Project Location[D:\spring3\core\netbeans].

h) Provide Main class Name[com.durgasoft.test.Test].
i) Click on "Finish" button.

3. Add Spring library to Project Library:

a) Right Click on "Libraries" folder in Project.
b) Select "Add Library".
c) Select "Spring Framewwork 4.0.1" Library.
d) Click on "Add Library" button.

4.Prepare Packages under "Source Packages":

a) Right Click on "Source Package".
b) Select "New".
c) Select "Java Package".
d) Provide package Name[com.durgasoft.beans].
e) Click on "Finish" button.

Note: Similarily prepare packagfe com.durgasoft.cfgs and com.durgasoft.test.

4. Prepare Bean classes as per the requirement.

a) Right Click on "com.durgasoft.beans" package.
b) Select "New"
c) Select "Java Class".
d) Provide class Name[User].
e) Click on "Fininsh".
f) In the generated class declare variables.
g) Generate setXXX() and getXXX() methods .
1) Right Click .
2) Select "INsert Code".
3) Select "Getters and Setters".
4) Select "User" check box.
5) Click on "Generate".

h) Provide other methods also as per the requirement.

User.java

```
package com.durgasoft.beans;
public class User {
   private String uname;
   private String uqual;
   private String uage;
   private String uaddr;
   private String uemail;
   private String umobile;

        setXXX()
        getXXX()

   public void display_User_Details(){
      System.out.println("User Details");
      System.out.println("--------------------");
      System.out.println("User Name         :"+uname);
      System.out.println("User Qualification  :"+uqual);
      System.out.println("User Age          :"+uage);
      System.out.println("User Address       :"+uaddr);
      System.out.println("User Email        :"+uemail);
      System.out.println("User Mobile       :"+umobile);
   }

}
```

6) Prepare COnfiguration File:

a) Right Click on "com.durgasoft.cfgs" under Source Packages.
b) Click on "New".

c) Select "Other".
d) Select "Other" under Catagories.
e) Select "SpringXMLConfig(-)" under Types.
f) Click on "Next" button.
g) Provide file name[spring_beans_config.xml].
h) Click on "Next" button.
i) Select required "Namespaces" .
j) Click on "Finish" button.
k) Provide beans configuration in the generated xml file.

spring_beans_config.xml

```
<beans>
   <bean id="user" class="com.durgasoft.beans.User">
      <property name="uname" value="Durga"/>
      <property name="uqual" value="MTech"/>
      <property name="uage" value="28"/>
      <property name="uaddr" value="Hyd"/>
      <property name="uemail" value="durga@durgasoft.com"/>
      <property name="umobile" value="91-9988776655"/>
   </bean>
</beans>
```

7) Create Test class:

Test class already created at the time of creating project, where provide application logic.

Test.java

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

BY NAGOOR BABU

```
import com.durgasoft.beans.User;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class Test {
public static void main(String[] args)throws Exception {
ApplicationContext context=new
ClassPathXmlApplicationContext("/com/durgasoft/cfgs/spring_beans_config.xml");
User user=(User)context.getBean("user");
user.display_User_Details();
}
}
```


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.,