**Spring Setter Injection Tutorial**

Configure the bean properties via setter injection.

In this example you will learn how to set a bean property via setter injection. Consider the following User bean class.

01.package com.tutorials4u;

02.

03.public class User {

04.

05.    private String name;

06.    private int age;

07.    private String country;

08.

09.    public String getName() {

10.        return name;

11.    }

12.

13.    public void setName(String name) {

14.        this.name = name;

15.    }

16.

17.    public int getAge() {

18.        return age;

19.    }

20.

21.    public void setAge(int age) {

22.        this.age = age;

23.    }

24.

25.    public String getCountry() {

26.        return country;

27.    }

28.

29.    public void setCountry(String country) {

30.        this.country = country;

31.    }

32.

33.    public String toString() {

34.        return name + " is " + age + " years old, living in " + country;

35.    }

36.}

The User bean class has three attributes viz. name, age and country. All the three attributes are set using the setter injection. The toString() method of the User bean class is overridden to display the user object.

Here the beans.xml file is used to do spring bean configuration. The following code shows how to set a property value thru setter injection.

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="user" class="com.tutorials4u.User" >

07.        <property name="name" value="Vijay" />

08.

09.        <property name="age" value="27"/>

10.        <property name="country" value="India"/>

11.    </bean>

12.

13.</beans>

The id attribute of the bean element is used to specify the bean name and the class attribute is used to specify the fully qualified class name of the bean. The property element with in the bean element is used to inject property value via setter injection. The name attribute of the property element represents the bean attribute and the value attribute specifies the corresponding property value.

Here we set "Vijay", "27" and "India" for the User bean properties name, age and country respectively. The following Main class is used to get the User bean from the Spring IoC container and dispaly its value it to the user.

01.package com.tutorials4u;

02.

03.import org.springframework.context.ApplicationContext;

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

05.

06.public class Main {

07.

08.    public static void main(String[] args) {

09.        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");

10.        User user = (User)context.getBean("user");

11.        System.out.println(user);

12.    }

13.

14.}

On executing the Main class the following message gets displayed on the console.

1.Vijay is 27 years old, living in India

**Spring Constructor Injection Tutorial**

Configure the bean properties via constructor injection.

In this example you will learn how to set a bean property via constructor injection. Consider the following User bean class.

01.package com.tutorials4u;

02.

03.    public class User {

04.

05.    private String name;

06.    private int age;

07.    private String country;

08.

09.    User(String name, int age, String country)

10.    {

11.        this.name=name;

12.        this.age=age;

13.        this.country=country;

14.    }

15.

16.    public String toString() {

17.        return name + " is " + age + " years old, living in " + country;

18.    }

19.}

The User bean class has three attributes viz. name, age and country. All the three attributes are set thru constructor injection. The toString() method of the User bean class is overridden to display the user object.

Here the beans.xml file is used to do spring bean configuration. The following code shows how to set a property value thru constructor injection.

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="user" class="com.tutorials4u.User" >

07.        <constructor-arg value="Vijay" />

08.

09.        <constructor-arg value="27"/>

10.        <constructor-arg value="India"/>

11.    </bean>

12.

13.</beans>

The constructor-arg element within the bean element is used to set the property value thru constructor injection. Since there is only one constructor in the User bean class, this code will work fine. When there is more than one constructor with the same number of arguments, then the following ambiguities will occur. Conside the following code.

01.User( int age, String country)

02.{

03.    this.age=age;

04.    this.country=country;

05.}

06.

07.User(String name, String country)

08.{

09.    this.name=name;

10.    this.country=country;

11.}

The bean configuration file.

1.<bean id="user" class="com.tutorials4u.User" >

2.    <constructor-arg value="27"/>

3.    <constructor-arg value="India"/>

4.</bean>

Now which constructor do you think will be invoked? The first one with the int and the String argument, right? But for your surprise it will call the second constructor with both String arguments. Though we know the first argument is of type int and the second argument is of type String, spring interprets both as String arguments. To avoid this confusion you need to specify the type attribute of the constructor-arg element. Now with the following bean configuration, the first constructor will be invoked.

1.<bean id="user" class="com.tutorials4u.User" >

2.    <constructor-arg type="int" value="27"/>

3.    <constructor-arg type="java.lang.String" value="India"/>

4.

5.</bean>

Now consider this case. We have the following constructors in the User bean class.

01.User(String name, int age)

02.{

03.    this.name=name;

04.    this.age=age;

05.}

06.

07.User( int age, String country)

08.{

09.    this.age=age;

10.    this.country=country;

11.}

The bean configuration file.

1.<bean id="user" class="com.tutorials4u.User" >

2.    <constructor-arg type="int" value="27"/>

3.    <constructor-arg type="java.lang.String" value="India"/>

4.</bean>

Now which constructor do you think will be called? The second constructor, right? But again for your surprise the first constructor will be called, this is because the order in which the arguments appear in the bean configuration file will not be considered while invoking the constructor. To solve this problem you can use the index attribute to specify the constructor argument index.

Here is the bean configuration file after adding the index attribute.

1.<bean id="user" class="com.tutorials4u.User" >

2.    <constructor-arg index="0" type="int" value="27"/>

3.    <constructor-arg index="1" type="java.lang.String" value="India"/>

4.</bean>