

SPRING AUTOWIRING BY CONSTRUCTOR

http://www.tutorialspoint.com/spring/spring_autowiring_byconstructor.htm

Copyright © tutorialspoint.com

This mode is very similar to *byType*, but it applies to constructor arguments. Spring container looks at the beans on which *autowire* attribute is set to *constructor* in the XML configuration file. It then tries to match and wire its constructor's argument with exactly one of the beans name in configuration file. If matches are found, it will inject those beans otherwise, it will throw exceptions.

For example, if a bean definition is set to autowire by *constructor* in configuration file, and it has a constructor with one of the arguments of *SpellChecker* type, Spring looks for a bean definition named *SpellChecker*, and uses it to set the constructor's argument. Still you can wire remaining arguments using `<constructor-arg>` tags. Following example will illustrate the concept.

Let us have working Eclipse IDE in place and follow the following steps to create a Spring application:

Step	Description
1	Create a project with a name <i>SpringExample</i> and create a package <i>com.tutorialspoint</i> under the src folder in the created project.
2	Add required Spring libraries using <i>Add External JARs</i> option as explained in the <i>Spring Hello World Example</i> chapter.
3	Create Java classes <i>TextEditor</i> , <i>SpellChecker</i> and <i>MainApp</i> under the <i>com.tutorialspoint</i> package.
4	Create Beans configuration file <i>Beans.xml</i> under the src folder.
5	The final step is to create the content of all the Java files and Bean Configuration file and run the application as explained below.

Here is the content of **TextEditor.java** file:

```
package com.tutorialspoint;

public class TextEditor {
    private SpellChecker spellChecker;
    private String name;

    public TextEditor( SpellChecker spellChecker, String name ) {
        this.spellChecker = spellChecker;
        this.name = name;
    }

    public SpellChecker getSpellChecker() {
        return spellChecker;
    }

    public String getName() {
        return name;
    }

    public void spellCheck() {
        spellChecker.checkSpelling();
    }
}
```

Following is the content of another dependent class file **SpellChecker.java**:

```
package com.tutorialspoint;

public class SpellChecker {
```

```

public SpellChecker(){
    System.out.println("Inside SpellChecker constructor." );
}

public void checkSpelling()
{
    System.out.println("Inside checkSpelling." );
}
}

```

Following is the content of the **MainApp.java** file:

```

package com.tutorialspoint;

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

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context =
            new ClassPathXmlApplicationContext("Beans.xml");

        TextEditor te = (TextEditor) context.getBean("textEditor");

        te.spellCheck();
    }
}

```

Following is the configuration file **Beans.xml** in normal condition:

```

<?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-3.0.xsd">

    <!-- Definition for textEditor bean -->
    <bean >
        <constructor-arg ref="spellChecker" />
        <constructor-arg value="Generic Text Editor"/>
    </bean>

    <!-- Definition for spellChecker bean -->
    <bean >
    </bean>

</beans>

```

But if you are going to use autowiring 'by constructor', then your XML configuration file will become as follows:

```

<?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-3.0.xsd">

    <!-- Definition for textEditor bean -->
    <bean
        autowire="constructor">
        <constructor-arg value="Generic Text Editor"/>
    </bean>

    <!-- Definition for spellChecker bean -->

```

```
<bean >  
</bean>  
  
</beans>
```

Once you are done with creating source and bean configuration files, let us run the application. If everything is fine with your application, this will print the following message:

```
Inside SpellChecker constructor.  
Inside checkSpelling.
```