

Spring 由构造函数自动装配

Spring 由构造函数自动装配

这种模式与 *byType* 非常相似，但它应用于构造器参数。Spring 容器看作 beans，在 XML 配置文件中 beans 的 *autowire* 属性设置为 *constructor*。然后，它尝试把它的构造函数的参数与配置文件中 beans 名称中的一个进行匹配和连线。如果找到匹配项，它会注入这些 bean，否则，它会抛出异常。

例如，在配置文件中，如果一个 bean 定义设置为通过构造函数自动装配，而且它有一个带有 *SpellChecker* 类型的参数之一的构造函数，那么 Spring 就会查找定义名为 *SpellChecker* 的 bean，并用它来设置构造函数的参数。你仍然可以使用 `<constructor-arg>` 标签连接其余属性。下面的例子将说明这个概念。

让我们在恰当的位置使用 Eclipse IDE，然后按照下面的步骤来创建一个 Spring 应用程序：

步骤	描述
1	创建一个名称为 <i>SpringExample</i> 的项目，并且在已创建的项目的 <code>src</code> 文件夹包 <i>com.tutorialspoint</i> 。
2	使用 <i>Add External JARs</i> 选项，添加所需的 Spring 库，在 <i>Spring Hello Wo</i>
3	在 <i>com.tutorialspoint</i> 包中创建 Java 类 <i>TextEditor</i> , <i>SpellChecker</i> 和
4	在 <code>src</code> 文件夹中创建 Beans 的配置文件 <i>Beans.xml</i> 。
5	最后一步是创建所有 Java 文件和 Bean 配置文件的内容，并运行该应用程序，正

这里是 **TextEditor.java** 文件的内容：

```
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();  
}  
}
```

下面是另一个依赖类文件 **SpellChecker.java** 的内容：

```
package com.tutorialspoint;  
public class SpellChecker {  
    public SpellChecker(){  
        System.out.println("Inside SpellChecker constructor." );  
    }  
    public void checkSpelling()  
    {  
        System.out.println("Inside checkSpelling." );  
    }  
}
```

下面是 **MainApp.java** 文件的内容：

```
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();  
    }  
}
```

下面是在正常情况下的配置文件 **Beans.xml** 文件：

```
<?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 id="textEditor" class="com.tutorialspoint.TextEditor">
    <constructor-arg ref="spellChecker" />
    <constructor-arg value="Generic Text Editor"/>
</bean>

<!-- Definition for spellChecker bean -->
<bean id="spellChecker" class="com.tutorialspoint.SpellChecker">
</bean>

</beans>
```

但是，如果你要使用自动装配 “by constructor” ，那么你的 XML 配置文件将成为如下：

```
<?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 id="textEditor" class="com.tutorialspoint.TextEditor"
        autowire="constructor">
        <constructor-arg value="Generic Text Editor"/>
    </bean>

    <!-- Definition for spellChecker bean -->
    <bean id="SpellChecker" class="com.tutorialspoint.SpellChecker">
    </bean>

</beans>
```

一旦你完成了创建源代码和 bean 的配置文件，我们就可以运行该应用程序。如果你的应用程序一切都正常，它将打印下面的消息：

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

