Spring 自动装配 byType

Spring 自动装配 byType

这种模式由属性类型指定自动装配。Spring 容器看作 beans，在 XML 配置文件中 beans 的 *autowire* 属性设置为 *byType*。然后，如果它的 **type** 恰好与配置文件中 beans 名称中的一个相匹配，它将尝试匹配和连接它的属性。如果找到匹配项，它将注入这些 beans，否则，它将抛出异常。

例如，在配置文件中，如果一个 bean 定义设置为自动装配 *byType*，并且它包含 *SpellChecker* 类型的 *spellChecker* 属性，那么 Spring 就会查找定义名为 *SpellChecker* 的 bean，并且用它来设置这个属性。你仍然可以使用 <property> 标签连接其余属性。下面的例子将说明这个概念，你会发现和上面的例子没有什么区别，除了 XML 配置文件已经被改变。

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 *SpringExample* 的项目，并且在已创建的项目的 src 文件夹包 *com.tutorialspoint*。
2	使用 *Add External JARs* 选项，添加所需的 Spring 库，在 *Spring Hello Wo*
3	在 *com.tutorialspoint* 包中创建 Java 类 *TextEditor*，*SpellChecker* 和
4	在 src 文件夹中创建 Beans 的配置文件 *Beans.xml*。
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 void setSpellChecker( SpellChecker spellChecker ) {
      this.spellChecker = spellChecker;
```

```
   }
   public SpellChecker getSpellChecker() {
      return spellChecker;
   }
   public void setName(String name) {
      this.name = name;
   }
   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">
      <property name="spellChecker" ref="spellChecker" />
      <property name="name" 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 “byType” ，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="byType">
      <property name="name" 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.
```