依赖注入（或有时称为布线）有助于把这些类粘合在一起，同时保持他们独立。

```
public class TextEditor {
   private SpellChecker spellChecker;
   public TextEditor() {
      spellChecker = new SpellChecker();
   }
}
```

```
public class TextEditor {
   private SpellChecker spellChecker;
   public TextEditor(SpellChecker spellChecker) {
      this.spellChecker = spellChecker;
   }
}
```

从 TextEditor 中删除了全面控制，并且把它保存到其他地方（即 XML 配置文件），且依赖关系（即 SpellChecker 类）通过**类构造函数**被注入到 TextEditor 类中。因此，控制流通过依赖注入（DI）已经“反转”，因为你已经有效地委托依赖关系到一些外部系统。

序号	依赖注入类型 & 描述
1	Constructor-based dependency injection 当容器调用带有多个参数的构造函数类时，实现基于构造函数的 DI，每个代表在其他类中的一个依赖关系。
2	Setter-based dependency injection 基于 setter 方法的 DI 是通过在调用无参数的构造函数或无参数的静态工厂方法实例化 bean 之后容器调用 beans 的 setter 方法来实现的。

ioc中定义bean前提：必须至少提供无参构造。

依赖注入三种方式：（ref指定传递的是对象类型，是对另一个bean的引用）

1.setter方式的依赖注入。

基于设值函数的依赖注入：当容器调用一个无参的构造函数或一个无参的静态 factory 方法来初始化你的 bean 后，通过容器在你的 bean

上调用设值函数，基于设值函数的 DI 就完成了。

<property name=" " value=" "></property> 或

<property name=" ">

<value></value>

</property>

ApplicationContext.xml配置Bean的property属性。底层是反射实现拿到该类的setter方法。

ps.赋值null：<property name=" "></null></property>没有<value>

2.构造器注入：通过构造方法赋值。（多个构造方法类似时调用首个。）

基于构造函数的依赖注入：当容器调用带有一组参数的类构造函数时，基于构造函数的 DI 就完成了。每个参数代表一个对其他类的依赖。

<constructor-arg value=" " (name/type/index)>

</constructor-arg>

Type属性指定构造函数参数类型。index指定索引。

3.p命名空间注入

基于设值函数的依赖注入。先引入命名空间：

<beans xmlns:p="http://www.springframework.org/schema/p">

直接在bean上赋值：

<bean name=" " class=" " p:name01=" " p:name02-ref=" ">

<property>属性注入的value与<value>注入方式的区别：

	<value>子元素注入	value属性注入
参数值位置	首尾标签中间，不加双引号（ “” ）	写在属性中，必须加双引号("")
type属性	可选，用来指定参数类型	无

参数值包含特殊值（ < ，& ）的处理方法	1.使用<![cDATA[]]>标记 eg.<![cDATA[<]]> 2.XMl预定义的实体引用	XML预定义的实体引用

（XML预定义的实体引用XML语法与元素.note）