控制反转（IOC，Inversion of Control），控制反转就是对对象控制权的转移，从程序代码本身反转到了外部容器。

IoC是一个概念，是一种思想，其实现方式多种多样。当前比较流行的实现方式有两种：依赖注入和依赖查找。依赖注入方式应用更为广泛。

* **依赖查找**：Dependency Lookup，DL，容器提供回调接口和上下文环境给组件，程序代码则需要提供具体的查找方式。比较典型的是依赖于JNDI系统的查找。
* **依赖注入**：Dependency Injection，DI，程序代码不做定位查询，这些工作由容器自行完成。

依赖注入DI是指程序运行过程中，若需要调用另一个对象协助时，无须在代码中创建被调用者，而是依赖于外部容器，由外部容器创建后传递给程序。

Spring的依赖注入对调用者与被调用者几乎没有任何要求，完全支持POJO之间依赖关系的管理。

依赖注入是目前最优秀的解耦方式。依赖注入让Spring的Bean之间以配置文件的方式组织在一起，而不是以硬编码的方式耦合在一起的。

###### Spring程序开发

###### 导入Jar包

###### 定义接口与实体类

###### 创建Spring配置文件

applicationContext.xml

###### 定义测试类

* ApplicationContext接口容器
* BeanFactory接口容器

两个接口容器的区别：

1. ApplicationContext容器中对象的装配时机
2. BeanFactory容器中对象的装配时机

ApplicationContext与BeanFactory容器的区别；

1)ApplicationContext容器在进行初始化时，会将其中的所有Bean(对象)进行创建

缺点：占用系统资源（内存、CPU等）

优点：响应速度快

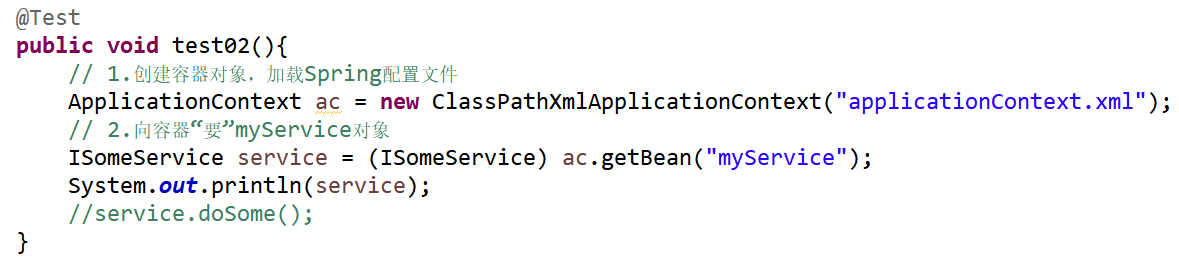
2)BeanFactory容器中的对象，在容器初始化时并不会被创建，而是在真正获取该对象时才被创建

缺点：相对来说响应速度慢

优点：不多占用系统资源



尽量使用name属性



IOC:控制反转,将对象的创建权交给了Spring.

DI:Dependency Injection依赖注入.需要有IOC的环境,Spring创建这个类的过程中,Spring将类的依赖的属性设置进去.