IOC

IOC是什么?

IOC，**控制反转**，Inverse of Control，是一种**编程原则**，它的设计和架构可以**实现组件间的解耦**，核心思想是**将控制权转移出去**。

- 编程原则：不是某种落地技术，是一种理论。
- 组件间的解耦：
  - 什么是解耦？就是**解除耦合**，**“耦合”就是代码模块之间相互依赖的程度**。高度耦合就是关系紧密，关系越紧密，双方（或多方）之间的影响就更深，受桎梏的越多。
  - 为什么要解耦？我们一般追求**低耦合**，所以要解耦，即解除对象间的依赖关系，**减少对象间的互相影响**，增强它们的独立性。
- 如何将控制权转移出去：将原有的对象间的**主动依赖改为被动接受型依赖**。

怎么使用IOC?

例子如下(Java):

```
1 | private DemoDao dao = new DemoDaoImpl();
2 |
3 | private DemoDao dao = (DemoDao) BeanFactory.getBean("demoDao");
```

上面的就是**强依赖**，下面就是用IOC（理论）实现的**弱依赖**。怎么理解？第一句的写法是**主动声明*DemoDao***的实现类，在编译时就必须保证***DemoDaoImpl***的存在。第二句的写法**没有指定实现的类**，是叫***BeanFactory***小弟去帮我们找一个***name***为***demoDao***的对象，只有到运行期反射创建时才知道***DemoDaoImpl***是否存在。

用第二句的方式，就不是我们自己去声明了。可以理解是***BeanFactory***去用获取对象的方式。即我们将控制权交给别人，这就叫**控制反转（Inverse of Control, IOC）**。而***BeanFactory***根据指定的***beanName***去获取和创建对象的过程，就可以称作：**依赖查找（Dependency Lookup, DL）**。