1. IoC是什么

Ioc—Inversion of Control，即“控制反转”，不是什么技术，而是一种设计思想。在Java开发中，**Ioc意味着将你设计好的对象交给容器控制，而不是传统的在你的对象内部直接控制。对于spring框架来说，就是由spring来负责控制对象的生命周期和对象间的关系.**如何理解好Ioc呢？理解好Ioc的关键是要明确“谁控制谁，控制什么，为何是反转（有反转就应该有正转了），哪些方面反转了”，那我们来深入分析一下：

●**谁控制谁，控制什么**：传统Java SE程序设计，我们直接在对象内部通过new进行创建对象，是程序主动去创建依赖对象；而IoC是有专门一个容器来创建这些对象，即由Ioc容器来控制对 象的创建；**谁控制谁？当然是IoC 容器控制了对象；控制什么？那就是主要控制了外部资源获取（不只是对象包括比如文件等）。**

●**为何是反转，哪些方面反转了**：有反转就有正转，传统应用程序是由我们自己在对象中主动控制去直接获取依赖对象，也就是正转；而反转则是由容器来帮忙创建及注入依赖对象；为何是反转？**因为由容器帮我们查找及注入依赖对象，对象只是被动的接受依赖对象，所以是反转；哪些方面反转了？依赖对象的获取被反转了。**

我们通常做事的方式，如果我们需要某个对象，一般都是采用这种直接创建的方式(`new BeautifulGirl()`)，这个过程复杂而又繁琐，而且我们必须要面对每个环节，同时使用完成之后我们还要负责销毁它，在这种情况下我们的对象与它所依赖的对象耦合在一起。

我们知道，我们依赖对象其实并不是依赖该对象本身(控制层调用实现层时,需要的是实现层里面内容)，而是依赖它所提供的服务，只要在我们需要它的时候，它能够及时提供服务即可，至于它是我们主动去创建的还是别人送给我们的，其实并不是那么重要。再说了，相比于自己千辛万苦去创建它还要管理、善后而言，直接有人送过来是不是显得更加好呢?

这个给我们送东西的“人” 就是 IoC，在上面的例子中，它就相当于一个婚介公司，作为一个婚介公司它管理着很多男男女女的资料，当我们需要一个女朋友的时候，直接跟婚介公司提出我们的需求，婚介公司则会根据我们的需求提供一个妹子给我们，我们只需要负责谈恋爱，生猴子就行了。你看，这样是不是很简单明了。

1.1 注入对象的方式

IOC Service Provider 为被注入对象提供被依赖对象也有如下几种方式：构造方法注入、stter方法注入、接口注入,静态工厂模式。

1.1.1 构造器注入

构造器注入，顾名思义就是被注入的对象通过在其构造方法中声明依赖对象的参数列表，让外部知道它需要哪些依赖对象。

```
YoungMan(BeautifulGirl beautifulGirl){
        this.beautifulGirl = beautifulGirl;
}
```

构造器注入方式比较直观，对象构造完毕后就可以直接使用，这就好比你出生你家里就给你指定了你媳妇。

1.1.2 setter 方法注入

对于 JavaBean 对象而言，我们一般都是通过 getter 和 setter 方法来访问和设置对象的属性。所以，当前对象只需要为其所依赖的对象提供相对应的 setter 方法，就可以通过该方法将相应的依赖对象设置到被注入对象中。如下：

```
public class YoungMan {
    private BeautifulGirl beautifulGirl;

    public void setBeautifulGirl(BeautifulGirl beautifulGirl) {
        this.beautifulGirl = beautifulGirl;
    }
}
```

相比于构造器注入，setter 方式注入会显得比较宽松灵活些，它可以在任何时候进行注入（当然是在使用依赖对象之前），这就好比你可以先把自己想要的妹子想好了，然后再跟婚介公司打招呼，你可以要林志玲款式的，赵丽颖款式的，甚至凤姐哪款的，随意性较强。

1.1.3 接口方式注入

接口方式注入显得比较霸道，因为它需要被依赖的对象实现不必要的接口，带有侵入性。一般都不推荐这种方式。

1.1.4 静态工厂模式

> https://blog.csdn.net/shadow_zed/article/details/72566611

1.2 springIOC的实现过程

1，资源定位，找到对应的(xml)的位置

2，将资源信息加载到BeanDefines的子类中，并不创建bean实例。

3，注册BeanDefines到BeanFactory,这时候也可能还不创建Bean实例。

4，将将Beandefines子类对象放入map ，判断是否为懒加载，如果非懒加载，则创建单例bean,并将所需要的对象进行依赖注入，若是懒加载，则在完成第三步的时候就进行实例话

并进行依赖注入。

> https://www.cnblogs.com/chenssy/p/9576769.html
> https://www.cnblogs.com/xdp-gacl/p/4249939.html
> 链接：https://www.jianshu.com/p/14dffdfc330d

2. DI(依赖注入)

IoC的一个重点是在系统运行中，动态的向某个对象提供它所需要的其他对象。这一点是通过DI（Dependency Injection，依赖注入）来实现的。比如对象A需要操作数据库，以前我们总是要在A中自己编写代码来获得一个Connection对象，有了 spring我们就只需要告诉spring，A中需要一个Connection，至于这个Connection怎么构造，何时构造，A不需要知道。在系统运行时，spring会在适当的时候制造一个Connection，然后像打针一样，注射到A当中，这样就完成了对各个对象之间关系的控制。A需要依赖 Connection才能正常运行，而这个Connection是由spring注入到A中的，依赖注入的名字就这么来的。那么DI是如何实现的呢？ Java 1.3之后一个重要特征是反射（reflection），它允许程序在运行的时候动态的生成对象、执行对象的方法、改变对象的属性，spring就是通过反射来实现注入的。

> **我觉得IoC是一种思想，一种理论。DI是控制反转理论的一种实现，也许还有其他的实现方式。**
> https://www.cnblogs.com/chenssy/p/9576769.html
> https://www.cnblogs.com/xdp-gacl/p/4249939.html