### 为什么需要依赖注入

如果一个类A 的功能实现需要借助于类B，那么就称类B是类A的依赖，如果在类A的内部去实例化类B，那么两者之间会出现较高的耦合，一但类B出现了问题,类A也需要进行改造，如果这样的情况较多,每个类之间都有很多依赖，那么就会出现牵一发而动全身的情况，程序会极难维护，很容易出现问题。

### 依赖注入的效果

目的：实现类的解耦，即让类相互之间的依赖性减弱，以便于后期的维护和修改

实例：Class A中用到了Class B的对象b，一般情况下，需要在A的代码中显式的new一个B的对象。采用依赖注入技术之后，A的代码只需要定义一个私有的B对象，不需要直接new来获得这个对象，而是通过相关的容器控制程序来将B对象在外部new出来并注入到A类里的引用中。

### 依赖注入的实现

控制反转是一种思想，是能够解决问题的一种可能的结果，而依赖注入(Dependency Injection)就是其最典型的实现方法 。 由第三方(我们称作IOC容器)来控制依赖，把他通过构造函数、属性或者工厂模式等方法,注入到类A内，这样就极大程度的对类A和类B进行了解耦。

把有依赖关系的类放到容器中，解析出这些类的实例，就是依赖注入。

DI—Dependency Injection，即“依赖注入”：组件之间依赖关系由容器在运行期决定，形象的说，即由容器动态的将某个依赖关系注入到组件之中。依赖注入的目的并非为软件系统带来更多功能，**而是为了提升组件重用的频率，并为系统搭建一个灵活、可扩展的平台。**

### 依赖注入的原理

DI主要是为了避免代码的高度耦合性（tightly-coupled）。DI的主要好处在于使代码具有可扩展性extensibility, 可测试性testability，和后期绑定late binding，帮助项目更好的接近面向对象的SOLID原则

在软件工程中，依赖注入（dependency injection）的意思为，给予调用方它所需要的事物。 “依赖”是指可被方法调用的事物。依赖注入形式下，调用方不再直接指使用“依赖”，取而代之是“注入” 。“注入”是指将“依赖”传递给调用方的过程。在“注入”之后，调用方才会调用该“依赖”。传递依赖给调用方，而不是让让调用方直接获得依赖，这个是该设计的根本需求。

在编程语言层次下，“调用方”为对象和类，“依赖”为变量。在提供服务的角度下，“调用方”为客户端，“依赖”为服务。

该设计的目的是为了分离关注点，分离调用方和依赖，从而提高可读性以及代码重用性。

假如A类对B类具有依赖关系，把A类对B类的控制权抽离出来，交给一个第三方去做，把控制权反转给第三方，就称作控制反转（IOC Inversion Of Control）。依赖注射是控制反转的最为常见的一种技术。由IOC容器来控制依赖，把他通过构造函数、属性或者工厂模式等方法，注入到类A内，这样就极大程度的对类A和类B进行了解耦。

在将B的实例注入到A中时，可以有多种实现方法，Constructor Injection通过A的构造函数传入B的实例，Property Injection通过设置A的属性（字段）传入B实例，Method Injection通过调用A中的方法（接口）来传入B实例。

目前来说，有一些可以被免费使用的IOC容器，像Unity, Castle Windsor, Ninject, Autofac, StructureMap, Spring.NET等。

### 依赖注入举例：



依赖注入代码

上面代码中的就是依赖注入，现在又有一个问题，假如，现在类b中的b方法现在改名为c了，那a类里面的a方法中，就要将b类中调用的b方法改成c方法。那依旧涉及到那个问题，如果有N多个类要用到B类呢？那需要做的也是去N多个类中不断得去改，这样对后期得维护，需求更改都是要花费很大的成本。

因此，PHP中有一个接口类的说法，接口类：

PHP接口（interface）的特点

1、接口的方法必须是公开的。

2、接口的方法默认是抽象的，所以不在方法名前面加abstract。

3、接口可以定义常量，但不能定义成员属性，常量的定义和用法和类中常量一样。

4、类可以实现多个接口（相当于把多个功能集于一身，如手机实现了小灵通、MP3、MP4的功能）

5、接口也可以继承接口。



图中可以看到，b类是一个接口类，c,d继承了b类，接口类的具体规则，则可以自己百度一下，代码中，我们延续了，第一图的代码做为修改，把普通类，改为接口类来实现依赖注入，就能解决那个遗留的问题，当依赖注入的类中的方法，改变了，那我们需要做的就是，在原有的类中，创建一个子类，接着写上相同得方法，然后在方法中写上需要的内容和逻辑就行了，当需要得时候，直接实例化，然后注入到需要的类中就行，哪怕是那些类中原有存在的方法都不需要更改。