**控制反转（ioc）**

控制反转：对象在获取它依赖的对象的过程被反转

如下示例：

public class A {

    public IB b { get; set; }

    public A(){

        // A对象依赖与IB接口，如果我们直接在构造函数中实例，那么会增加A对B的耦合

        // 对于日后对A的测试或B的功能扩展都难以展开

        b = new B();

    }

}

public interface IB {}

public class B : IB {

}

public class AppService{

    public void Fun1() {

        var a = new A();

    }

}

对于A所存在的问题，我们决定引入一个IocContainer对象，让A不在负责B的创建，而由IocContainer创建B，并传给A，这样就完成了对象B的创建反转到IocContainer中

public class A {

    public IB b { get; set; }

    public A(){

        this.b = IocContainer.GetService<IB>();

    }

}

public interface IB {}

public class B : IB {

}

**依赖注入（DI）**

依赖注入：将对象的依赖传递给对象来创建对象来实现控制反转

依赖注入与控制反转的实现都是相同的，可以说依赖注入就是控制反转，反正我说不清