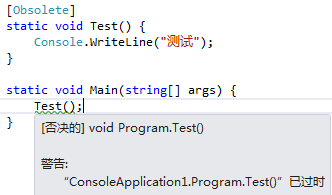
特性(Attributes)，MSDN的定义是：公共语言运行时允许你添加类似关键字的描述声明，叫做attributes, 它对程序中的元素进行标注，如类型、字段、方法和属性等。Attributes和Microsoft .NET Framework文件的元数据保存在一起，可以用来向运行时描述你的代码，或者在程序运行的时候影响应用程序的行为。  
例如，在一个方法前标注[Obsolete]特性，则调用该方法时VS则会提示该方法已过期的警告，如下图：  


又如，在.Net Remoting的远程对象中，如果要调用或传递某个对象，例如类，或者结构，则该类或结构则必须标注[Serializable]特性。还有，我们在构建XML Web服务时用得很多的一个特性就是[WebMegthod]，它可让通过HTTP请求的公开方法的返回值编码成XML进行传递。

特性实际上就是一个类，[Obsolete]特性的实际类名是ObsoleteAttribute，但我们在标注的时候可以不带Attribute后缀，系统在名称转换时会自动给我们加上。

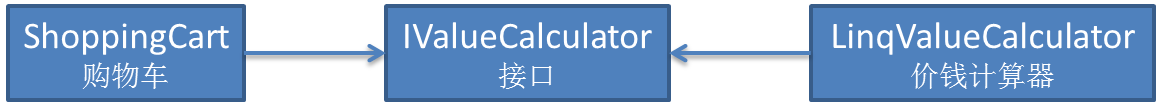
在一个类内部，不通过创建对象的实例而能够获得某个实现了公开接口的对象的引用。这种“需要”，就称为DI（依赖注入，Dependency Injection），和所谓的IoC（控制反转，Inversion of Control ）是一个意思。

DI是一种通过接口实现松耦合的设计模式。初学者可能会好奇网上为什么有那么多技术文章对DI这个东西大兴其笔，是因为DI对于基于几乎所有框架下，要高效开发应用程序，它都是开发者必须要有的一个重要的理念，包括MVC开发。它是解耦的一个重要手段。

DI模式可分为两个部分。一是移除对组件（上面示例中的LinqValueCalculator）的依赖，二是通过类的构造函数（或类的Setter访问器）来传递实现了公开接口的组件的引用。如下面代码所示：



这样我们就彻底断开了ShoppingCart和LinqValueCalculator之间的依赖关系。某个实现了IValueCalculator接口的类(示例中的LinqValueCalculator) 的实例引用作为参数，传递给ShoppingCart类的构造函数。但是ShoppingCart类不知道也不关心这个实现了 IValueCalculator接口的类是什么，更没有责任去操作这个类。 这时我们可以用下图来描述ShoppingCart、LinqValueCalculator和IValueCalculator之间的关系：



在程序运行的时候，依赖被注入到ShoppingCart，这个依赖就是，通过ShoppingCart构造函数传递实现了 IValueCalculator接口的类的实例引用。在程序运行之前（或编译时），ShoppingCart和任何实现 IValueCalculator接口的类没有任何依赖关系。（注意，程序运行时是有具体依赖关系的。）

注意，上面示例使用的注入方式称为“构造注入”，我们也可以通过属性来实现注入，这种注入被称为“setter 注入”，就不举例了，朋友们可以看看[T2噬菌体](http://home.cnblogs.com/u/leoo2sk/)的文章[依赖注入那些事儿](http://www.cnblogs.com/leoo2sk/archive/2009/06/17/di-and-ioc.html)来对DI进行更多的了解。

由于经常会在编程时使用到DI，所以出现了一些DI的辅助工具（或叫DI容器），如Unity和Ninject等。由于Ninject的轻量和使用 简单，加上本人只用过Ninject，所以本系列文章选择用它来开发MVC应用程序。下面开始介绍Ninject，但在这之前，先来介绍一个安装 Ninject需要用到的插件-NuGet。