**反射Reflection**

**反射是什么：**

1. 一个程序集通过反射，动态的获得这个程序集中的元数据
2. 通过反射，我们不需要拿到一个程序集中的某个类去创建对象就可以拿到该类中的成员
3. 反射，改变了创建对象的方式

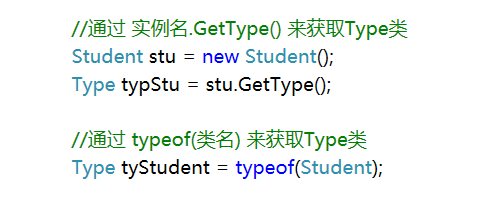
**反射带来的好处：**

1. 反射是.NET中的重要机制，通过反射，可以在运行时获得一个指定程序集中的所有元数据（类、结构、委托、接口、枚举、属性、字段、函数等等）
2. 反射能够降低程序的耦合度，提升自适应能力，还能够提升程序的可扩展性

**实现反射的重要类 ---> Type：**

Type类的详解：

1. Type类，是一个描述类的类型，任何类型都可以转换成Type类，当转换后，Type类就能够说明该类的类名以及该类的相关的属性、方法等等，总而言之，Type类能够给出一个类的相关信息
2. Type类主要应用于反射中  
   如何获取Type类：
3. **普通**的使用方式：我们可以把任何类型转换成Type类，并通过Type类的对象获取该类的相关信息



1. 使用**反射时**的使用方式：当我们使用反射载入了一个程序集，我们便可以使用以下方法来获取我们想要在程序集中获取的某个类的Type类型，而Type类型用于描述该类的相关信息，所以我们可以使用该类的Type类型的对象，对该类中的所有成员进行读取、使用、赋值等操作



**反射的具体使用：使用Type类的对象获取该类的相关信息**

1. 我们先获取程序集的绝对路径，通过 Assembly 去载入一个程序集，然后我们可以通过方法一、二、三，来获取程序集中我们所需的指定的类型或所有的类型，这些方法返回一个Type类型的对象或数组，Type类型的对象用于描述我们需要获取的类的一切相关讯息



1. 当我们获得了该类的Type类型的对象后，我们便可以使用Type对象，对该类进行创建对象的操作，我们可以通过以下函数，动态的创建程序集中指定类型的对象：

[ object Activator.CreateInstance(Type t,params objetc[] os) ]

Type t：需要创建对象的类的Type类

params可变参数：该类型的构造函数所需的参数

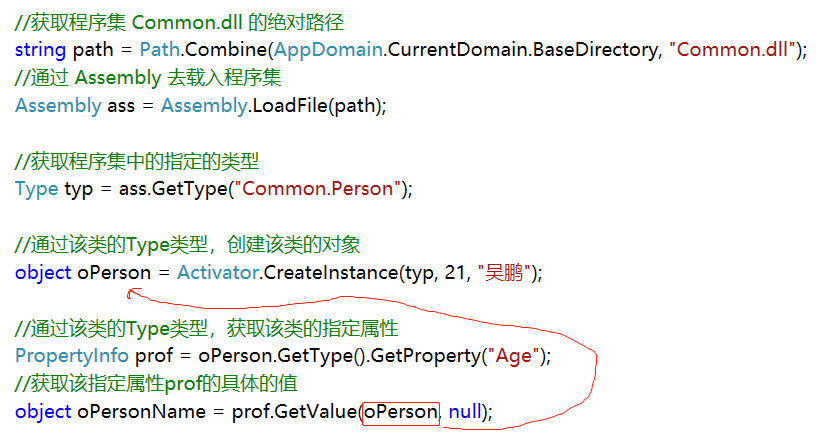


1. 当我们创建完对象后，这个对象中的属性、字段等等都拥有了相应的值，但是上面步骤返回的却是个object类，我们无法直接通过object类来访问我们所需类中的成员，但是我们可以通过 GetType() 把它转成Type类型，然后再通过Type的对象对该类的属性等成员的值进行访问操作，我们通过Type类型对该类的成员进行操作用到了以下函数：

PropertyInfo type.GetProperty("name")：获取该类的属性值名为：name的属性**(还不能直接使用)**

PropertyInfo[] type.GetProperties()：获取该类的所有属性，返回一个集合**(还不能直接使用)**

object properyinfo.GetValue(指定类的对象,null)：获取该类指定属性的值



1. 我们除了能通过该类的Type类型来获取该类的属性之外还能通过该类的Type类型来获取该类的方法、字段等等，以下只演示获取函数的方法，方法类似于上一步骤，以下列出三种获取函数的方法：

MethodInfo type.GetMethod("name")：获取该类的方法名为：name的方法**(还不能直接使用)**

MethodInfo[] type.GetMethods()：获取该类的所有方法，返回一个集合**(还不能直接使用)**

methodinfo.Invoke(指定类的对象,方法的参数)：执行该类的函数



1. Type类的4个拓展常用的函数：

