

# “委托”基础

---

北京理工大学计算机学院  
金旭亮

在学习具体内容之前，先强调一下：

不学会委托（Delegate），  
等于没学.NET编程！

# 委托引例 ( FirstDelegateExample )

1

这是一个普通的类，它定义了两个将被委托变量接收的方法

```
public class MathOpt
{
    2个引用
    public int Add(int x, int y)
    {
        return x + y;
    }

    2个引用
    public static int Max(int x,int y)
    {
        return (x > y || x == y) ? x : y;
    }
}
```

2

定义一个委托类型，它有两个int类型的参数，返回一个int类型的数值

```
public delegate int MathOptDelegate(int value1, int value2);
```

3

委托是一种用户自定义的数据类型，可以用于定义变量

```
MathOptDelegate oppDel;
```

4

```
//委托变量可以接收一个实例方法引用
MathOpt obj = new MathOpt();
oppDel = obj.Add;
//也可以接收一个静态方法引用
oppDel = MathOpt.Max;
```

5

委托变量可以当成普通方法那样调用

```
Console.WriteLine(oppDel(1, 2)); //输出 3
```

6

可以定义委托类型的参数:

```
static int UseDelegate(MathOptDelegate option, int x,int y)
{
    return option(x, y);
}
```

7

可以把方法引用直接传给委托类型的参数

```
Console.WriteLine(UseDelegate(obj.Add,10,20));
Console.WriteLine(UseDelegate(MathOpt.Max,100,200));
```

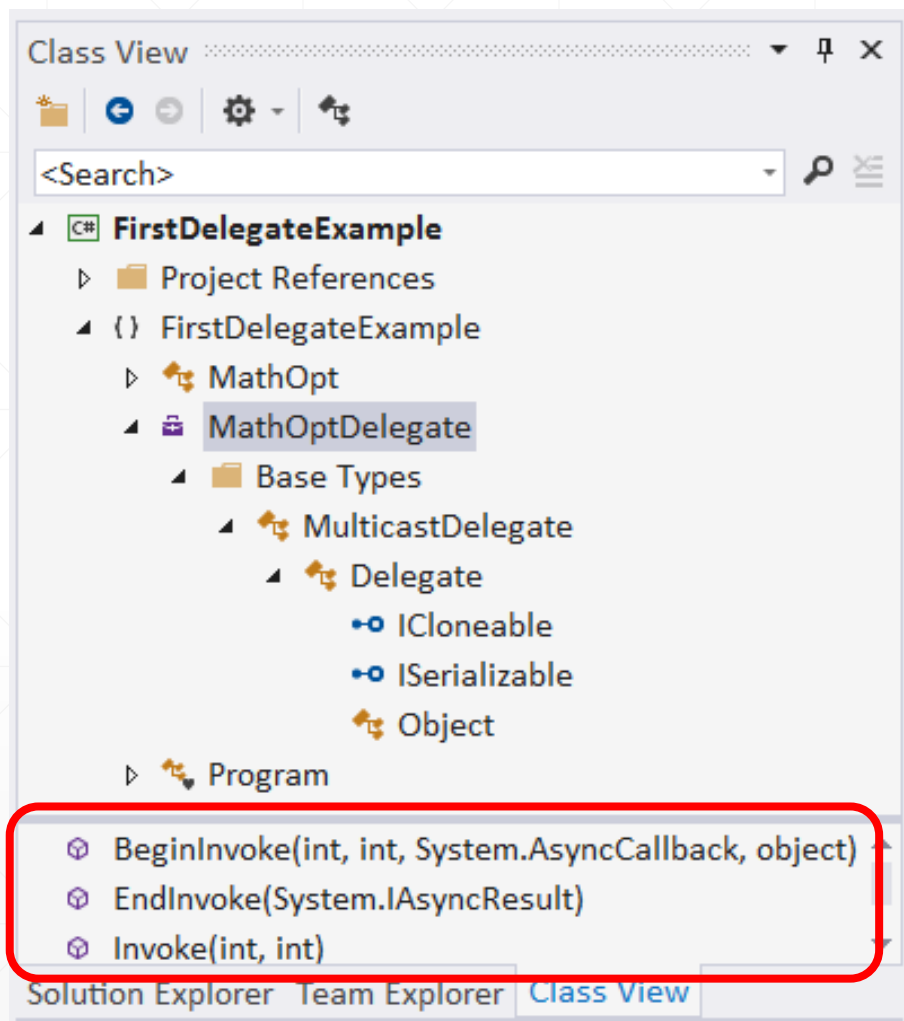
# 从引例中我们可以知道

委托实际上是一种“**数据类型**”，我们可以使用它来定义变量。

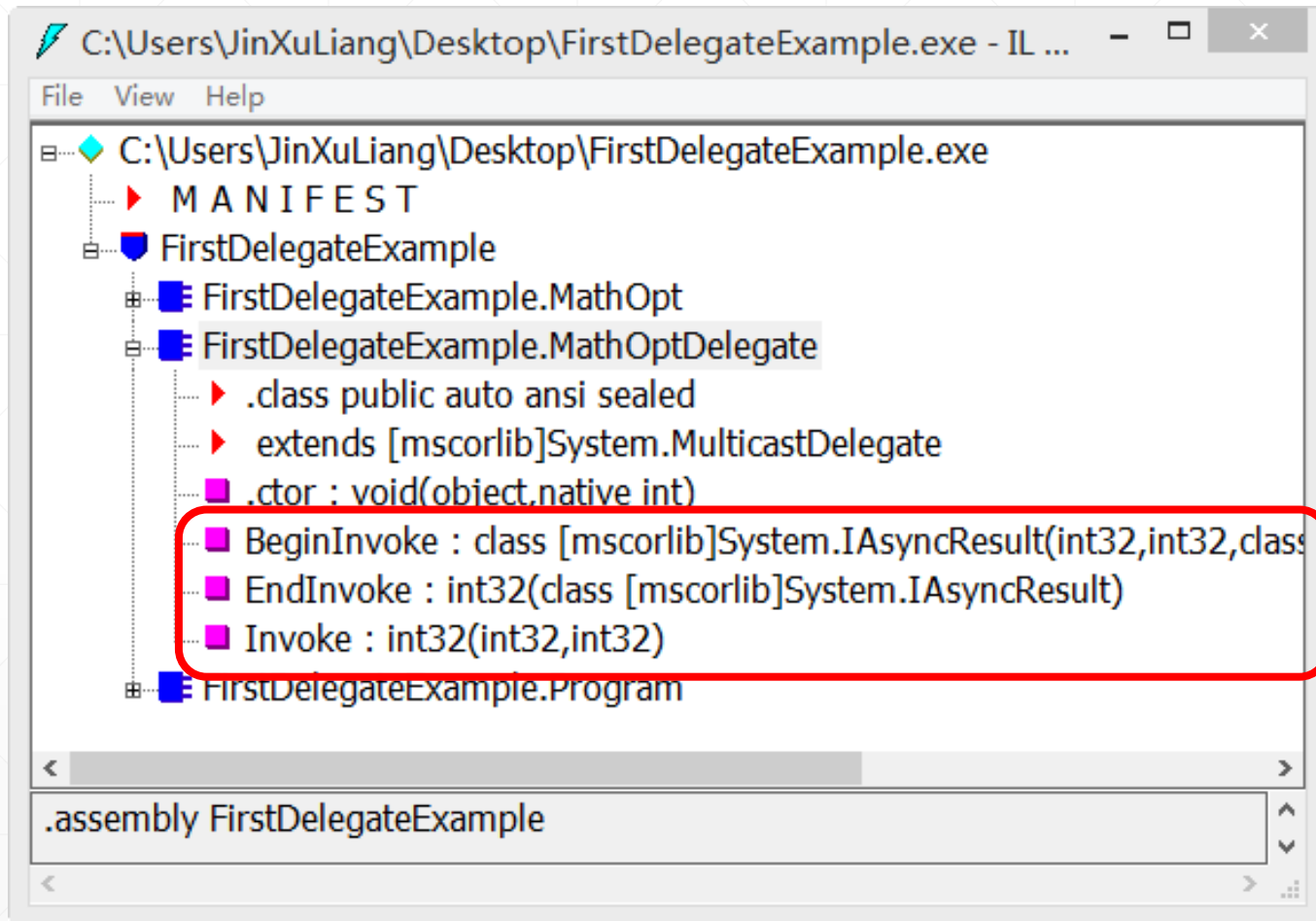
一个委托类型的变量，可以引用任何一个**满足其要求**的方法（包括静态或实例方法）。

委托变量有点像是一个方法的“**容器**”，将某一具体的方法“装入”后，这个委托变量就可以当成方法一样调用。

# 详解委托类型



委托的继承树



编译器为委托生成的类型

```
public delegate int MathOptDelegate(int value1, int value2);
```

C#编译器

当调用实例方法时，此参数引用被调用方法的那个对象

这个参数引用被调用的那个方法

```
public class MathOptDelegate : System.MulticastDelegate
{
    public MathOptDelegate(Object target, Int32 methodPtr);
    public virtual Int32 Invoke(Int32 value1, Int32 value2);
    public IAsyncResult BeginInvoke(
        Int32 value1, Int32 value2,
        AsyncCallback callback, Object object);
    public Int32 EndInvoke ( IAsyncResult result);
}
```

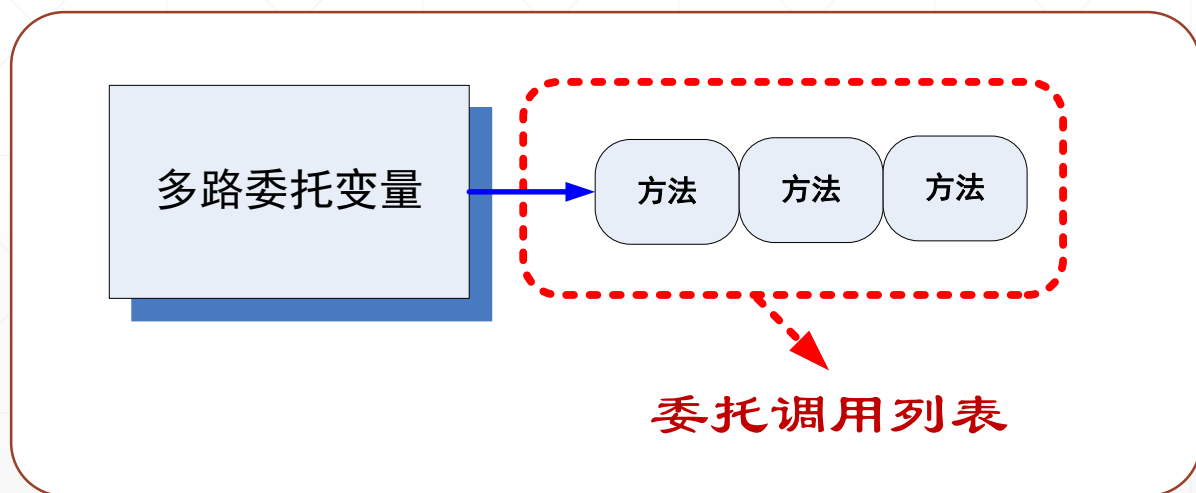
当通过委托变量调用方法时，实际上调用的就是这个Invoke()方法

委托内置对异步调用的支持。

# 委托的组合与分解

一个委托变量可以使用“**+=**”不断“挂接”多个方法，也能使用“**-=**”动态地移除某个方法引用，还可以把多个委托变量所引用的方法合并起来。


引用多个方法委托变量称为“**多路委托**”。



示例：MulticastDelegateInvocationList



# 委托应用之“定时回调”



A screenshot of a Windows command prompt window. The title bar shows the file path: file:///E:/MOOC/录课视频相关文件/Week2/3 .NET核心技术选... The window contains the following text:

```
敲任意键结束.....  
(1)2014/12/7 13:49:31  
(2)2014/12/7 13:49:32  
(3)2014/12/7 13:49:33  
(4)2014/12/7 13:49:34  
(5)2014/12/7 13:49:35  
(6)2014/12/7 13:49:36  
(7)2014/12/7 13:49:37  
(8)2014/12/7 13:49:38  
(9)2014/12/7 13:49:39  
(10)2014/12/7 13:49:40
```

Timer是.NET基类库中所提供的一个定时器对象，它的构造函数接收一个TimerCallBack委托对象，此对象引用一个将被Timer对象定时“回调”的方法

示例：UseTimerCallback