C# 多线程参数传递 - 文章

1、通过实体类来传递（可以传递多个参数与获取返回值），demo如下：

需要在线程中调用的函数：

```
C#
namespace ThreadParameterDemo
{
        public class FunctionClass
        {
                public static string TestFunction(string name, int age)
                {
                        //内部处理省略
                        return name + " 的年龄是：" + age;
                }
        }
}C#
namespace ThreadParameterDemo
{
        ///
        /// 过渡类
        ///
        public class TransitionalClass
        {
                private string name = string.Empty;
                private int age;
                public string acceptResults = string.Empty;
                public TransitionalClass(string name, int age)
                {
                        this.name = name;
                        this.age = age;
                }
                public void TestFunction()
                {
                        acceptResults = FunctionClass.TestFunction(this.name, this.age);
                }
```

```
        }
}
    private void Form1_Load(object sender, EventArgs e)
                {
                        //实例化ThreadWithState类，为线程提供参数
                        TransitionalClass tc = new TransitionalClass(" Jack", 42);
                        // 创建执行任务的线程，并执行
                        Thread t = new Thread(new ThreadStart(tc.TestFunction));
                        t.Start();
                        //获取返回值，通过 tc.acceptResults;
                }
```

小注：

- 必须注意IsBackground的问题，如果IsBackground为false的，则Windows程序在退出的时候，不会为你自动退出该线程。也就是实际上你的应用程序未结束。
- MSDN推荐：多线程方法调用提供参数的最好办法是将目标方法包裹在类中，并为该类定义字段，这些字段将被用作新线程的参数。
- 这种方法的优点是，任何时候想要启动新线程，都可以创建类的新实例，该实例带有自身的参数。
- ThreadStart中的函数是没有返回值和参数的

2、异步调用中的参数和返回值

能完美解决参数和返回值的是使用异步调用的方式。异步调用和Thread相比，一个最大的劣势是不能控制其优先级。

具体代码如下：

```
                public delegate string delegateFunction(string name,int age);//委托
                delegateFunction df;
                private void Form1_Load(object sender, EventArgs e)
                {
                        //指向需要调用的方法
                        df = new delegateFunction(FunctionClass.TestFunction);
                        string name = "my name";//输入参数
                        int age = 19;
                        IAsyncResult result = df.BeginInvoke(name,age, null, null);
                        string myResult = df.EndInvoke(result);//用于接收返回值
                        MessageBox.Show(myResult);
                }
public Func     df;//委托
                private void Form1_Load(object sender, EventArgs e)
                {
                        //指向需要调用的方法
                        df += FunctionClass.TestFunction;
```

```
                string name = "my name";//输入参数
                int age = 19;
                IAsyncResult result = df.BeginInvoke(name, age, null, null);
                string myResult = df.EndInvoke(result);//用于接收返回值
                MessageBox.Show(myResult);
            }
```

小注：

通过这种方式生成新线程是运行在后台的（background）,优先级为normal

3、使用 BackgroundWorker

多线程返回值最简单方法是：使用 BackgroundWorker 组件来管理线程，在任务完成时引发事件，然后用事件处理程序处理结果。

小注：
BackgroundWorker 组件用来执行诸如数据库事务、文件下载等耗时的异步操作。
在应用程序中添加一个BackgroundWorker实例，如果用的是VS，可以从工具上直接拖到应用程序：

C#
1
BackgroundWorker backgroundWorker1 = new BackgroundWorker();
为了开始在后台操作，必须调用BackgroundWorker的RunWorkerAsync()方法，当调用此方时，
BackgroundWorker 通过触发DoWork 事件，开始执行后台操作，DoWork 事件的代码是在另一个线程里执行的。

当后台操作完成以后，无论是completed 还是cancelled，则RunWorkerCompleted 事件被触发，通过此方法可以将后台操作的完成结果反馈给用户。

另外，通过RunWorkerCompletedEventArgs实例的Cancelled 属性，以判断是否是Cancel操作使得后台操作终止。

具体demo如下：

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }
        private void Form2_Load(object sender, EventArgs e)
```

```
        {
                //TransitionalClass tc = new TransitionalClass("xiaoming", 10);
                //ThreadPool.QueueUserWorkItem(new
WaitCallback(TransitionalClass.TestFunction), tc);
        }
        private void button1_Click(object sender, EventArgs e)
        {
                this.TestArea2();
        }
        private System.ComponentModel.BackgroundWorker BackgroundWorker1
    = new System.ComponentModel.BackgroundWorker();
        private void TestArea2()
        {
                InitializeBackgroundWorker();
                AreaClass2 AreaObject2 = new AreaClass2();
                AreaObject2.Base = 30;
                AreaObject2.Height = 40;
                // Start the asynchronous operation.
                BackgroundWorker1.RunWorkerAsync(AreaObject2);
        }
        private void InitializeBackgroundWorker()
        {
                // Attach event handlers to the BackgroundWorker object.
                BackgroundWorker1.DoWork +=
                        new
System.ComponentModel.DoWorkEventHandler(BackgroundWorker1_DoWork);
                BackgroundWorker1.RunWorkerCompleted +=
                        new
System.ComponentModel.RunWorkerCompletedEventHandler(BackgroundWorker1_RunWorkerCompleted);
        }
        private void BackgroundWorker1_DoWork(
                object sender,
                System.ComponentModel.DoWorkEventArgs e)
        {
                //在执行DoWork 事件时，DoWorkEventArgs 实例的Result 属性，返回值到
用户；在RunWorkerCompleted 事件里，RunWorkerCompletedEventArgs 实例的Result 属性接收值；
                AreaClass2 AreaObject2 = (AreaClass2)e.Argument;
                // Return the value through the Result property.
                e.Result = AreaObject2.CalcArea();
        }
        private void BackgroundWorker1_RunWorkerCompleted(
                object sender,
```

```
                System.ComponentModel.RunWorkerCompletedEventArgs e)
        {
            // Access the result through the Result property.
            double Area = (double)e.Result;
            MessageBox.Show("The area is: " + Area.ToString());
        }
    }
}
```

demo代码来自MSDN：点击打开链接

4、如果不如返回值的时候，应该怎么优雅的写呢？匿名函数啊

FunctionClass类新增，测试函数如下：

C#