

C# 事件 (Event)

事件 (Event) 基本上说是一个用户操作，如按键、点击、鼠标移动等等，或者是一些出现，如系统生成的通知。应用程序需要在事件发生时响应事件。例如，中断。事件是用于进程间通信。

通过事件使用委托

事件在类中声明且生成，且通过使用同一个类或其他类中的委托与事件处理程序关联。包含事件的类用于发布事件。这被称为**发布者 (publisher)** 类。其他接受该事件的类被称为**订阅器 (subscriber)** 类。事件使用**发布-订阅 (publisher-subscriber)** 模型。

发布者 (publisher) 是一个包含事件和委托定义的对象。事件和委托之间的联系也定义在这个对象中。发布者 (publisher) 类的对象调用这个事件，并通知其他的对象。

订阅器 (subscriber) 是一个接受事件并提供事件处理程序的对象。在发布者 (publisher) 类中的委托调用订阅器 (subscriber) 类中的方法 (事件处理程序)。

声明事件 (Event)

在类的内部声明事件，首先必须声明该事件的委托类型。例如：

```
public delegate void BoilerLogHandler(string status);
```

然后，声明事件本身，使用 **event** 关键字：

```
// 基于上面的委托定义事件
public event BoilerLogHandler BoilerEventLog;
```

上面的代码定义了一个名为 *BoilerLogHandler* 的委托和一个名为 *BoilerEventLog* 的事件，该事件在生成的时候会调用委托。

实例

实例 1

```
using System;
namespace SimpleEvent
{
    using System;
    /***** 发布者类 *****/
    public class EventTest
    {
        private int value;

        public delegate void NumManipulationHandler();
```

```
public event NumManipulationHandler ChangeNum;
protected virtual void OnNumChanged()
{
    if ( ChangeNum != null )
    {
        ChangeNum(); /* 事件被触发 */
    }else {
        Console.WriteLine( "event not fire" );
        Console.ReadKey(); /* 回车继续 */
    }
}

public EventTest()
{
    int n = 5;
    SetValue( n );
}

public void SetValue( int n )
{
    if ( value != n )
    {
        value = n;
        OnNumChanged();
    }
}

/*****订阅器类*****/

public class subscribEvent
{
    public void printf()
    {
        Console.WriteLine( "event fire" );
        Console.ReadKey(); /* 回车继续 */
    }
}

/*****触发*****/
public class MainClass
{
    public static void Main()
    {
        EventTest e = new EventTest(); /* 实例化对象, 第一次没有触发事件 */
        subscribEvent v = new subscribEvent(); /* 实例化对象 */
        e.ChangeNum += new EventTest.NumManipulationHandler( v.printf ); /* 注册 */
        e.SetValue( 7 );
        e.SetValue( 11 );
    }
}
```

```
}  
}  
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
event not fire  
event fire  
event fire
```

本实例提供一个简单的用于热水锅炉系统故障排除的应用程序。当维修工程师检查锅炉时，锅炉的温度和压力会随着维修工程师的备注自动记录到日志文件中。

实例 2

```
using System;  
using System.IO;  
  
namespace BoilerEventAppl  
{  
  
    // boiler 类  
    class Boiler  
    {  
        private int temp;  
        private int pressure;  
        public Boiler(int t, int p)  
        {  
            temp = t;  
            pressure = p;  
        }  
  
        public int getTemp()  
        {  
            return temp;  
        }  
        public int getPressure()  
        {  
            return pressure;  
        }  
    }  
  
    // 事件发布者  
    class DelegateBoilerEvent  
    {  
        public delegate void BoilerLogHandler(string status);  
  
        // 基于上面的委托定义事件  
        public event BoilerLogHandler BoilerEventLog;  
  
        public void LogProcess()  
        {  
            string remarks = "0. K";  
        }  
    }  
}
```

```

        Boiler b = new Boiler(100, 12);
        int t = b.getTemp();
        int p = b.getPressure();
        if(t > 150 || t < 80 || p < 12 || p > 15)
        {
            remarks = "Need Maintenance";
        }
        OnBoilerEventLog("Logging Info:\n");
        OnBoilerEventLog("Temperature " + t + "\nPressure: " + p);
        OnBoilerEventLog("\nMessage: " + remarks);
    }

    protected void OnBoilerEventLog(string message)
    {
        if (BoilerEventLog != null)
        {
            BoilerEventLog(message);
        }
    }
}
// 该类保留写入日志文件的条款
class BoilerInfoLogger
{
    FileStream fs;
    StreamWriter sw;
    public BoilerInfoLogger(string filename)
    {
        fs = new FileStream(filename, FileMode.Append, FileAccess.Write);
        sw = new StreamWriter(fs);
    }
    public void Logger(string info)
    {
        sw.WriteLine(info);
    }
    public void Close()
    {
        sw.Close();
        fs.Close();
    }
}
// 事件订阅器
public class RecordBoilerInfo
{
    static void Logger(string info)
    {
        Console.WriteLine(info);
    } //end of Logger

    static void Main(string[] args)
    {
        BoilerInfoLogger filelog = new BoilerInfoLogger("e:\\boiler.txt");
        DelegateBoilerEvent boilerEvent = new DelegateBoilerEvent();
        boilerEvent.BoilerEventLog += new
        DelegateBoilerEvent.BoilerLogHandler(Logger);
    }
}

```

```
boilerEvent.BoilerEventLog += new
DelegateBoilerEvent.BoilerLogHandler(filelog.Logger);
boilerEvent.LogProcess();
Console.ReadLine();
filelog.Close();
} //end of main

} //end of RecordBoilerInfo
}
```

当上面的代码被编译和执行时，它会产生下列结果：

Logging info:

Temperature 100

Pressure 12

Message: 0. K

← C# 委托 (Delegate)

C# 集合 (Collection) →



8 篇笔记

 写笔记