How to: Raise and Consume Events

03/29/2017 • 5 minutes to read • +2

In this article

The examples in this topic show how to work with events. They include examples of the EventHandler delegate, the EventHandler<TEventArgs> delegate, and a custom delegate, to illustrate events with and without data.

The examples use concepts described in the Events article.

Example

The first example shows how to raise and consume an event that doesn't have data. It contains a class named `Counter` that has an event named `ThresholdReached`. This event is raised when a counter value equals or exceeds a threshold value. The EventHandler delegate is associated with the event, because no event data is provided.

C# Copy

```csharp
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Counter c = new Counter(new Random().Next(10));
            c.ThresholdReached += c_ThresholdReached;

            Console.WriteLine("press 'a' key to increase total");
            while (Console.ReadKey(true).KeyChar == 'a')
            {
                Console.WriteLine("adding one");
                c.Add(1);
            }
        }

        static void c_ThresholdReached(object sender, EventArgs e)
        {
            Console.WriteLine("The threshold was reached.");
            Environment.Exit(0);
        }
    }

    class Counter
    {
        private int threshold;
        private int total;

        public Counter(int passedThreshold)
        {
            threshold = passedThreshold;
        }

        public void Add(int x)
        {
            total += x;
            if (total >= threshold)
            {
                OnThresholdReached(EventArgs.Empty);
            }
        }

        protected virtual void OnThresholdReached(EventArgs e)
        {
            EventHandler handler = ThresholdReached;
            if (handler != null)
            {
                handler(this, e);
            }
        }

        public event EventHandler ThresholdReached;
    }
}
```

Example

The next example shows how to raise and consume an event that provides data. The EventHandler<TEventArgs> delegate is associated with the event, and an instance of a custom event data object is provided.

C# Copy

```
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Counter c = new Counter(new Random().Next(10));
            c.ThresholdReached += c_ThresholdReached;

            Console.WriteLine("press 'a' key to increase total");
            while (Console.ReadKey(true).KeyChar == 'a')
            {
                Console.WriteLine("adding one");
                c.Add(1);
            }
        }

        static void c_ThresholdReached(object sender, ThresholdReachedEventArgs
e)
        {
            Console.WriteLine("The threshold of {0} was reached at {1}.",
e.Threshold,  e.TimeReached);
            Environment.Exit(0);
        }
    }

    class Counter
    {
        private int threshold;
        private int total;

        public Counter(int passedThreshold)
        {
            threshold = passedThreshold;
        }

        public void Add(int x)
        {
            total += x;
            if (total >= threshold)
            {
                ThresholdReachedEventArgs args = new
ThresholdReachedEventArgs();
                args.Threshold = threshold;
                args.TimeReached = DateTime.Now;
                OnThresholdReached(args);
            }
        }

        protected virtual void OnThresholdReached(ThresholdReachedEventArgs e)
        {
            EventHandler<ThresholdReachedEventArgs> handler = ThresholdReached;
            if (handler != null)
            {
                handler(this, e);
            }
        }

        public event EventHandler<ThresholdReachedEventArgs> ThresholdReached;
    }

    public class ThresholdReachedEventArgs : EventArgs
    {
        public int Threshold { get; set; }
        public DateTime TimeReached { get; set; }
    }
}
```

Example

The next example shows how to declare a delegate for an event. The delegate is named `ThresholdReachedEventHandler`. This is just an illustration. Typically, you do not have to declare a delegate for an event, because you can use either the EventHandler or the EventHandler<TEventArgs> delegate. You should declare a delegate only in rare scenarios, such as making your class available to legacy code that cannot use generics.

C# Copy

```
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Counter c = new Counter(new Random().Next(10));
            c.ThresholdReached += c_ThresholdReached;
            Console.WriteLine("press 'a' key to increase total");
            while (Console.ReadKey(true).KeyChar == 'a')
            {
                Console.WriteLine("adding one");
                c.Add(1);
            }
        }

        static void c_ThresholdReached(Object sender, ThresholdReachedEventArgs
e)
        {
            Console.WriteLine("The threshold of {0} was reached at {1}.",
e.Threshold, e.TimeReached);
            Environment.Exit(0);
        }
    }

    class Counter
    {
        private int threshold;
        private int total;

        public Counter(int passedThreshold)
        {
            threshold = passedThreshold;
        }

        public void Add(int x)
        {
            total += x;
            if (total >= threshold)
            {
                ThresholdReachedEventArgs args = new
ThresholdReachedEventArgs();
                args.Threshold = threshold;
                args.TimeReached = DateTime.Now;
                OnThresholdReached(args);
            }
        }

        protected virtual void OnThresholdReached(ThresholdReachedEventArgs e)
        {
            ThresholdReachedEventHandler handler = ThresholdReached;
            if (handler != null)
            {
                handler(this, e);
            }
        }

        public event ThresholdReachedEventHandler ThresholdReached;
    }

    public class ThresholdReachedEventArgs : EventArgs
    {
        public int Threshold { get; set; }
        public DateTime TimeReached { get; set; }
    }

    public delegate void ThresholdReachedEventHandler(Object sender,
ThresholdReachedEventArgs e);
}
```

See also

- Events