عنوان مضمون

Visual Programming-II

توسط : صفری

بهار1398

Events

What Is an Event?

- **Events** are user actions such as key press, clicks, mouse movements, etc., or some occurrence such as system generated notifications.
- Applications need to respond to events when they occur. For example, interrupts. Events are used for inter-process communication.
- **Events** enable an object to notify other objects when something of interest occurs. The object that raises the event is called the **publisher** and the objects that handle the event are called **subscribers.**

Publisher & subscriber

- A **publisher** is an object that contains the definition of the event and the delegate. The event-delegate association is also defined in this object. A publisher class object invokes the event and it is notified to other objects.
- A **subscriber** is an object that accepts the event and provides an event handler. The delegate in the publisher class invokes the method (event handler) of the subscriber class.

Event handlers themselves are simply methods. The only restriction on an event handler method is that it must match the return type and parameters required by the event. This restriction is part of the definition of an event and is specified by a *delegate*.

Delegate

- C# delegates are similar to pointers to functions, in C or C++. A **delegate** is a reference type variable that holds the reference to a method. The reference can be changed at runtime.
- Delegates are especially used for implementing events and the call-back methods. All delegates are implicitly derived from the **System.Delegate** class.

Declaring Delegates

Delegate declaration determines the methods that can be referenced by the delegate. A delegate can refer to a method, which has the same signature as that of the delegate.

Syntax for delegate declaration is :

```
{access-modifier} delegate {return-type} {name}([parameters]);
```

For example, consider a delegate :

```
public delegate int MethodPointer(int number1, int number2);
```

The preceding delegate can be used to reference any method that have two int parameter and returns an int type variable.

Instantiating Delegates

- Once a delegate type is declared, a delegate object must be created with the **new** keyword and be associated with a particular method. When creating a delegate, the argument passed to the **new** expression is written similar to a method call, but without the arguments to the method. For example :

```
public static int Sum(int number1, int number2)
{
    return number1 + number2;
}
static void Main(string[] args)
{
    MethodPointer pointer = new MethodPointer( Sum);
    Console.WriteLine(pointer(2, 5));
    Console.ReadKey();
}
```

Multicasting of a Delegate

- Delegate objects can be composed using the "+" operator. A composed delegate calls the two delegates it was composed from. Only delegates of the same type can be composed. The "-" operator can be used to remove a component delegate from a composed delegate.
- Using this property of delegates you can create an invocation list of methods that will be called when a delegate is invoked. This is called **multicasting** of a delegate. The next slide program demonstrates multicasting of a delegate

```
public delegate viod MethodPointer(int number1, int number2);
public static void Sum(int number1, int number2)
   {     Console.WriteLine(number1 + number2);
   }
public static void Substract(int number1, int number2)
   {      Console.WriteLine(number1 - number2);
   }
static void Main(string[] args)
   {     MethodPointer pointer = new MethodPointer( Sum);
         pointer += Substract;
         pointer(8, 3);
         Console.ReadKey();
   }
```

Declaring Events

- To declare an event inside a class, first a delegate type for the event must be declared. For example,

```
public delegate string MyDel(string str);
```

- Next, the event itself is declared, using the event keyword −

```
event Delegate-name Event-name;
```

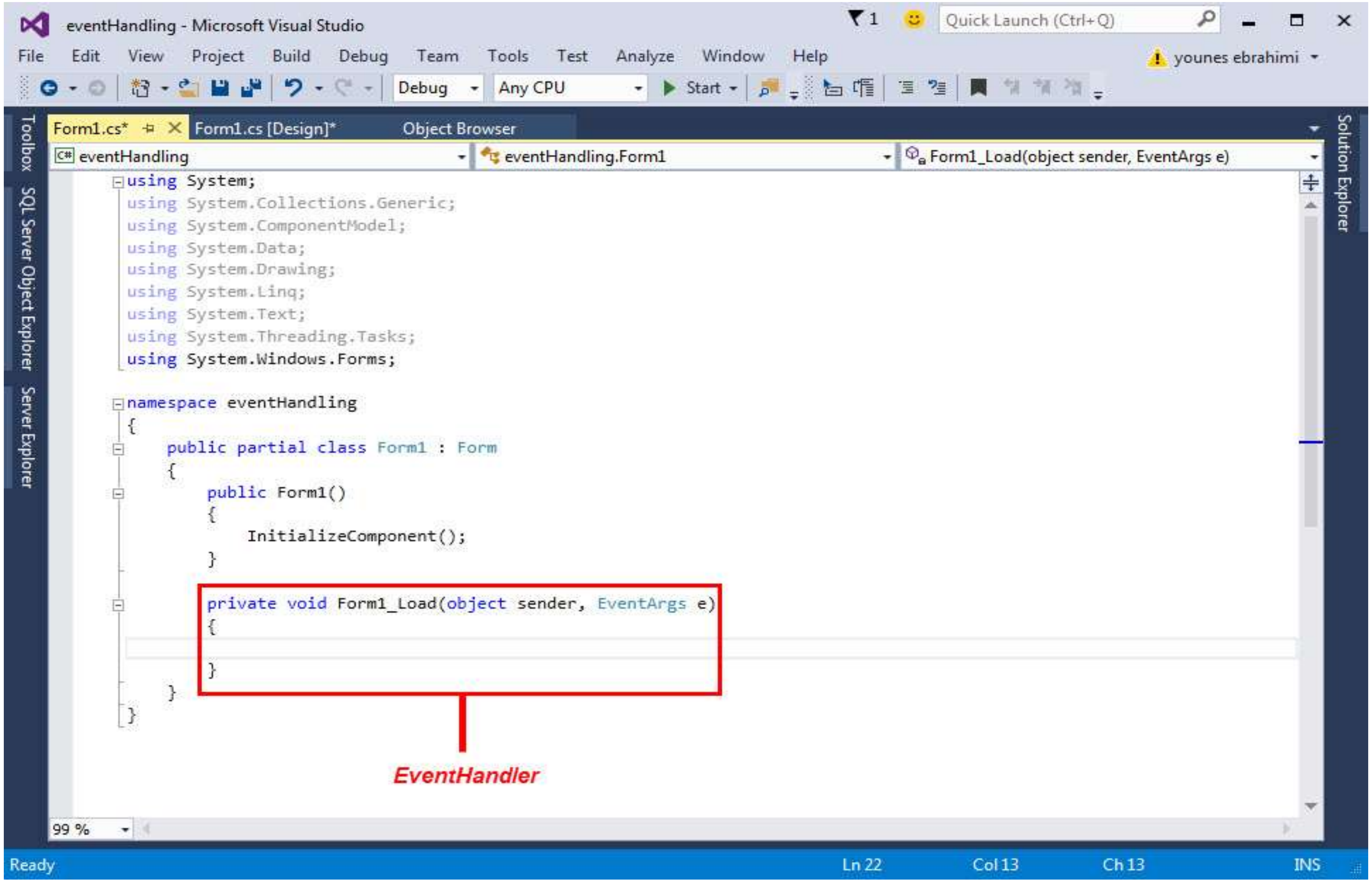
```
event MyDel MyEvent;
```

- The preceding code defines a delegate named MyDel and an event named MyEvent, which invokes the delegate when it is raised.

```
using System;
namespace SampleApp {
  public delegate string MyDel(string str);
  class EventProgram {
    event MyDel MyEvent;
    public EventProgram() {
          MyDel a= new MyDel(WelcomeUser);
          MyEvent =a;
          // this.MyEvent += new MyDel(this.WelcomeUser);
    }
    public string WelcomeUser(string username) {
      return "Welcome " + username;
    }
    static void Main(string[] args) {
      EventProgram obj1 = new EventProgram();
      string result = obj1.MyEvent("Tutorials Point");
      Console.WriteLine(result);
    }
  }
}
```

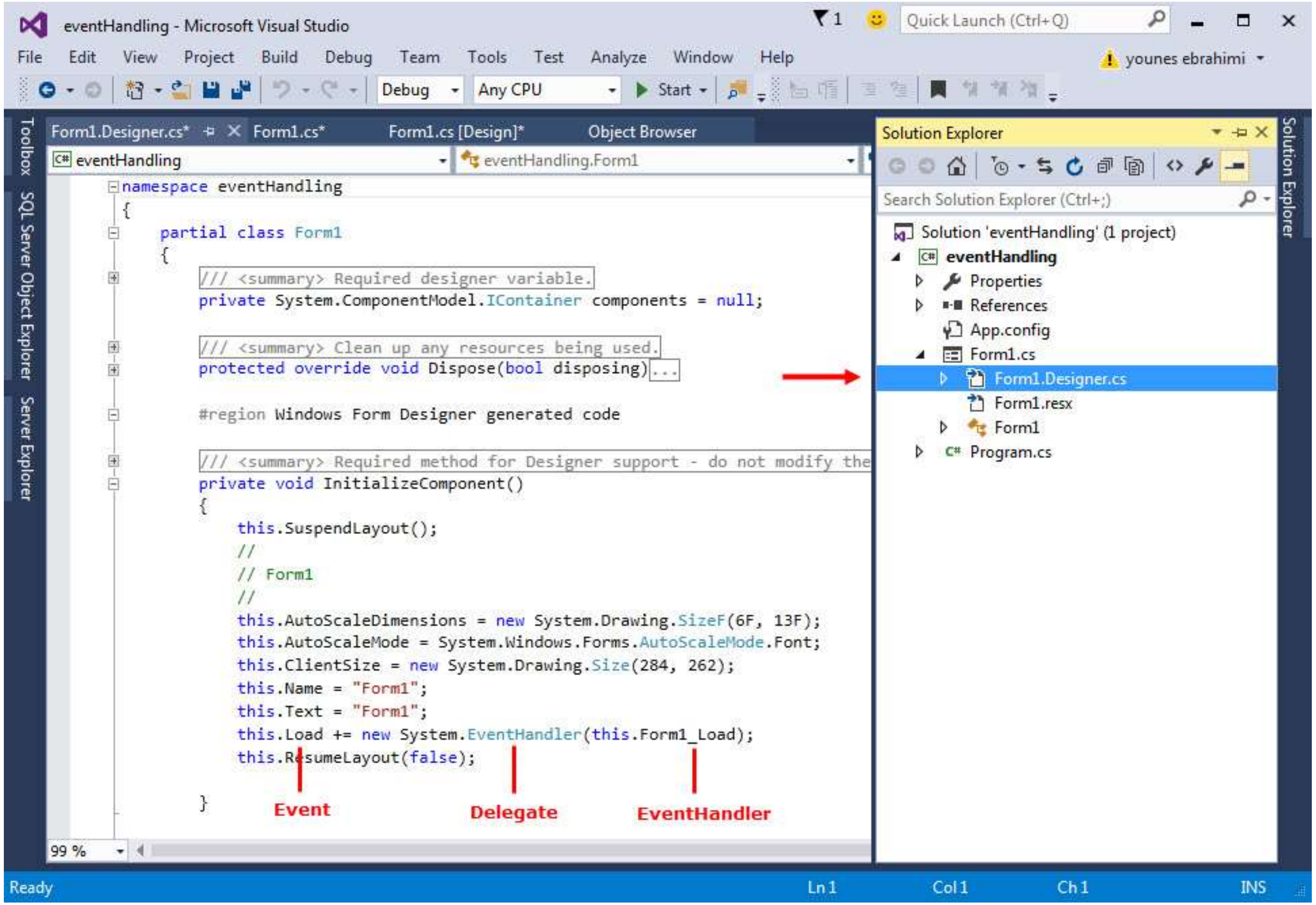
Events in Visual Programming

DoubleClick on form



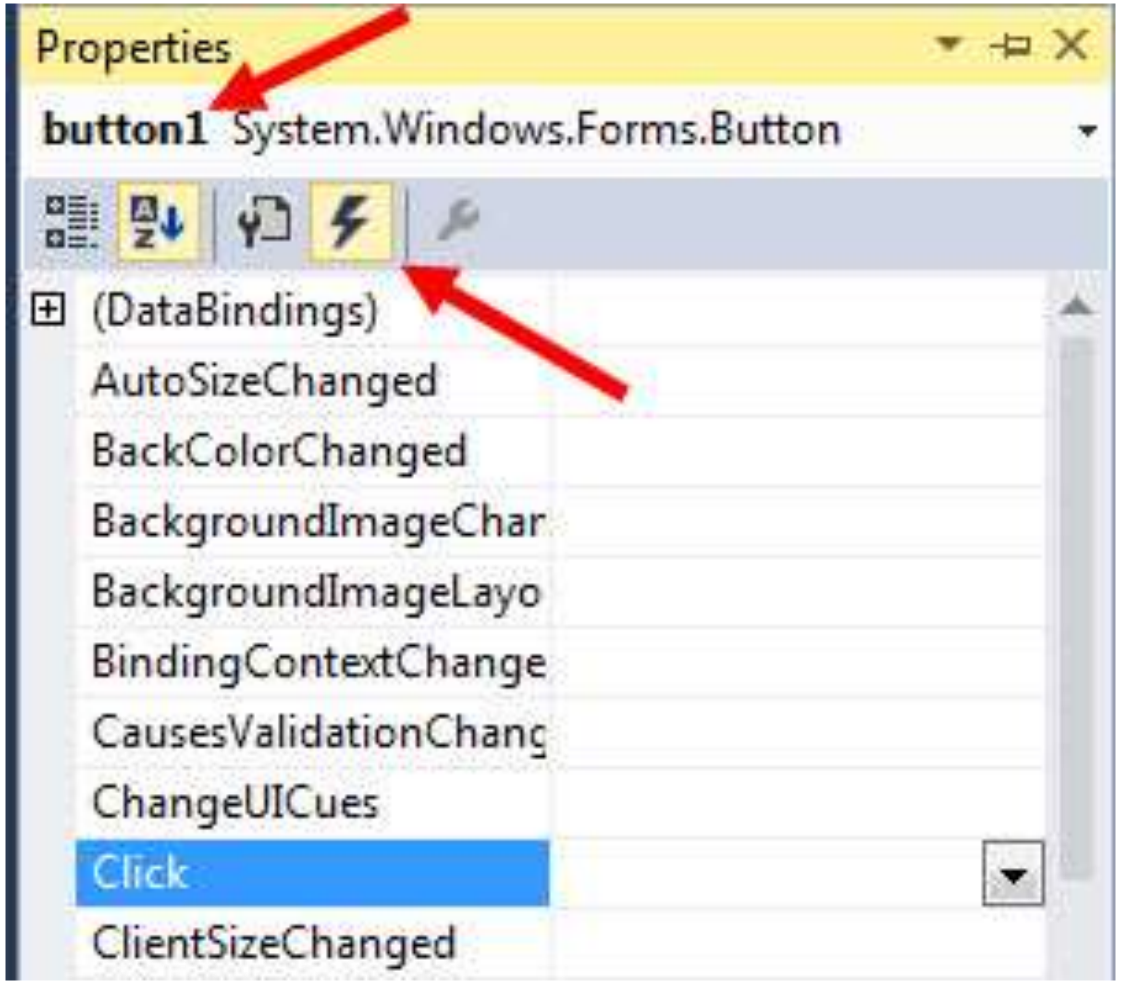
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace eventHandling
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
```

EventHandler



```
namespace eventHandling
{
    partial class Form1
    {
        /// <summary> Required designer variable.
        private System.ComponentModel.IContainer components = null;

        /// <summary> Clean up any resources being used.
        protected override void Dispose(bool disposing)...

        #region Windows Form Designer generated code

        /// <summary> Required method for Designer support - do not modify the
        private void InitializeComponent()
        {
            this.SuspendLayout();
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(284, 262);
            this.Name = "Form1";
            this.Text = "Form1";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.ResumeLayout(false);

        }
```

List of all Events



(DataBindings)	
AutoSizeChanged	
BackColorChanged	
BackgroundImageChar	
BackgroundImageLayo	
BindingContextChange	
CausesValidationChang	
ChangeUICues	
Click	
ClientSizeChanged	