How do I set a click event for a form?

Asked 16 years, 2 months ago Modified 11 years, 10 months ago

Viewed 30k times

8

I have a c# form (let's call it MainForm) with a number of custom controls on it. I'd like to have the MainForm.OnClick() method fire anytime someone clicks on the form regardless of whether the click happened on the form or if the click was on one of the custom controls. I'm looking for behavior similar to the KeyPreview feature of forms except for mouse clicks rather than key presses.

c# winforms

Share

Improve this question

Follow

edited Oct 7, 2008 at 21:13

GEOCHET

21.3k ● 15 ● 77 ● 99

asked Oct 7, 2008 at 21:12

Mykroft

13.4k ● 13 ● 48 ● 74

6 Answers

Sorted by: Highest score (default)

I recommend creating a base form for the other forms in your application to inherit. Add this code to your base

7

form to create a new event called GlobalMouseClickEventHandler:

```
namespace Temp
{
    public delegate void GlobalMouseClickEventHander(o
MouseEventArgs e);

    public partial class TestForm : Form
    {
        [Category("Action")]
        [Description("Fires when any control on the fo
        public event GlobalMouseClickEventHander Globa

        public TestForm()
        {
            InitializeComponent();
            BindControlMouseClicks(this);
        }

        private void BindControlMouseClicks(Control co
        {
            con.MouseClick += delegate(object sender,
            {
                TriggerMouseClicked(sender, e);
            };
            // bind to controls already added
            foreach (Control i in con.Controls)
            {
                BindControlMouseClicks(i);
            }
            // bind to controls added in the future
            con.ControlAdded += delegate(object sender
            {
                BindControlMouseClicks(e.Control);
            };
        }

        private void TriggerMouseClicked(object sender
        {
            if (GlobalMouseClick != null)
            {
```

```
                GlobalMouseClick(sender, e);
            }
        }
    }
}
```

This solution will work not only for top-level controls, but also nested controls such as controls placed inside of panels.

Share Improve this answer

Follow

answered Aug 16, 2012 at 15:18

ChéDon

141 ● 2 ● 3

> Thanks a ton! A simple and elegant solution. Wonder why there isn't something equivalent of `KeyPreview` for mouse clicks – Pavan Manjunath Feb 27, 2015 at 1:51

3

In the form's ControlAdded event, add a MouseClick handler to the control, with the Address of the form's click event. I haven't tested this, but it might work.

```
Private Sub Example_ControlAdded(ByVal sender As Objec
System.Windows.Forms.ControlEventArgs) Handles Me.Cont

    AddHandler e.Control.MouseClick, AddressOf Example
End Sub

Private Sub Example_MouseClick(ByVal sender As Object,
System.Windows.Forms.MouseEventArgs) Handles Me.MouseC
    MessageBox.Show("Click")
End Sub
```

Share Improve this answer

Follow

answered Oct 7, 2008 at 21:40

Burton

This still seems to be a little bit of a hack but this seems to be the best option I have. – Mykroft Oct 8, 2008 at 12:58

1

The only way I've ever managed to do this is to handle the [c]Click[/c] event of every control. I don't believe the event is raised before the control processes it.

In WPF, there are "tunneling" preview events that provide this functionality, but that doesn't really help you in WinForms.

Share Improve this answer

Follow

answered Oct 7, 2008 at 21:39

OwenP

25.4k ● 13 ● 70 ● 105

Well it's too late to switch to WPF now but I'll have to seriously consider using it in the future I guess. – Mykroft Oct 8, 2008 at 12:57

1

You can hook all the control's events, if you like, and then monitor that way. I assume there is some uber fancy Win32 api way to trap them all, but that is beyond me at the moment.

```
public Form1()
{
```

```
        InitializeComponent();
        HookEvents();
    }

    private void HookEvents() {
        foreach (Control ctl in this.Controls) {
            ctl.MouseClick += new MouseEventHandler(Fo
        }
    }

    void Form1_MouseClick(object sender, MouseEventArg
    {
        LogEvent(sender, "MouseClick");
    }

    // and then this just logs to a multiline textbox
form
    private void LogEvent(object sender, string msg) {
        this.textBox1.Text = string.Format("{0} {1} ({
            DateTime.Now.TimeOfDay.ToString(),
            msg,
            sender.GetType().Name,
            textBox1.Text
            );
    }
```

The output is something like this, showing all the events and who "sent" them up:

```
14:51:42.3381985 MouseClick (Form1)
14:51:40.6194485 MouseClick (RichTextBox)
14:51:40.0100735 MouseClick (TextBox)
14:51:39.6194485 MouseClick (Form1)
14:51:39.2131985 MouseClick (RichTextBox)
14:51:38.8694485 MouseClick (Button)
```

HTH.

Share Improve this answer answered Oct 7, 2008 at 21:57

Follow

Andrew
8,654 ● 3 ● 50 ● 73

0

Catching a click on an open space on the form is easy, but to get a click that's actually on a control, you'll need the cooperation of that control to send it to the form.

One possibility is to place a transparent control over the entire form, and accept clicks onto that, deal with them, and then pass them onto the proper control underneath.

Share Improve this answer

Follow

answered Oct 7, 2008 at 21:18

James Curran
103k ● 37 ● 185 ● 262

> I feel like there must be a more elegant solution than this.
> – Mykroft Oct 7, 2008 at 21:25

0

This is a common pattern in development, its called the Observer pattern. There are lots of examples of Observer patterns and c# here is 1 example http://msdn.microsoft.com/en-us/library/ms954621.aspx but have a good google around.

Share Improve this answer

Follow

answered Oct 7, 2008 at 21:32

Tim Jarvis
18.8k ● 10 ● 59 ● 95