

Forum:[Hack Forums](#)

Thread: [Catch form closing event](#)

AP?X - December 20, 2018, 6:11 am

Hi HF,

I'm trying to catch my dashboard.vb form when closing so I can close my parent form (the login system) after doing a bit of research I found that you can't really close the parent form from the child form so what I decided is to just hide the parent form after a successful login and show the child form. The problem I'm having is I'm trying to close the parent form (its still running in the background of the computer but is hidden) by catching when the child form is being exited by the user.

Currently, I'm using this code but have no result when closing the child form:

```
Code      private void Dashboard_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
    MessageBox.Show("WAIT!");
    //Login.Close();
}
```

If someone could help me that would be great, or even better suggest a better way to handle all of this, I'm used to VB.NET where parent and child forms don't really exist (from what I know) just regular forms.

itcorp - December 20, 2018, 7:08 am

(Dec 20, 2018 07:11 AM)AP?X Wrote: [\[CITE\]](#) Hi HF,

I'm trying to catch my dashboard.vb form when closing so I can close my parent form (the login system) after doing a bit of research I found that you can't really close the parent form from the child form so what I decided is to just hide the parent form after a successful login and show the child form. The problem I'm having is I'm trying to close the parent form (its still running in the background of the computer but is hidden) by catching when the child form is being exited by the user.

Currently, I'm using this code but have no result when closing the child form:

```
Code      private void Dashboard_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
    MessageBox.Show("WAIT!");
    //Login.Close();
}
```

If someone could help me that would be great, or even better suggest a better way to handle all of this, I'm used to VB.NET where parent and child forms don't really exist (from what I know) just regular forms.

You can do something like this

```
Code private static Dashboard staticInstance = null;
```

```
public Dashboard()
{
    InitializeComponent();
}
```

```
Closed += Dashboard_Closed;
}
```

```
private void Dashboard_Closed(object sender, EventArgs e)
{
    staticInstance = null;
}
```

```
public static Dashboard GetInstance()
{
    if (staticInstance == null)
    {
        staticInstance = new Dashboard();
    }

    return staticInstance;
}
usage
```

```
Code Dashboard.GetInstance().Closing += (b, a) =>
{

};
```

Dashboard.GetInstance().ShowDialog(); You can also set up in application ShutdownMode="OnLastWindowClose" and in app.cs define the login form to open first on start.

HFDone - December 20, 2018, 7:48 am

note that closing event will work just if you close the form with the x button

n0\$f3ratu\$ - December 20, 2018, 12:36 pm

You can use a separate thread.

It will get the job done.

```
Code private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    Thread thread2 = new Thread(new ParameterizedThreadStart(Form2));
    thread2.SetApartmentState(ApartmentState.STA);
    thread2.Start();
}

public static void Form2(object obj)
{
    Application.Run(new Form2());
}
```

BahNahNah - December 20, 2018, 7:52 pm

On the parent form, use ShowDialog insted of .Show()

Code this.Hide(); //Hide "parent" form

```
using(var form2 = new Form2()){
    form2.ShowDialog(); //Show child form and wait for close
}
```

Application.Exit(); //Terminate application
Or a better way in my opinion, in the main form constructor:

```
Code public MainWindow(){
    using(var frm = new LoginForm()){
        if(frm.ShowDialog() != DialogResult.OK){
            Application.Exit();
        }
        //Get needed info from the login form
    }
    InitiliseComponents();
}
or even better, in the Program.cs (or vb equivalent)
Code public static void Main(string[] arg){
    using(var frm = new LoginForm()){
        if(frm.ShowDialog() != DialogResult.OK){
            Application.Exit();
        }
    }
    Application.Run(new MainWindow());
}
```

Sorry if the code is botched not in windows atm.

gigajew - December 20, 2018, 7:58 pm

Use a controller class that controls the appearance and disappearance of forms. Inherit from application context in the controller class and listen to the FormClosed or FormClosing event to decide what comes next

SlowMehMo - December 22, 2018, 3:23 am

When using WPF, I usually use shellview but

I like to create custom forms and include my own buttons, etc

if your able and want to make a stunning user experience then i would suggest making some controls in C# and including that.

AP?X - December 22, 2018, 8:36 am

Resolved, thanks @everyone who helped!

//Closed