

Masked TextBox Host



```
/*
User Interfaces in C#: Windows Forms and Custom Controls
by Matthew MacDonald

Publisher: Apress
ISBN: 1590590457
*/

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace MaskedTextBoxHost
{
    ///
    /// Summary description for MaskedTextBoxHost.
    ///
    public class MaskedTextBoxHost : System.Windows.Forms.Form
    {
        ///
        /// Required designer variable.
        ///
        private System.ComponentModel.Container components = null;

        public MaskedTextBoxHost()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();
        }
    }
}
```

```

//
// TODO: Add any constructor code after InitializeComponent
call
//
}

///

/// Clean up any resources being used.
///
protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

///

/// Required method for Designer support – do not modify
/// the contents of this method with the code editor.
///
private void InitializeComponent()
{
//
// MaskedTextBoxHost
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(292, 266);
this.Name = "MaskedTextBoxHost";
this.Text = "MaskedTextBoxHost";
this.Load += new
System.EventHandler(this.MaskedTextBoxHost_Load);

```

```

}

///  

///  

///  

///  

///  

[STAThread]  

static void Main()  

{  

Application.Run(new MaskedTextBoxHost());  

}  

  

private void MaskedTextBoxHost_Load(object sender,  

System.EventArgs e)  

{  

MaskedTextBox txtMask = new  

MaskedTextBox();  

txtMask.Location = new Point(10, 10);  

txtMask.Mask = "(###) ###-####";  

this.Controls.Add(txtMask);  

  

}  

}  

public class MaskedTextBox : TextBox  

{  

private string mask;  

public string Mask  

{  

get  

{  

return mask;  

}  

set  

{  

mask = value;  

this.Text = "";  

}  

}  

}

```

```
protected override void OnKeyPress(KeyPressEventArgs e)
{
if (Mask != "")
{
// Suppress the typed character.
e.Handled = true;

string newText = this.Text;

// Loop through the mask, adding fixed characters as needed.
// If the next allowed character matches what the user has
// typed in (a number or letter), that is added to the end.
bool finished = false;
for (int i = this.SelectionStart; i < mask.Length; i++) {
switch (mask[i].ToString()) { case "#" : // Allow the keypress
as long as it is a number. if (Char.IsDigit(e.KeyChar)) {
newText += e.KeyChar.ToString(); finished = true; break; }
else { // Invalid entry; exit and don't change the text.
return; } case "." : // Allow the keypress as long as it is a
letter. if (Char.IsLetter(e.KeyChar)) { newText +=
e.KeyChar.ToString(); finished = true; break; } else { //
Invalid entry; exit and don't change the text. return; }
default : // Insert the mask character. newText += mask[i];
break; } if (finished) { break; } } // Update the text.
this.Text = newText; this.SelectionStart = this.Text.Length; }
} protected override void OnKeyDown(KeyEventArgs e) { // Stop
special characters. e.Handled = true; } } } [/csharp]
```