PowerShell 6

Version

6

5.1

5.0

4.0

3.0

Creating a Custom Input Box

06/05/2017 3 minutes to read Contributors

In this article

Script a graphical custom input box by using Microsoft .NET Framework form-building features in Windows PowerShell 3.0 and later releases.

Create a custom, graphical input box

Copy and then paste the following into Windows PowerShell ISE, and then save it as a Windows PowerShell script (.ps1).

PowerShell Copy

```
Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing

$form = New-Object System.Windows.Forms.Form
$form.Text = 'Data Entry Form'
$form.Size = New-Object System.Drawing.Size(300,200)
$form.StartPosition = 'CenterScreen'

$OKButton = New-Object System.Windows.Forms.Button
$OKButton.Location = New-Object System.Drawing.Point(75,120)
$OKButton.Size = New-Object System.Drawing.Size(75,23)
$OKButton.Text = 'OK'
$OKButton.DialogResult = [System.Windows.Forms.DialogResult]::OK
$form.AcceptButton = $OKButton
$form.Controls.Add($OKButton)

$CancelButton = New-Object System.Windows.Forms.Button
$CancelButton.Location = New-Object System.Drawing.Point(150,120)
$CancelButton.Size = New-Object System.Drawing.Size(75,23)
$CancelButton.Text = 'Cancel'
```

```
$CancelButton.DialogResult = [System.Windows.Forms.DialogResult]::Cancel
$form.CancelButton = $CancelButton
$form.Controls.Add($CancelButton)

$label = New-Object System.Windows.Forms.Label
$label.Location = New-Object System.Drawing.Point(10,20)
$label.Size = New-Object System.Drawing.Size(280,20)
$label.Text = 'Please enter the information in the space below:'
$form.Controls.Add($label)

$textBox = New-Object System.Windows.Forms.TextBox
$textBox.Location = New-Object System.Drawing.Point(10,40)
$textBox.Size = New-Object System.Drawing.Size(260,20)
$form.Controls.Add($textBox)

$form.Topmost = $true

$form.Add_Shown({$textBox.Select()})
$result = $form.ShowDialog()

if ($result -eq [System.Windows.Forms.DialogResult]::OK)
{
    $x = $textBox.Text
    $x
}
```

The script begins by loading two .NET Framework classes: **System.Drawing** and **System.Windows.Forms**. You then start a new instance of the .NET Framework class **System.Windows.Forms.Form**; that provides a blank form or window to which you can start adding controls.

PowerShell Copy

```
$form = New-Object System.Windows.Forms.Form
```

After you create an instance of the Form class, assign values to three properties of this class.

- **Text.** This becomes the title of the window.
- **Size.** This is the size of the form, in pixels. The preceding script creates a form that's 300 pixels wide by 200 pixels tall.
- **StartingPosition.** This optional property is set to **CenterScreen** in the preceding script. If you don't add this property, Windows selects a location when the form is opened. By setting the **StartingPosition** to **CenterScreen**, you're automatically displaying the form in the middle of the screen each time it loads.

PowerShell Copy

```
$form.Text = 'Data Entry Form'
$form.Size = New-Object System.Drawing.Size(300,200)
$form.StartPosition = 'CenterScreen'
```

Next, create an **OK** button for your form. Specify the size and behavior of the **OK** button. In this example, the button position is 120 pixels from the form's top edge, and 75 pixels from the left edge. The button height is 23 pixels, while the button length is 75 pixels. The script uses predefined Windows Forms types to determine the button behaviors.

PowerShell Copy

```
$OKButton = New-Object System.Windows.Forms.Button
$OKButton.Location = New-Object System.Drawing.Point(75,120)
$OKButton.Size = New-Object System.Drawing.Size(75,23)
$OKButton.Text = 'OK'
$OKButton.DialogResult = [System.Windows.Forms.DialogResult]::OK
$form.AcceptButton = $OKButton
$form.Controls.Add($OKButton)
```

Similarly, you create a **Cancel** button. The **Cancel** button is 120 pixels from the top, but 150 pixels from the left edge of the window.

PowerShell Copy

```
$CancelButton = New-Object System.Windows.Forms.Button
$CancelButton.Location = New-Object System.Drawing.Point(150,120)
$CancelButton.Size = New-Object System.Drawing.Size(75,23)
$CancelButton.Text = 'Cancel'
$CancelButton.DialogResult = [System.Windows.Forms.DialogResult]::Cancel
$form.CancelButton = $CancelButton
$form.Controls.Add($CancelButton)
```

Next, provide label text on your window that describes the information you want users to provide.

PowerShell Copy

```
$label = New-Object System.Windows.Forms.Label
$label.Location = New-Object System.Drawing.Point(10,20)
$label.Size = New-Object System.Drawing.Size(280,20)
$label.Text = 'Please enter the information in the space below:'
$form.Controls.Add($label)
```

Add the control (in this case, a text box) that lets users provide the information you've described in your label text. There are many other controls you can apply besides text boxes; for more controls, see System.Windows.Forms Namespace on MSDN.

PowerShell Copy

```
$textBox = New-Object System.Windows.Forms.TextBox
$textBox.Location = New-Object System.Drawing.Point(10,40)
$textBox.Size = New-Object System.Drawing.Size(260,20)
$form.Controls.Add($textBox)
```

Set the **Topmost** property to **$true** to force the window to open atop other open windows and dialog boxes.

PowerShell Copy

```
$form.Topmost = $true
```

Next, add this line of code to activate the form, and set the focus to the text box that you created.

PowerShell Copy

```
$form.Add_Shown({$textBox.Select()})
```

Add the following line of code to display the form in Windows.

PowerShell Copy

```
$result = $form.ShowDialog()
```

Finally, the code inside the **If** block instructs Windows what to do with the form after users provide text in the text box, and then click the **OK** button or press the **Enter** key.

PowerShell Copy

```
if ($result -eq [System.Windows.Forms.DialogResult]::OK)
{
    $x = $textBox.Text
    $x
}
```

See Also

- Hey Scripting Guy: Why don't these PowerShell GUI examples work?
- GitHub: Dave Wyatt's WinFormsExampleUpdates
- Windows PowerShell Tip of the Week: Creating a Custom Input Box