

Panel

Ein `Panel` ist ein einfacher Container für andere Controls. Es dient dazu, mehrere Controls logisch und visuell zu einer Einheit zusammenzufassen.

Im Gegensatz zu einem `TableLayoutPanel` besitzt ein `Panel` kein eigenes Raster oder automatisches Layoutsystem. Die enthaltenen Controls werden grundsätzlich über ihre `Location`- und `Size`-Eigenschaften positioniert und dimensioniert.

Ein Panel kann selbst wiederum in anderen Containern liegen, wodurch sich komplexe Benutzeroberflächen aus mehreren verschachtelten Bereichen aufbauen lassen.

Grundlagen

Das `Panel` ist ein **Container für Controls**.

```
Panel
├─ Label
├─ TextBox
├─ Button
└─ CheckBox
```

Vergleich mit anderen Container-Controls:

- **Panel** → freie Positionierung über `Location`
- **FlowLayoutPanel** → automatische Anordnung hintereinander
- **TableLayoutPanel** → Anordnung in Zeilen und Spalten

Panel erstellen

```
# Klassisch
$panel = New-Object System.Windows.Forms.Panel

# .NET-Style
$panel = [System.Windows.Forms.Panel]::new()
```

Panel hinzufügen

Ein `Panel` wird wie jedes andere Control der `Controls`-Collection seines Parent-Containers hinzugefügt.

```
$form.Controls.Add($panel)
```

oder

```
$tabPage.Controls.Add($panel)
```

Controls hinzufügen

Controls werden über die `Controls`-Collection des Panels hinzugefügt.

```
$panel.Controls.Add($button)
```

```
$panel.Controls.Add($label)
```

```
$panel.Controls.Add($textBox)
```

Die Position der Controls wird anschließend über deren `Location` festgelegt.

```
$button.Location = "10, 10"
```

```
$label.Location = "10, 50"
```

```
$textBox.Location = "100, 50"
```

Controls entfernen

Ein Control kann über seine Referenz entfernt werden

```
$panel.Controls.Remove($button)
```

oder mit seinen Index in der `Controls`-Collection

```
$panel.Controls.RemoveAt(0)
```

Mit `Clear()` werden alle enthaltenen Controls entfernt.

```
$panel.Controls.Clear()
```

Eigenschaften

Eigenschaft	Beschreibung
`AutoScroll`	Aktiviert automatisch Scrollleisten, wenn der Inhalt größer als das Panel ist.
`AutoSize`	Passt die Größe automatisch an den Inhalt an.
`AutoSizeMode`	Bestimmt, in welche Richtung sich das Panel bei `AutoSize` anpassen darf.
`BackColor`	Legt die Hintergrundfarbe des Panels fest.
`BorderStyle`	Bestimmt, ob und wie das Panel einen Rahmen darstellt.
`Dock`	Dockt das Panel an einer Seite seines Parent-Containers an.
`Anchor`	Verankert das Panel an den Rändern seines Parent-Containers.
`Padding`	Legt den Innenabstand zwischen Panelrand und enthaltenen Controls fest.
`Margin`	Legt den äußeren Abstand des Panels zu anderen Controls fest.
`Controls`	Enthält alle Controls, die sich innerhalb des Panels befinden.
`Location`	Bestimmt die Position des Panels im Parent-Container.
`Size`	Bestimmt Breite und Höhe des Panels.
`MinimumSize`	Definiert die minimal zulässige Größe des Panels.
`MaximumSize`	Definiert die maximal zulässige Größe des Panels.
`Name`	Legt den internen Namen des Panels fest.
`Visible`	Legt fest, ob das Panel sichtbar ist.
`Enabled`	Legt fest, ob das Panel und seine enthaltenen Controls aktiviert sind.

AutoScroll

Typ = [System.Boolean]

Die Eigenschaft `AutoScroll` legt fest, ob das Panel automatisch Scrollleisten anzeigt, wenn sein Inhalt größer als der sichtbare Bereich ist.

Standardmäßig ist `AutoScroll` auf `False` gesetzt.

Wird die Eigenschaft auf `True` gesetzt, erscheinen horizontale und/oder vertikale Scrollleisten automatisch, sobald der Inhalt nicht mehr vollständig in das Panel passt.

```
$panel.AutoScroll = $true
```

AutoSize

Typ = `[System.Boolean]`

Die Eigenschaft `AutoSize` legt fest, ob das Panel seine Größe automatisch an seinen Inhalt anpasst.

Standardmäßig besitzt diese Eigenschaft den Wert `False`.

Ist `AutoSize` aktiviert, wird die benötigte Größe des Panels anhand seiner enthaltenen Controls und deren Layoutinformationen bestimmt.

```
$panel.AutoSize = $true
```

AutoSizeMode

Typ = `[System.Windows.Forms.AutoSizeMode]`

Die Eigenschaft `AutoSizeMode` bestimmt, wie sich das Panel bei aktiviertem `AutoSize` an seinen Inhalt anpasst.

Es stehen zwei Werte zur Verfügung:

- `GrowAndShrink` → Das Panel kann seine Größe sowohl vergrößern als auch verkleinern.
- `GrowOnly` → Das Panel kann nur größer werden, aber nicht automatisch kleiner.

Standardmäßig ist `GrowOnly` eingestellt.

```
$panel.AutoSize = $true  
$panel.AutoSizeMode = "GrowAndShrink"
```

BackColor

Typ = [System.Drawing.Color]

Die Eigenschaft `BackColor` legt die Hintergrundfarbe des Panels fest.

```
$panel.BackColor = "LightBlue"
```

Die Hintergrundfarbe betrifft ausschließlich die vom Panel selbst dargestellte Fläche. Die enthaltenen Controls behalten grundsätzlich ihre eigene Darstellung.

BorderStyle

Typ = [System.Windows.Forms.BorderStyle]

Die Eigenschaft `BorderStyle` bestimmt, ob das Panel einen Rahmen darstellt.

Mögliche Werte:

- `None` → kein Rahmen
- `FixedSingle` → einfacher Rahmen
- `Fixed3D` → dreidimensionaler Rahmen

Standardmäßig besitzt `BorderStyle` den Wert `None`.

```
$panel.BorderStyle = "FixedSingle"
```

Der Rahmen dient hauptsächlich der visuellen Abgrenzung des Panels und verändert nicht die grundlegende Funktion des Containers.

Controls

Typ = [System.Windows.Forms.Control+ControlCollection]

Die Eigenschaft `Controls` enthält alle Controls, die direkt innerhalb des Panels liegen.

Über diese Collection können Controls hinzugefügt, entfernt oder abgerufen werden.

```
$panel.Controls.Add($button)
```

```
$panel.Controls.Remove($button)
```

```
$panel.Controls.Clear()
```

Ein Control kann über seinen Index abgerufen werden:

```
$panel.Controls[0]
```

Die Collection enthält dabei nur die **direkt** im Panel enthaltenen Controls. Controls, die wiederum innerhalb eines anderen Panels liegen, gehören nicht direkt zu dieser Collection.

Dock

Typ = [System.Windows.Forms.DockStyle]

Die Eigenschaft `Dock` legt fest, wie das Panel innerhalb seines Parent-Containers angedockt wird.

Standardmäßig besitzt `Dock` den Wert `None`.

Alternativ kann das Panel an einer Seite angedockt werden:

- `Top`
- `Bottom`
- `Left`
- `Right`
- `Fill`

Mit `Fill` nimmt das Panel den gesamten verfügbaren Bereich des Parent-Containers ein.

```
$panel.Dock = "Fill"
```

Dies ist besonders praktisch, wenn ein Panel als Arbeitsbereich innerhalb eines `Form`, `TabPage` oder eines anderen Containers verwendet wird.

Anchor

Typ = [System.Windows.Forms.AnchorStyles]

Die Eigenschaft `Anchor` legt fest, an welchen Rändern des Parent-Containers das Panel verankert bleibt.

Standardmäßig ist das Panel an `Top` und `Left` verankert.

```
$panel.Anchor = "Top, Left, Right"
```

Wird ein Panel beispielsweise an `Left` und `Right` verankert, passt sich seine Breite an die Größe des Parent-Containers an.

Im Gegensatz zu `Dock` behält `Anchor` die Abstände zu den angegebenen Rändern bei.

Padding

Typ = [System.Windows.Forms.Padding]

Die Eigenschaft `Padding` legt den Innenabstand zwischen dem Rand des Panels und seinen enthaltenen Controls fest.

```
$panel.Padding = 10
```

Dadurch beginnen enthaltene Controls nicht direkt am Rand des Panels.

Der Unterschied zu `Margin`:

Margin → Abstand außerhalb des Panels

Padding → Abstand innerhalb des Panels

Margin

Typ = [System.Windows.Forms.Padding]

Die Eigenschaft `Margin` legt den äußeren Abstand des Panels zu anderen Controls fest.

```
$panel.Margin = 10
```

Die Eigenschaft wird insbesondere von Layout-Containern wie `FlowLayoutPanel` und `TableLayoutPanel` berücksichtigt.

Location

Typ = `[System.Drawing.Point]`

Die Eigenschaft `Location` bestimmt die Position des Panels innerhalb seines Parent-Containers.

Die Position wird über die X- und Y-Koordinate angegeben.

```
$panel.Location = "20, 40"
```

Da ein normales Panel kein eigenes Layoutsystem besitzt, ist `Location` insbesondere für die Positionierung der enthaltenen Controls relevant.

```
$button.Location = "10, 10"  
$label.Location = "10, 50"
```

Size

Typ = `[System.Drawing.Size]`

Die Eigenschaft `Size` bestimmt die Breite und Höhe des Panels.

```
$panel.Size = "300, 200"
```

Die Größe kann auch durch `Dock`, `Anchor` oder `AutoSize` beeinflusst werden.

Visible

Typ = `[System.Boolean]`

Die Eigenschaft `Visible` legt fest, ob das Panel sichtbar dargestellt wird.

```
$panel.Visible = $false
```

Wird das Panel ausgeblendet, werden auch seine enthaltenen Controls nicht sichtbar dargestellt.

Enabled

Typ = `[System.Boolean]`

Die Eigenschaft `Enabled` legt fest, ob das Panel aktiviert ist.

```
$panel.Enabled = $false
```

Wird ein Panel deaktiviert, können auch seine enthaltenen Controls nicht mehr über die Benutzeroberfläche bedient werden.

Methoden

Methode	Beschreibung
<code>BringToFront()</code>	Bringt das Panel innerhalb seines Parent-Containers in den Vordergrund
<code>SendToBack()</code>	Verschiebt das Panel innerhalb seines Parent-Containers in den Hintergrund
<code>ScrollControlIntoView()</code>	Scrollt das angegebene Control in den sichtbaren Bereich
<code>PerformLayout()</code>	Erzwingt eine erneute Layoutberechnung des Panels
<code>Refresh()</code>	Aktualisiert die Darstellung des Panels

BringToFront()

Die Methode `BringToFront()` bringt das Panel innerhalb seines Parent-Containers in den Vordergrund.

Dies beeinflusst die Z-Reihenfolge der Controls.

```
$panel.BringToFront()
```

Befinden sich mehrere Controls übereinander, kann dadurch festgelegt werden, welches Control sichtbar über den anderen liegt.

SendToBack()

Die Methode `SendToBack()` verschiebt das Panel innerhalb seines Parent-Containers in den Hintergrund.

```
$panel.SendToBack()
```

Dadurch können andere Controls, die sich an derselben Position befinden, vor dem Panel dargestellt werden.

ScrollControlIntoView()

Die Methode `ScrollControlIntoView()` sorgt dafür, dass ein angegebenes Control innerhalb des Panels sichtbar wird.

Sie ist insbesondere relevant, wenn `AutoScroll` aktiviert ist.

```
$panel.AutoScroll = $true  
  
$panel.ScrollControlIntoView($button)
```

Befindet sich das angegebene Control außerhalb des momentan sichtbaren Bereichs, wird das Panel entsprechend gescrollt.

PerformLayout()

Die Methode `PerformLayout()` erzwingt eine sofortige Neuberechnung des Layouts.

```
$panel.PerformLayout()
```

Dies kann nützlich sein, wenn mehrere Eigenschaften oder Controls verändert wurden und die Layoutberechnung anschließend unmittelbar durchgeführt werden soll.

Refresh()

Die Methode `Refresh()` aktualisiert die Darstellung des Panels.

```
$panel.Refresh()
```

Dabei wird das Panel neu gezeichnet.

Events

Übersicht

Event	Beschreibung
`ControlAdded`	Wird ausgelöst, wenn ein Control zum Panel hinzugefügt wird.
`ControlRemoved`	Wird ausgelöst, wenn ein Control aus dem Panel entfernt wird.
`Layout`	Wird ausgelöst, wenn das Layout des Panels neu berechnet wird.
`Paint`	Wird ausgelöst, wenn das Panel gezeichnet bzw. neu gezeichnet wird.
`Resize`	Wird ausgelöst, wenn sich die Größe des Panels ändert.
`Enter`	Wird ausgelöst, wenn das Panel bzw. ein darin enthaltener Fokusbereich betreten wird.
`Leave`	Wird ausgelöst, wenn das Panel bzw. ein darin enthaltener Fokusbereich verlassen wird.

ControlAdded

Wird ausgelöst, sobald ein Control zum Panel hinzugefügt wird.

```
$panel.Add_ControlAdded({  
    param($sender, $e)  
  
    Write-Host "Control hinzugefügt: $($e.Control.Name)"  
})
```

ControlRemoved

Wird ausgelöst, sobald ein Control aus dem Panel entfernt wird.

```
$panel.Add_ControlRemoved({  
    param($sender, $e)  
  
    Write-Host "Control entfernt: $($e.Control.Name)"  
})
```

Layout

Das Event wird ausgelöst, wenn das Panel sein Layout neu berechnet.

Dies kann beispielsweise durch folgende Änderungen geschehen:

- Größe des Panels geändert
- Controls hinzugefügt oder entfernt
- Größe eines enthaltenen Controls geändert
- Änderungen an `Dock`
- Änderungen an `Anchor`
- Änderungen an `Padding`

```
$panel.Add_Layout({  
    param($sender, $e)  
  
    Write-Host "Layout aktualisiert"  
})
```

Resize

Wird ausgelöst, wenn sich die Größe des Panels ändert.

```
$panel.Add_Resize({  
    param($sender, $e)  
  
    Write-Host "Panel-Größe geändert"  
})
```

Paint

Wird ausgelöst, wenn das Panel neu gezeichnet wird.

```
$panel.Add_Paint({  
    param($sender, $e)  
  
    # Eigene Zeichenlogik  
})
```

Typische Stolperfallen

- **Control ist nicht sichtbar**
 - Das Control wurde nicht zur `Controls`-Collection des Panels hinzugefügt.
- **Control sitzt an der falschen Position**
 - `Location` des Controls überprüfen.
- **Panel passt sich nicht an den Parent an**
 - `Dock` oder `Anchor` überprüfen.
- **Panel wächst nicht mit seinem Inhalt**
 - `AutoSize` aktivieren.
- **Inhalt ist außerhalb des sichtbaren Bereichs nicht erreichbar**
 - `AutoScroll` aktivieren.
- **Controls liegen nicht mit dem gewünschten Abstand am Rand**
 - `Padding` des Panels überprüfen.
- **Panel liegt hinter einem anderen Control**
 - `BringToFront()` oder `SendToBack()` verwenden.

Mentales Modell

Das `Panel` ist ein **Container ohne eigenes Layoutsystem**.

Es stellt hauptsächlich einen abgegrenzten Bereich bereit, in dem andere Controls platziert werden können.

```
Parent
|
└─ Panel
    ├── Label
    ├── TextBox
    └─ Button
```

Die Positionierung der enthaltenen Controls erfolgt grundsätzlich über deren eigene Eigenschaften:

```
Panel
|
└─ Location
└─ Size
└─ Controls
    |
    ├── Button.Location
    ├── Label.Location
    └─ TextBox.Location
```

Damit unterscheidet sich das Panel grundlegend von einem `TableLayoutPanel`: Das Panel **ordnet nichts automatisch an**, sondern stellt lediglich den Container bereit. Das entspricht auch dem von dir bereits verwendeten Vergleich in der `TableLayoutPanel`-Doku.

Wann sinnvoll?

Ein `Panel` eignet sich besonders für:

- Gruppierung zusammengehöriger Controls
- Aufbau mehrteiliger Benutzeroberflächen
- Verschachtelung von UI-Bereichen
- Bereiche, die gemeinsam ein- oder ausgeblendet werden sollen
- eigene Layoutlogik über `Location` und `Size`
- scrollbare Inhaltsbereiche

Wann anderes Control verwenden?

Ein anderes Container-Control ist sinnvoller, wenn die Positionierung automatisch erfolgen soll:

- `FlowLayoutPanel` → Controls automatisch hintereinander anordnen
- `TableLayoutPanel` → Controls in Zeilen und Spalten anordnen
- `TabControl` → zwischen mehreren Inhaltsbereichen umschalten

Das ist eigentlich die wichtigste Erkenntnis der ganzen Seite: **Panel ist der einfache Container.** Es macht nicht heimlich Layoutmagie im Hintergrund. Und genau deshalb ist es oft das angenehmste Control von allen. Welch seltene Ausnahme in der Windows-Forms-Welt.

Revision #4

Created 2026-08-11 13:57:03 UTC by John-Andreas Borinas

Updated 2026-08-25 15:30:16 UTC by John-Andreas Borinas