

FlowLayoutPanel

Ein `FlowLayoutPanel` ist ein Layout-Container, der seine enthaltenen Controls automatisch hintereinander anordnet.

Im Gegensatz zu einem normalen `Panel` müssen die enthaltenen Controls nicht über ihre `Location` positioniert werden. Das `FlowLayoutPanel` übernimmt die Anordnung anhand der festgelegten Flussrichtung.

Wird der verfügbare Platz überschritten, können die Controls automatisch in eine neue Zeile beziehungsweise Spalte umgebrochen werden.

`Panel`

→ freie Positionierung über `Location`

`FlowLayoutPanel`

→ automatische Anordnung hintereinander

`TableLayoutPanel`

→ Anordnung in Zeilen und Spalten

Grundlagen

Ein `FlowLayoutPanel` eignet sich besonders für Benutzeroberflächen, bei denen Controls dynamisch nebeneinander oder untereinander angeordnet werden sollen.

Typische Einsatzgebiete sind beispielsweise:

- Button-Leisten
- dynamische Listen von Controls
- Werkzeugleisten
- Filter- und Einstellungsbereiche
- Karten- oder Kachelansichten
- automatisch umbrechende Control-Gruppen

FlowLayoutPanel erstellen

```
# Klassisch
$flow = New-Object System.Windows.Forms.FlowLayoutPanel

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

Controls hinzufügen

Controls werden wie bei anderen Container-Controls über die `Controls`-Collection hinzugefügt.

```
$flow.Controls.Add($button)
$flow.Controls.Add($label)
$flow.Controls.Add($textBox)
```

Die Position der Controls wird anschließend automatisch durch das Layoutsystem bestimmt.

Flussrichtung

Standardmäßig werden Controls von links nach rechts angeordnet.

```
$flow.FlowDirection = "LeftToRight"
```

Alternativ kann die Anordnung beispielsweise von oben nach unten erfolgen:

```
$flow.FlowDirection = "TopDown"
```

Automatischer Umbruch

Ist `WrapContents` aktiviert, werden Controls automatisch in eine neue Zeile oder Spalte verschoben, sobald der verfügbare Platz nicht mehr ausreicht.

```
$flow.WrapContents = $true
```

Dadurch kann sich das Layout automatisch an die Größe des Containers anpassen.

Eigenschaften

Eigenschaft	Beschreibung
AutoScroll	Aktiviert automatisch Scrollleisten, wenn der Inhalt den verfügbaren Bereich überschreitet.
AutoSize	Passt die Größe des Panels automatisch an seinen Inhalt an.
BackColor	Legt die Hintergrundfarbe des Panels fest.
BorderStyle	Bestimmt die Darstellung des Rahmens.
Controls	Enthält alle Controls des Panels.
Dock	Dockt das Panel an einer Seite seines Parent-Containers an.
FlowDirection	Bestimmt die Richtung, in der Controls angeordnet werden.
Location	Bestimmt die Position des Panels im Parent-Container.
Margin	Legt den äußeren Abstand des Panels zu seinem Parent fest.
MaximumSize	Definiert die maximal zulässige Größe.
MinimumSize	Definiert die minimal zulässige Größe.
Name	Legt den internen Namen des Panels fest.
Padding	Legt den Innenabstand zwischen Rand und enthaltenen Controls fest.
Size	Bestimmt Breite und Höhe des Panels.
WrapContents	Legt fest, ob Controls automatisch in eine neue Zeile oder Spalte umgebrochen werden.
Visible	Legt fest, ob das Panel sichtbar ist.
Enabled	Legt fest, ob das Panel und seine enthaltenen Controls verwendet werden können.

AutoScroll

AutoScroll

Typ = [System.Boolean]

Der Wert von `AutoScroll` legt fest, ob das `FlowLayoutPanel` automatisch Scrollleisten anzeigt, wenn der enthaltene Inhalt größer als der sichtbare Bereich ist.

Standardmäßig besitzt diese Eigenschaft den Wert `False`.

Ist `AutoScroll` auf `True` gesetzt, werden horizontale und/oder vertikale Scrollleisten angezeigt, sobald die enthaltenen Controls nicht mehr vollständig in den verfügbaren Bereich passen.

```
$flow.AutoScroll = $true
```

“ **Hinweis** `AutoScroll` ist besonders nützlich, wenn die Anzahl oder Größe der enthaltenen Controls zur Laufzeit variieren kann.

AutoSize

AutoSize

Typ = `[System.Boolean]`

Der Wert von `AutoSize` legt fest, ob sich die Größe des `FlowLayoutPanel` automatisch an seinen Inhalt anpasst.

Standardmäßig besitzt diese Eigenschaft den Wert `False`.

Ist `AutoSize` aktiviert, kann das Panel seine Größe anhand der enthaltenen Controls und deren Layout bestimmen.

```
$flow.AutoSize = $true
```

“ **Hinweis** Das tatsächliche Verhalten von `AutoSize` hängt unter anderem von `FlowDirection`, `WrapContents`, `Dock` und den Größen der enthaltenen Controls ab.

BackColor

BackColor

Typ = `[System.Drawing.Color]`

Der Wert von `BackColor` legt die Hintergrundfarbe des `FlowLayoutPanel` fest.

```
$flow.BackColor = "WhiteSmoke"
```

BorderStyle

BorderStyle

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

Der Wert von `BorderStyle` bestimmt, ob und wie das `FlowLayoutPanel` einen sichtbaren Rahmen besitzt.

Folgende Werte stehen zur Verfügung:

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

```
$flow.BorderStyle = "FixedSingle"
```

Controls

Controls

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

Die Eigenschaft `Controls` enthält alle Controls, die sich innerhalb des `FlowLayoutPanel` befinden.

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

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

Mehrere Controls können beispielsweise über eine Schleife hinzugefügt werden:

```
foreach ($button in $buttons) {  
    $flow.Controls.Add($button)  
}
```

```
}
```

Die Reihenfolge der Controls innerhalb der Collection entspricht dabei grundsätzlich auch der Reihenfolge, in der sie vom Layoutsystem angeordnet werden.

Dock

Dock

Typ = [System.Windows.Forms.DockStyle]

Der Wert von `Dock` legt fest, an welcher Seite seines Parent-Containers das `FlowLayoutPanel` angedockt wird.

```
$flow.Dock = "Fill"
```

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

Weitere mögliche Werte sind:

- `None`
- `Top`
- `Bottom`
- `Left`
- `Right`
- `Fill`

“ **Hinweis** Wird `Dock = "Fill"` verwendet, passt sich das Panel automatisch an die Größe des Parent-Containers an. Dadurch kann insbesondere `WrapContents` seine Wirkung beim Vergrößern oder Verkleinern des Fensters zeigen.

FlowDirection

FlowDirection

Typ = [System.Windows.Forms.FlowDirection]

Der Wert von `FlowDirection` bestimmt die Richtung, in der die enthaltenen Controls angeordnet werden.

Standardmäßig besitzt diese Eigenschaft den Wert `LeftToRight`.

Folgende Werte stehen zur Verfügung:

Wert	Beschreibung
<code>LeftToRight</code>	Controls werden von links nach rechts angeordnet.
<code>RightToLeft</code>	Controls werden von rechts nach links angeordnet.
<code>TopDown</code>	Controls werden von oben nach unten angeordnet.
<code>BottomUp</code>	Controls werden von unten nach oben angeordnet.

```
$flow.FlowDirection = "LeftToRight"
```

Beispiel für eine vertikale Anordnung:

```
$flow.FlowDirection = "TopDown"
```

“ **Hinweis** Zusammen mit `WrapContents` bestimmt `FlowDirection`, in welche Richtung das Layout zunächst fließt und wann ein Umbruch erfolgt.

Location

Location

Typ = [System.Drawing.Point]

Der Wert von `Location` legt die Position des `FlowLayoutPanel` innerhalb seines Parent-Containers fest.

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

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

📌 **Hinweis** Wird zusätzlich `Dock` verwendet, wird die Position durch das Layoutsystem bestimmt.

Margin

Margin

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

Der Wert von `Margin` legt den äußeren Abstand des `FlowLayoutPanel` zu seinem Parent-Container fest.

```
$flow.Margin = [System.Windows.Forms.Padding]::new(10)
```

Bei den enthaltenen Controls besitzt `Margin` eine zusätzliche Bedeutung: Das `FlowLayoutPanel` berücksichtigt den Außenabstand der einzelnen Controls bei der automatischen Anordnung.

```
$button.Margin = [System.Windows.Forms.Padding]::new(5)
```

Dadurch entsteht beispielsweise ein Abstand von 5 Pixeln zwischen den einzelnen Controls.

MaximumSize

MaximumSize

Typ = `[System.Drawing.Size]`

Der Wert von `MaximumSize` legt die maximal zulässige Größe des `FlowLayoutPanel` fest.

Standardmäßig besitzt diese Eigenschaft den Wert `(0,0)`, wodurch keine Größenbegrenzung besteht.

```
$flow.MaximumSize = "600, 400"
```

MinimumSize

MinimumSize

Typ = [System.Drawing.Size]

Der Wert von `MinimumSize` legt die minimal zulässige Größe des `FlowLayoutPanel` fest.

```
$flow.MinimumSize = "200, 100"
```

Wird versucht, das Panel kleiner als die angegebene Mindestgröße zu machen, bleibt diese Mindestgröße erhalten.

Name

Name

Typ = [System.String]

Der Wert von `Name` legt den internen Namen des `FlowLayoutPanel` fest.

Der Name dient ausschließlich zur Identifikation innerhalb des Programms.

```
$flow.Name = "flowButtons"
```

Padding

Padding

Typ = [System.Windows.Forms.Padding]

Der Wert von `Padding` legt den Innenabstand zwischen dem Rand des `FlowLayoutPanel` und seinen enthaltenen Controls fest.

```
$flow.Padding = [System.Windows.Forms.Padding]::new(10)
```

Dadurch werden die enthaltenen Controls mit einem Abstand von 10 Pixeln zum Rand des Panels angeordnet.

“ **Hinweis** `Padding` betrifft den Innenbereich des Panels, während `Margin` den Außenabstand des Panels beziehungsweise der enthaltenen Controls beschreibt.

Size

Size

Typ = `[System.Drawing.Size]`

Der Wert von `Size` legt die Breite und Höhe des `FlowLayoutPanel` fest.

```
$flow.Size = "500, 300"
```

Die tatsächliche Größe kann durch `AutoSize`, `Dock` oder andere Layoutmechanismen beeinflusst werden.

WrapContents

WrapContents

Typ = `[System.Boolean]`

Der Wert von `WrapContents` legt fest, ob Controls automatisch in eine neue Zeile beziehungsweise Spalte umgebrochen werden, wenn der verfügbare Platz nicht mehr ausreicht.

Standardmäßig besitzt diese Eigenschaft den Wert `True`.

```
$flow.WrapContents = $true
```

Bei einem horizontalen Layout:

[Button 1] [Button 2] [Button 3]

[Button 4] [Button 5] [Button 6]

Wird `WrapContents` deaktiviert, versucht das `FlowLayoutPanel`, alle Controls in der ursprünglichen Flussrichtung anzuordnen.

```
$flow.WrapContents = $false
```

Bei `LeftToRight` werden die Controls dadurch beispielsweise weiterhin horizontal angeordnet, auch wenn dadurch der verfügbare Bereich überschritten wird.

“ **Hinweis** `WrapContents` wirkt immer zusammen mit `FlowDirection`. Die Flussrichtung bestimmt, wohin die Controls zunächst angeordnet werden, während `WrapContents` bestimmt, ob bei fehlendem Platz ein Umbruch erfolgt.

Visible

Visible

Typ = `[System.Boolean]`

Der Wert von `Visible` legt fest, ob das `FlowLayoutPanel` sichtbar dargestellt wird.

```
$flow.Visible = $false
```

Enabled

Enabled

Typ = `[System.Boolean]`

Der Wert von `Enabled` legt fest, ob das `FlowLayoutPanel` aktiviert ist.

Standardmäßig besitzt diese Eigenschaft den Wert `True`.

```
$flow.Enabled = $false
```

Wird das Panel deaktiviert, werden auch die enthaltenen Controls entsprechend deaktiviert dargestellt beziehungsweise reagieren nicht mehr auf Eingaben.

Methoden

Übersicht

Methode	Beschreibung
<code>GetFlowBreak()</code>	Ermittelt, ob nach einem bestimmten Control ein Zeilenbeziehungsweise Spaltenumbruch erfolgt.
<code>SetFlowBreak()</code>	Legt fest, ob nach einem bestimmten Control ein Zeilenbeziehungsweise Spaltenumbruch erfolgen soll.

GetFlowBreak()

GetFlowBreak()

Die Methode `GetFlowBreak()` ermittelt, ob für ein bestimmtes Control ein manueller Umbruch festgelegt wurde.

```
$flow.GetFlowBreak($button)
```

Der Rückgabewert ist ein `Boolean`.

True → Nach dem Control erfolgt ein Umbruch.

False → Es erfolgt kein manueller Umbruch.

Beispiel:

```
if ($flow.GetFlowBreak($button)) {  
    Write-Host "Nach dem Button beginnt eine neue Zeile."
```

```
}
```

SetFlowBreak()

SetFlowBreak()

Die Methode `SetFlowBreak()` legt fest, ob nach einem bestimmten Control ein manueller Umbruch erfolgen soll.

```
$flow.SetFlowBreak($button, $true)
```

Der erste Parameter bestimmt das Control, nach dem der Umbruch erfolgen soll.

Der zweite Parameter bestimmt, ob der Umbruch aktiviert oder deaktiviert wird.

```
$flow.SetFlowBreak($button, $true)
```

→ Nach dem Button beginnt ein neuer Layoutabschnitt.

```
$flow.SetFlowBreak($button, $false)
```

→ Der manuelle Umbruch wird wieder entfernt.

Beispiel

```
$flow = [System.Windows.Forms.FlowLayoutPanel]::new()  
  
$button1 = [System.Windows.Forms.Button]::new()  
$button1.Text = "Button 1"  
  
$button2 = [System.Windows.Forms.Button]::new()  
$button2.Text = "Button 2"  
  
$button3 = [System.Windows.Forms.Button]::new()  
$button3.Text = "Button 3"  
  
$flow.Controls.Add($button1)
```

```
$flow.Controls.Add($button2)
$flow.Controls.Add($button3)

$flow.SetFlowBreak($button2, $true)
```

Das Layout kann dadurch beispielsweise so aussehen:

```
[Button 1] [Button 2]
[Button 3]
```

“ **Hinweis** `SetFlowBreak()` ist besonders praktisch, wenn einzelne Controls unabhängig von der verfügbaren Breite einen neuen Layoutabschnitt beginnen sollen.

Events

Das `FlowLayoutPanel` besitzt die üblichen Events eines Windows-Forms-Controls. Besonders relevant sind Events, die durch Änderungen am enthaltenen Layout oder an der `Controls`-Collection ausgelöst werden.

Event	Beschreibung
<code>ControlAdded</code>	Wird ausgelöst, wenn ein Control hinzugefügt wird.
<code>ControlRemoved</code>	Wird ausgelöst, wenn ein Control entfernt wird.
<code>Layout</code>	Wird ausgelöst, wenn das Layout des Panels neu berechnet wird.
<code>SizeChanged</code>	Wird ausgelöst, wenn sich die Größe des Panels ändert.

ControlAdded

ControlAdded

Das Event `ControlAdded` wird ausgelöst, sobald ein Control zur `Controls`-Collection des `FlowLayoutPanel` hinzugefügt wurde.

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

Das Event kann beispielsweise verwendet werden, um neu hinzugefügte Controls automatisch zu konfigurieren.

ControlRemoved

ControlRemoved

Das Event `ControlRemoved` wird ausgelöst, sobald ein Control aus der `Controls`-Collection entfernt wurde.

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

Layout

Layout

Das Event `Layout` wird ausgelöst, wenn das Layout des `FlowLayoutPanel` neu berechnet wird.

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

Das Event kann beispielsweise verwendet werden, wenn auf Änderungen der Größe oder Anordnung der enthaltenen Controls reagiert werden soll.

“ ⚠ **Hinweis** Das `Layout`-Event kann relativ häufig ausgelöst werden. Aufwendige Operationen sollten daher nicht unkontrolliert innerhalb dieses Events ausgeführt werden.

Beispiel

Das folgende Beispiel erstellt ein `FlowLayoutPanel`, das mehrere Buttons automatisch horizontal anordnet und bei Bedarf in eine neue Zeile umbricht.

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

$form = [System.Windows.Forms.Form]::new()
$form.Text = "FlowLayoutPanel"
$form.Size = "500, 300"

$flow = [System.Windows.Forms.FlowLayoutPanel]::new()
$flow.Dock = "Fill"
$flow.Padding = [System.Windows.Forms.Padding]::new(10)
$flow.FlowDirection = "LeftToRight"
$flow.WrapContents = $true
$flow.AutoScroll = $true

$form.Controls.Add($flow)

foreach ($i in 1..10) {

    $button = [System.Windows.Forms.Button]::new()
    $button.Text = "Button $i"
    $button.Size = "100, 40"
    $button.Margin = [System.Windows.Forms.Padding]::new(5)
```

```
$flow.Controls.Add($button)
}

$form.ShowDialog()
```

Das Layout passt sich automatisch an die verfügbare Breite des Fensters an:

```
| [Button 1] [Button 2] [Button 3] |
| [Button 4] [Button 5] [Button 6] |
| [Button 7] [Button 8] [Button 9] |
| [Button 10]                      |
```

Wird das Fenster breiter, können mehr Controls in einer Zeile dargestellt werden. Wird es schmaler, werden die Controls automatisch in weitere Zeilen umgebrochen.

FlowLayoutPanel vs. andere Container

Control	Anordnung
Panel	Freie Positionierung über Location
FlowLayoutPanel	Automatische Anordnung hintereinander
TableLayoutPanel	Anordnung in Zeilen und Spalten
TabPage	Container für den Inhalt eines Tabs

Ein FlowLayoutPanel ist damit besonders dann geeignet, wenn Controls **in einer bestimmten Reihenfolge automatisch angeordnet**, aber nicht an feste Zeilen oder Spalten gebunden werden sollen.

Hinweise

- FlowDirection bestimmt die Richtung des Layouts.
- WrapContents bestimmt, ob bei fehlendem Platz automatisch umgebrochen wird.

- `Margin` der enthaltenen Controls wird beim Layout berücksichtigt.
 - `Padding` des `FlowLayoutPanel` bestimmt den Abstand der Controls zum Rand.
 - `SetFlowBreak()` ermöglicht manuelle Umbrüche unabhängig vom verfügbaren Platz.
 - `AutoScroll` kann verwendet werden, wenn der Inhalt größer als der sichtbare Bereich werden kann.
 - `AutoSize` und `WrapContents` können sich gegenseitig stark auf das Layoutverhalten auswirken.
 - `Location` der enthaltenen Controls sollte bei Verwendung eines `FlowLayoutPanel` normalerweise nicht manuell gesetzt werden, da ihre Position vom Layoutsystem bestimmt wird.
-

Revision #1

Created 2026-08-15 15:40:15 UTC by John-Andreas Borinas

Updated 2026-08-23 19:25:50 UTC by John-Andreas Borinas