

Label

Ein `Label` ist ein Steuerelement zur **Anzeige von Text und optionalen Bildern**.

Im Gegensatz zu interaktiven Controls wie `Button` oder `TextBox` dient ein `Label` hauptsächlich dazu, Informationen, Beschriftungen oder Hinweise innerhalb einer Benutzeroberfläche darzustellen.

Ein `Label` kann jedoch auch auf Mausereignisse reagieren und beispielsweise über das `Click`-Event als anklickbares Element verwendet werden.

Grundlagen

Ein `Label` wird hauptsächlich verwendet, um Informationen innerhalb eines Formulars darzustellen.

- `Label` → zeigt Informationen oder Beschriftungen an
 - `Button` → führt eine Aktion aus
 - `TextBox` → ermöglicht Texteingaben
-

Label erstellen

```
# Klassisch
$label = New-Object System.Windows.Forms.Label

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

Label hinzufügen

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

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

# oder
```

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

Text festlegen

Der sichtbare Inhalt eines Labels wird über die Eigenschaft `Text` festgelegt.

```
$label.Text = "Benutzername:"
```

Eigenschaften

Eigenschaft	Beschreibung
<code>AutoEllipsis</code>	Zeigt bei nicht vollständig darstellbarem Text automatisch <code>...</code> an.
<code>AutoSize</code>	Passt die Größe des Labels automatisch an dessen Inhalt an.
<code>BackColor</code>	Legt die Hintergrundfarbe des Labels fest.
<code>BorderStyle</code>	Bestimmt, ob und wie ein Rahmen um das Label dargestellt wird.
<code>Cursor</code>	Legt den Mauszeiger fest, wenn sich der Mauszeiger über dem Label befindet.
<code>Dock</code>	Dockt das Label an einer Seite seines Parent-Containers an.
<code>Enabled</code>	Legt fest, ob das Label aktiviert dargestellt wird und auf Eingaben reagieren kann.
<code>FlatStyle</code>	Bestimmt die Darstellungsart des Labels.
<code>Font</code>	Legt Schriftart, -größe und -stil fest.
<code>ForeColor</code>	Legt die Textfarbe fest.
<code>Image</code>	Zeigt ein Bild im Label an.
<code>ImageAlign</code>	Bestimmt die Position des Bildes innerhalb des Labels.
<code>ImageIndex</code>	Wählt ein Bild anhand seines Indexes aus der <code>ImageList</code> aus.
<code>ImageKey</code>	Wählt ein Bild anhand seines Namens aus der <code>ImageList</code> aus.
<code>ImageList</code>	Legt die Bildersammlung für das Label fest.

Eigenschaft	Beschreibung
Location	Bestimmt die Position des Labels im Parent-Container.
Margin	Legt den äußeren Abstand zu benachbarten Controls fest.
MaximumSize	Definiert die maximal zulässige Größe.
MinimumSize	Definiert die minimal zulässige Größe.
Name	Legt den internen Namen des Labels fest.
Padding	Legt den Innenabstand zwischen Rand und Inhalt fest.
Size	Bestimmt Breite und Höhe des Labels.
TabIndex	Legt die Reihenfolge der Tabulator-Navigation fest.
TabStop	Legt fest, ob das Label per Tabulator fokussiert werden kann.
Text	Bestimmt den sichtbaren Text des Labels.
TextAlign	Legt die Position des Textes innerhalb des Labels fest.
UseCompatibleTextRendering	Bestimmt, ob für die Textdarstellung die ältere GDI+- oder die neuere GDI-Textdarstellung verwendet wird.
UseMnemonic	Aktiviert Tastenkombinationen über <code>&</code> im Text.
Visible	Legt fest, ob das Label sichtbar ist.

AutoEllipsis

AutoEllipsis

Typ = `[System.Boolean]`

Der Wert von `AutoEllipsis` legt fest, ob zu langer Text automatisch mit `...` gekürzt werden soll, wenn dieser nicht vollständig dargestellt werden kann.

Standardmäßig ist diese Eigenschaft auf `False` gesetzt.

```
$label.AutoEllipsis = $true
```

“ **Hinweis** `AutoEllipsis` ist insbesondere dann sinnvoll, wenn die Größe des Labels begrenzt ist und der vollständige Text nicht angezeigt werden kann.

AutoSize

AutoSize

Typ = [System.Boolean]

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

Beim `Label` ist `AutoSize` standardmäßig auf `True` gesetzt.

Dadurch wird die Größe automatisch anhand des enthaltenen Textes beziehungsweise Bildes bestimmt.

```
$label.AutoSize = $true
```

Soll das Label eine feste Größe besitzen, kann `AutoSize` deaktiviert werden.

```
$label.AutoSize = $false  
$label.Size = "200, 40"
```

BackColor

BackColor

Typ = [System.Drawing.Color]

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

```
$label.BackColor = "LightBlue"
```

Standardmäßig übernimmt das Label die Hintergrundfarbe seines Parent-Controls beziehungsweise die von Windows vorgegebene Standardfarbe.

BorderStyle

BorderStyle

Typ = [System.Windows.Forms.BorderStyle]

Der Wert von `BorderStyle` bestimmt, ob und wie ein Rahmen um das Label dargestellt wird.

Folgende Werte stehen zur Verfügung:

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

Standardmäßig besitzt diese Eigenschaft den Wert `None`.

```
$label.BorderStyle = "FixedSingle"
```

Cursor

Cursor

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

Der Wert von `Cursor` legt fest, welcher Mauszeiger angezeigt wird, wenn sich der Mauszeiger über dem Label befindet.

```
$label.Cursor = [System.Windows.Forms.Cursors]::Hand
```

Dies kann beispielsweise verwendet werden, wenn das Label wie ein anklickbarer Link verwendet wird.

Dock

Dock

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

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

Standardmäßig besitzt diese Eigenschaft den Wert `None`.

```
$label.Dock = "Top"
```

Alternativ kann das Label mit `Fill` den gesamten verfügbaren Bereich einnehmen.

```
$label.Dock = "Fill"
```

Im Gegensatz zu `Anchor` übernimmt `Dock` sowohl die Positionierung als auch die Größenanpassung des Controls.

Enabled

Enabled

Typ = `[System.Boolean]`

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

Standardmäßig besitzt diese Eigenschaft den Wert `True`.

Wird `Enabled` auf `False` gesetzt, wird das Label deaktiviert dargestellt und reagiert nicht mehr auf Eingaben.

```
$label.Enabled = $false
```

“ **Hinweis** Bei einem gewöhnlichen Label betrifft `Enabled` hauptsächlich die Darstellung. Ein Label ist standardmäßig ohnehin nicht über die Tabulator-Navigation fokussierbar.

FlatStyle

FlatStyle

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

Der Wert von `FlatStyle` bestimmt die Darstellungsart des Labels.

Folgende Werte stehen zur Verfügung:

- **Flat** → flache Darstellung
- **Popup** → flache Darstellung mit Hervorhebung bei Interaktion
- **Standard** → Standarddarstellung
- **System** → Darstellung durch Windows

```
$label.FlatStyle = "Flat"
```

Font

Font

Typ = [System.Drawing.Font]

Der Wert von `Font` legt die Schriftart fest, mit der der Text des Labels dargestellt wird.

Änderungen an dieser Eigenschaft beeinflussen Schriftart, Schriftgröße und Schriftstil.

```
$label.Font = [System.Drawing.Font]::new(  
    "Segoe UI",  
    10,  
    "Bold"  
)
```

ForeColor

ForeColor

Typ = [System.Drawing.Color]

Der Wert von `ForeColor` legt die Farbe fest, mit der der Text des Labels dargestellt wird.

```
$label.ForeColor = "White"
```

Image

Image

Typ = [System.Drawing.Image]

Der Wert von `Image` legt das Bild fest, das im Label angezeigt werden soll.

```
$label.Image = [System.Drawing.Image]::FromFile(  
    "C:\Icons\Info.png"  
)
```

Ist gleichzeitig Text vorhanden, bestimmt `TextAlign` beziehungsweise `ImageAlign`, wie die Inhalte innerhalb des Labels positioniert werden.

ImageAlign

ImageAlign

Typ = [System.Drawing.ContentAlignment]

Der Wert von `ImageAlign` legt fest, an welcher Position das Bild innerhalb des Labels dargestellt wird.

```
$label.ImageAlign = "MiddleCenter"
```

ImageIndex

ImageIndex

Typ = [System.Int32]

Der Wert von `ImageIndex` bestimmt den Index des Bildes innerhalb der zugewiesenen `ImageList`.

Standardmäßig ist kein Bild ausgewählt.

```
$label.ImageList = $imageList  
$label.ImageIndex = 0
```

“ **Hinweis** `ImageIndex` und `ImageKey` dienen demselben Zweck. Es sollte immer nur eine der beiden Eigenschaften verwendet werden.

ImageKey

ImageKey

Typ = [System.String]

Der Wert von `ImageKey` legt den Namen eines Bildes innerhalb der zugewiesenen `ImageList` fest.

```
$label.ImageList = $imageList  
$label.ImageKey = "Info"
```

Im Gegensatz zu `ImageIndex` erfolgt die Auswahl über den Namen des Bildes.

ImageList

ImageList

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

Der Wert von `ImageList` legt die Bildersammlung fest, aus der das Label ein Bild beziehen kann.

Welche Grafik verwendet wird, wird anschließend über `ImageIndex` oder `ImageKey` bestimmt.

```
$label.ImageList = $imageList  
$label.ImageIndex = 0
```

Location

Location

Typ = `[System.Drawing.Point]`

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

Die Position wird über die X- und Y-Koordinate relativ zum Parent-Control angegeben.

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

“ **Hinweis** Wird das Label durch einen Layout-Container oder `Dock` positioniert, wird `Location` vom Layoutsystem verwaltet.

Margin

Margin

Typ = [System.Windows.Forms.Padding]

Der Wert von `Margin` legt den äußeren Abstand des Labels zu benachbarten Controls fest.

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

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

MaximumSize

MaximumSize

Typ = [System.Drawing.Size]

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

Standardmäßig besitzt diese Eigenschaft den Wert `(0,0)`. Dadurch existiert keine Größenbegrenzung.

```
$label.MaximumSize = "300, 100"
```

MinimumSize

MinimumSize

Typ = [System.Drawing.Size]

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

```
$label.MinimumSize = "100, 25"
```

Name

Name

Typ = [System.String]

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

Der Name dient ausschließlich der Identifikation innerhalb des Programms und wird dem Benutzer nicht angezeigt.

```
$label.Name = "lblUsername"
```

Padding

Padding

Typ = [System.Windows.Forms.Padding]

Der Wert von `Padding` legt den Innenabstand zwischen dem Rand des Labels und dessen Inhalt fest.

```
$label.Padding = [System.Windows.Forms.Padding]::new(8)
```

Size

Size

Typ = [System.Drawing.Size]

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

```
$label.Size = "200, 35"
```

Ist `AutoSize` aktiviert, kann die Größe automatisch anhand des Inhalts angepasst werden.

TabIndex

TabIndex

Typ = [System.Int32]

Der Wert von `TabIndex` legt die Reihenfolge fest, in der das Label den Fokus erhält, wenn der Benutzer die **Tabulator-Taste** betätigt.

Standardmäßig besitzt ein Label einen `TabIndex`, obwohl es normalerweise nicht über die Tabulator-Navigation fokussiert werden kann.

```
$label.TabIndex = 2
```

“ **Hinweis** `TabIndex` wird beim Label erst relevant, wenn `TabStop` auf `True` gesetzt wird.

TabStop

TabStop

Typ = `[System.Boolean]`

Der Wert von `TabStop` legt fest, ob das Label über die **Tabulator-Taste** den Fokus erhalten kann.

Beim `Label` ist diese Eigenschaft standardmäßig auf `False` gesetzt.

```
$label.TabStop = $true
```

Dadurch kann das Label Teil der Tabulator-Navigation werden.

“ **Hinweis** Ein gewöhnliches Label ist kein interaktives Eingabeelement. Deshalb ist `TabStop` standardmäßig deaktiviert.

Text

Text

Typ = `[System.String]`

Der Wert von `Text` legt den sichtbaren Text des Labels fest.

```
$label.Text = "Benutzername:"
```

Standardmäßig besitzt diese Eigenschaft den Wert `""` (leerer String).

TextAlign

TextAlign

Typ = [System.Drawing.ContentAlignment]

Der Wert von `TextAlign` legt fest, an welcher Position der Text innerhalb des Labels dargestellt wird.

```
$label.TextAlign = "MiddleCenter"
```

Dadurch wird der Text sowohl horizontal als auch vertikal zentriert.

UseCompatibleTextRendering

UseCompatibleTextRendering

Typ = [System.Boolean]

Der Wert von `UseCompatibleTextRendering` bestimmt, welche Textdarstellung für das Label verwendet wird.

Bei `False` wird die neuere GDI-basierte Textdarstellung verwendet. Bei `True` wird die ältere GDI+-basierte Darstellung verwendet.

```
$label.UseCompatibleTextRendering = $true
```

Die Eigenschaft ist hauptsächlich für Kompatibilität mit älteren Anwendungen relevant.

“ **Hinweis** In den meisten Fällen kann der Standardwert verwendet werden. Die Eigenschaft wird insbesondere dann interessant, wenn Unterschiede bei der Textdarstellung, beispielsweise bei Zeilenumbruch oder Schriftmessung, auftreten.

UseMnemonic

UseMnemonic

Typ = [System.Boolean]

Der Wert von `UseMnemonic` legt fest, ob das Zeichen `&` im Text zur Definition eines **Tastaturkürzels** verwendet wird.

Ist die Eigenschaft aktiviert, wird das Zeichen direkt vor einem Buchstaben als Kennzeichnung für eine Mnemonic-Taste interpretiert.

```
$label.Text = "&Name:"  
$label.UseMnemonic = $true
```

Das `N` kann dadurch als Zugriffstaste verwendet werden.

Soll das Zeichen `&` tatsächlich angezeigt werden, kann es durch `&&` maskiert werden.

```
$label.Text = "Speichern && Beenden"
```

Visible

Visible

Typ = `[System.Boolean]`

Der Wert von `Visible` legt fest, ob das Label sichtbar ist.

Standardmäßig besitzt diese Eigenschaft den Wert `True`.

```
$label.Visible = $false
```

Wird die Eigenschaft auf `False` gesetzt, wird das Label ausgeblendet.

Events

Ein `Label` kann auf verschiedene Ereignisse reagieren. Besonders häufig werden Mausereignisse verwendet, wenn das Label wie ein anklickbares Element eingesetzt wird.

Event	Beschreibung
<code>Click</code>	Das Label wurde angeklickt.
<code>DoubleClick</code>	Das Label wurde doppelt angeklickt.

Event	Beschreibung
MouseEnter	Der Mauszeiger befindet sich über dem Label.
MouseLeave	Der Mauszeiger verlässt das Label.
MouseDown	Eine Maustaste wurde über dem Label gedrückt.
MouseUp	Eine Maustaste wurde über dem Label losgelassen.
TextChanged	Der Text des Labels wurde geändert.
VisibleChanged	Die Sichtbarkeit des Labels wurde geändert.
EnabledChanged	Der Aktivierungszustand des Labels wurde geändert.
FontChanged	Die Schriftart des Labels wurde geändert.
ForeColorChanged	Die Textfarbe des Labels wurde geändert.
SizeChanged	Die Größe des Labels wurde geändert.

Event hinzufügen

Events werden beim `Label` mit dem Präfix `Add_` hinzugefügt.

```
$label.Add_Click({  
    param($sender, $e)  
  
    Write-Host "Label wurde geklickt."  
})
```

Event entfernen

Ein zuvor hinzugefügtes Event kann über das entsprechende `Remove_`-Präfix wieder entfernt werden.

```
$label.Remove_Click($event)
```

Tipps

Label automatisch an den Text anpassen

Mit `AutoSize` kann die Größe automatisch an den Inhalt angepasst werden.

```
$label.AutoSize = $true
```

Typische Verwendung

Ein `Label` wird häufig für folgende Aufgaben verwendet:

- Beschriftungen von Eingabefeldern
- Überschriften
- Statusinformationen
- Hinweise und Hilfetexte
- Anzeigen von Werten
- klickbare Text-Elemente
- Anzeige von Icons oder Bildern

Ein typisches Formular kann beispielsweise so aufgebaut sein:

```
$label = [System.Windows.Forms.Label]::new()  
  
$label.Text      = "Benutzername:"  
$label.AutoSize  = $true  
$label.Location  = "20, 20"  
  
$form.Controls.Add($label)
```

Label als klickbares Element

Obwohl ein `Label` hauptsächlich zur Darstellung von Informationen dient, kann es auf Mausereignisse reagieren.

Dadurch kann es beispielsweise als einfacher Link oder als Schaltfläche für kleinere Aktionen verwendet werden.

```
$label.Text      = "Über das Programm"  
$label.Cursor    = [System.Windows.Forms.Cursors]::Hand  
  
$label.Add_Click({
```

Für komplexere Aktionen sollte jedoch weiterhin ein `Button` verwendet werden.

“ **Hinweis** Ein `Label` besitzt zwar ein `Click`-Event, ist aber semantisch kein `Button`. Bei wichtigen oder häufig verwendeten Aktionen ist ein `Button` daher die bessere Wahl.

Hinweise

- `Label` dient hauptsächlich zur **Darstellung von Informationen**.
- `AutoSize` ist beim `Label` standardmäßig aktiviert.
- `TabStop` ist beim `Label` standardmäßig deaktiviert.
- `TabIndex` ist vorhanden, wird aber erst relevant, wenn `TabStop` aktiviert wird.
- Über `Image`, `ImageList`, `ImageIndex` und `ImageKey` können Bilder angezeigt werden.
- Mit `TextAlign` kann der Text innerhalb des Labels positioniert werden.
- Über `Click` kann ein Label interaktiv gemacht werden.
- Für echte Benutzeraktionen sollte in der Regel ein `Button` verwendet werden.

Eine kleine Korrektur gegenüber meiner vorherigen Antwort ist dabei wichtig: `**`TabIndex` und `TabStop` sind beim `Label` tatsächlich vorhanden, aber `TabStop` steht standardmäßig auf `False`**`. Das ist für deine Doku die relevante Information, statt so zu tun, als wäre die bloße Existenz der beiden Properties schon praktisch bedeutungsvoll. WinForms liebt solche geerbten Eigenschaften. :contentReference[oaicite:1]{index=1}