TabControl

Ein `TabControl` ist ein Container, der mehrere `TabPage`-Instanzen verwaltet und zwischen ihnen umschaltet.
Es stellt die Tabs (Reiter) dar und bestimmt, welche `TabPage` aktuell sichtbar ist.

Grundlagen

Das `TabControl` ist **die Steuerung**, nicht der Inhalt.

- `TabControl` → verwaltet Tabs
- `TabPage` → enthält den eigentlichen Inhalt

TabControl erstellen

```
# Klassisch
$tabControl = New-Object System.Windows.Forms.TabControl


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

TabPage hinzufügen

Ein `TabPage` wird nicht direkt zum `TabControl` hinzugefügt, sondern zur enthaltenen Sammlung `$tabControl.TabPages`.
Die Sammlung `TabPages` stellt mehrere Methoden zum Hinzufügen von `TabPage`-Instanzen bereit:

- `TabPages.Add($tabPage1)` – Fügt das `TabPage` `$tabPage1` zur Sammlung hinzu
- `TabPages.AddRange(@($tabPage2, $tabPage3))` – Fügt mehrere `TabPage`-Instanzen gleichzeitig als Array hinzu
- `TabPages.Insert(0, $tabPage4)` – Fügt das `TabPage` an der gewünschten Position innerhalb der Sammlung ein

```
$tabControl.TabPages.Add($tabPage1)


$tabControl.TabPages.AddRange(@(
    $tabPage2,
```

```
    $tabPage3
))

$tabControl.TabPages.Insert(0, $tabPage4)

# Technisch möglich, aber nicht empfohlen
$tabControl.Controls.Add($tabPage4)
$tabControl.Controls.AddRange(@(
    $tabPage5,
    $tabPage6
))
```

Über `Add()` kann zusätzlich direkt ein neues `TabPage` erstellt werden:

```
# Erstellt ein neues TabPage
$tabControl.TabPages.Add("TabText")

# Erstellt ein neues TabPage mit Name + Text
$tabControl.TabPages.Add("TabName", "TabText")
```

TabPage entfernen

Ein `TabPage` kann aus dem `TabControl` mit der Referenz zum TabPage und `Remove()` oder über den Index mit `RemoveAt()` entfernt werden.

```
$tabControl.TabPages.Remove($tabPage1) # mit Referenz
$tabControl.TabPages.RemoveAt(0) # mit Index
```

Mit `Clear()` werden alle `TabPage`-Instanzen entfernt.

```
# Alle entfernen
$tabControl.TabPages.Clear()
```

TabPage Auswahl/Zugriff

Mit dem jeweiligen Index vom TabPage, kann in TabPages direkt auf das TabPage zugegriffen werden.

```
# Zugriff auf einzelnes TabPage
$tabControl.TabPages[0]
```

```
# Aktiven Tab setzen
$tabControl.SelectedIndex = 0
$tabControl.SelectedTab = $tabPage1
```

Eigenschaften

Eigenschaften

- **Alignment** – Position der Tabs (`Top`, `Bottom`, `Left`, `Right`)
- **Anchor** – Verankerung an den Rändern des Parent-Containers
- **Appearance** – Darstellung der Tabs (`Normal`, `Buttons`, `FlatButtons`)
- **Dock** – Automatische Ausrichtung im Parent-Container
- **DrawMode** – Zeichenmodus der Tabs
- **HotTrack** – Hover-Effekt über Tabs
- **ImageList** – Sammlung der für Tabs verfügbaren Bilder
- **ItemSize** – Größe einzelner Tabs (nur relevant bei `SizeMode = Fixed`)
- **Multiline** – Mehrere Reihen von Tabs erlauben
- **Padding** – Innenabstand des Tab-Headers
- **RowCount** – Anzahl der Tab-Reihen (relevant bei `Multiline`)
- **SelectedImageIndex** – Icon für den ausgewählten Tab
- **SelectedIndex** – Index des aktuell aktiven Tabs
- **SelectedTab** – Referenz auf die aktuell aktive `TabPage`
- **ShowToolTips** – Tooltips für Tabs aktivieren
- **SizeMode** – Größe der Tabs (`Normal`, `Fixed`)
- **TabPages** – Sammlung aller enthaltenen `TabPage`-Instanzen

Alignment [Systems.Windows.Forms.TabAlignment]

Der Wert von `Alignment` legt fest, an welcher Seite des `TabControl` die Tabs dargestellt werden. Standardmäßig ist diese Eigenschaft auf `Top` gesetzt, wodurch sich die Tabs oberhalb des Inhaltsbereichs befinden. Alternativ können die Tabs auch am unteren (`Bottom`), linken (`Left`) oder rechten (`Right`) Rand angezeigt werden. Die Position der Tabs beeinflusst lediglich deren Darstellung und hat keinen Einfluss auf die enthaltenen `TabPage`-Instanzen oder deren Funktionalität.

Anchor [Systems.Windows.Forms.AnchorStyles]

Der Wert von `Anchor` legt fest, an welchen Rändern seines Parent-Containers ein Control verankert ist. Standardmäßig ist diese Eigenschaft auf `Top, Left` gesetzt, wodurch das Control seinen Abstand zum oberen und linken Rand beibehält. Wird die Größe des Parent-Containers verändert,

passt das Control seine Position oder Größe entsprechend den festgelegten Verankerungen an.

Mehrere Verankerungen können kombiniert werden. Ist ein Control beispielsweise an `Left` und `Right` verankert, wird seine Breite automatisch angepasst, um den Abstand zu beiden Rändern beizubehalten. Durch die Kombination verschiedener Werte lässt sich das Verhalten eines Controls bei Größenänderungen flexibel steuern.

Appearance [System.Windows.Forms.TabAppearance]

Der Wert von `Appearance` legt fest, wie die Tabs eines `TabControl` dargestellt werden. Standardmäßig ist diese Eigenschaft auf `Normal` gesetzt, wodurch die Tabs im klassischen Registerkarten-Stil angezeigt werden. Alternativ können die Tabs als Schaltflächen (`Buttons`) oder als flache Schaltflächen (`FlatButtons`) dargestellt werden.

- **Normal** → klassische Registerkarten
- **Buttons** → Tabs werden wie normale Schaltflächen dargestellt
- **FlatButtons** → Tabs werden wie flache Schaltflächen dargestellt

Die Eigenschaft beeinflusst ausschließlich das Erscheinungsbild der Tabs und hat keinen Einfluss auf die Funktionalität des `TabControl` oder der enthaltenen `TabPage`-Instanzen. Unabhängig von der gewählten Darstellung können Tabs weiterhin ausgewählt und gewechselt werden.

Dock [Systems.Windows.Forms.DockStyle]

Der Wert von `Dock` legt fest, an welcher Seite seines Parent-Containers ein Control angedockt wird. Standardmäßig ist diese Eigenschaft auf `None` gesetzt, wodurch die Position und Größe des Controls ausschließlich durch dessen `Location`- und `Size`-Eigenschaften bestimmt werden. Alternativ kann das Control an den oberen (`Top`), unteren (`Bottom`), linken (`Left`) oder rechten (`Right`) Rand angedockt oder mit `Fill` auf die gesamte verfügbare Fläche des Parent-Containers ausgedehnt werden.

Im Gegensatz zu `Anchor` bestimmt `Dock` nicht die Abstände zu den Rändern, sondern übernimmt die automatische Positionierung und Größenanpassung des Controls. Wird beispielsweise `Fill` verwendet, füllt das Control den gesamten verfügbaren Bereich seines Parent-Containers aus.

DrawMode [Systems.Windows.Forms.TabDrawMode]

Der Wert von `DrawMode` legt fest, wie die Tabs des `TabControl` gezeichnet werden. Standardmäßig ist diese Eigenschaft auf `Normal` gesetzt, wodurch das Betriebssystem die Darstellung der Tabs vollständig übernimmt. Wird `DrawMode` auf `OwnerDrawFixed` gesetzt, ist der Entwickler für das Zeichnen der Tabs verantwortlich und kann deren Aussehen individuell gestalten.

Die Einstellung `OwnerDrawFixed` wird häufig verwendet, um eigene Farben, Schriftarten oder Symbole für Tabs darzustellen. Da die Tabs dabei selbst gezeichnet werden müssen, wird zusätzlich das `DrawItem`-Event benötigt, in dem die eigentliche Darstellung implementiert wird.

HotTrack [System.Boolean]

Der Wert von `HotTrack` legt fest, ob Tabs auf Mausbewegungen reagieren sollen. Standardmäßig ist diese Eigenschaft auf `False` gesetzt, wodurch Tabs ihr Aussehen beim Überfahren mit dem Mauszeiger nicht verändern. Wird `HotTrack` auf `True` gesetzt, hebt das `TabControl` den Tab unter dem Mauszeiger visuell hervor, um die Interaktion für den Benutzer deutlicher zu machen.

ImageList [Systems.Windows.Forms.ImageList]

Der Wert von `ImageList` legt die Bildersammlung fest, aus der die Tabs ihre Symbole beziehen. Standardmäßig ist diese Eigenschaft auf `$null` gesetzt, wodurch keine Symbole angezeigt werden. Die Eigenschaft dient lediglich als Quelle der verfügbaren Bilder. Welche Bilder tatsächlich in den Tab-Headern angezeigt werden, wird über die Eigenschaften `ImageIndex` oder `ImageKey` der jeweiligen `TabPage` festgelegt.

ItemSize [System.Drawing.Size]

Der Wert von `ItemSize` legt die Größe der einzelnen Tabs fest. Standardmäßig besitzt diese Eigenschaft den Wert `(Width=0, Height=0)`, wodurch die Größe der Tabs automatisch durch das `TabControl` bestimmt wird. Die Eigenschaft wird erst relevant, wenn `SizeMode` auf `Fixed` gesetzt ist. In diesem Fall verwendet das `TabControl` die in `ItemSize` festgelegte Breite und Höhe für alle Tabs.

Multiline [System.Boolean]

Der Wert von `Multiline` legt fest, ob die Tabs auf mehrere Reihen verteilt werden dürfen. Standardmäßig ist diese Eigenschaft auf `False` gesetzt, wodurch alle Tabs in einer einzelnen Reihe dargestellt werden. Wird `Multiline` auf `True` gesetzt, erstellt das `TabControl` bei Platzmangel automatisch zusätzliche Reihen, sodass alle Tabs sichtbar bleiben können.

Padding [System.Windows.Forms.Padding]

Der Wert von `Padding` legt den Innenabstand innerhalb der Tab-Header fest. Standardmäßig ist diese Eigenschaft auf `(6, 3)` gesetzt. Dadurch wird zwischen dem Rand eines Tabs und dessen Inhalt, beispielsweise dem Text oder einem Icon, ein zusätzlicher Abstand eingefügt.

Die Eigenschaft beeinflusst nicht den Inhalt der enthaltenen `TabPage`-Instanzen, sondern ausschließlich die Darstellung der Tabs selbst. Durch größere Werte kann mehr Platz zwischen dem Rand eines Tabs und dessen Inhalt geschaffen werden, während kleinere Werte zu einer kompakteren Darstellung führen.

RowCount [System.Int32]

Der Wert von `RowCount` gibt an, aus wie vielen Reihen die Tabs aktuell bestehen. Standardmäßig beträgt der Wert `0`, solange sich keine `TabPage` im `TabControl` befindet. Die Eigenschaft wird vom `TabControl` automatisch ermittelt und kann nicht direkt festgelegt werden. Besonders relevant ist

`RowCount`, wenn `Multiline` auf `True` gesetzt ist, da die Tabs dann auf mehrere Reihen verteilt werden können.

SelectedImageIndex [System.Int32]

Der Wert von `SelectedImageIndex` legt den Index des Bildes fest, das für den aktuell ausgewählten Tab verwendet werden soll. Standardmäßig ist diese Eigenschaft auf `-1` gesetzt, wodurch kein spezielles Bild für den aktiven Tab definiert ist. Die Bilder werden dabei aus der dem `TabControl` zugewiesenen `ImageList` bezogen.

Ist ein gültiger Bildindex angegeben, kann für den ausgewählten Tab ein anderes Symbol als für die übrigen Tabs dargestellt werden. Die Eigenschaft wird hauptsächlich in Verbindung mit einer `ImageList` verwendet und hat ohne zugewiesene Bilder keine sichtbare Auswirkung.

SelectedIndex [System.Int32]

Der Wert von `SelectedIndex` entspricht dem Index des aktuell aktiven `TabPage`. Die `TabPages`-Collection ist 0-basiert, weshalb das erste `TabPage` den Index `0` besitzt. Befindet sich mindestens ein `TabPage` im `TabControl`, ist standardmäßig das erste `TabPage` aktiv. Ist die `TabPages`-Collection leer, beträgt der Wert von `SelectedIndex` `-1`.

SelectedTab [System.Windows.Forms.TabPage]

Der Wert von `SelectedTab` enthält eine Referenz auf die aktuell aktive `TabPage` des `TabControl`. Standardmäßig ist diese Eigenschaft auf `$null` gesetzt, solange sich keine `TabPage` in der `TabPages`-Collection befindet. Sobald mindestens ein `TabPage` vorhanden ist, verweist `SelectedTab` auf das aktuell ausgewählte `TabPage`. Über diese Eigenschaft kann sowohl das aktive `TabPage` ausgelesen als auch ein anderes `TabPage` direkt ausgewählt werden.

ShowToolTips [System.Boolean]

Der Wert von `ShowToolTips` legt fest, ob für die Tabs eines `TabControl` Tooltips angezeigt werden dürfen. Standardmäßig ist diese Eigenschaft auf `$false` gesetzt, wodurch keine Tooltips dargestellt werden. Wird `ShowToolTips` auf `$true` gesetzt, können einzelnen `TabPage`-Instanzen Tooltip-Texte zugewiesen werden, die beim Überfahren des jeweiligen Tabs mit dem Mauszeiger angezeigt werden.

SizeMode [Systems.Windows.Forms.TabSizeMode]

Der Wert von `SizeMode` legt fest, wie die Größe der einzelnen Tabs bestimmt wird. Standardmäßig ist diese Eigenschaft auf `Normal` gesetzt, wodurch die Breite jedes Tabs automatisch anhand seines Inhalts berechnet wird. Wird `SizeMode` auf `Fixed` gesetzt, erhalten alle Tabs dieselbe Größe, die über die Eigenschaft `ItemSize` festgelegt werden kann.

TabPages

[Systems.Windows.Forms.TabControl.TabPageCollection]

Der Wert von `TabPages` enthält die Sammlung aller `TabPage`-Instanzen, die dem `TabControl` hinzugefügt wurden. Standardmäßig ist diese Sammlung leer. Über `TabPages` können `TabPage`-Instanzen hinzugefügt, entfernt oder anhand ihres Indexes bzw. ihrer Referenz abgerufen werden. Die Reihenfolge der Elemente innerhalb der Sammlung entspricht dabei der Reihenfolge der Tabs im `TabControl`.

Methoden

Methoden

TabControl

- **GetTabRect** – Gibt die Position und Größe vom TabPage-Reiter zurück

TabControl.TabPages

- **Add** – Fügt eine `TabPage`-Instanz hinzu
- **AddRange** – Fügt mehrere `TabPage`-Instanzen hinzu
- **Clear** – Entfernt alle `TabPage`-Instanzen
- **Insert** – Fügt eine `TabPage`-Instanz an einer gewünschten Stelle hinzu
- **Remove** – Entfernt eine `TabPage`-Instanz
- **RemoveAt** – Entfernt mit dem Index eine `TabPage`-Instanz

TabControl

GetTabRect()

Datentyp: [System.Drawing.Rectangle]
Rückgabewert: Position und Größe des Tab-Headers

```
GetTabRect( Index )
```

Die Methode `GetTabRect()` gibt die Position und Größe eines Tabs innerhalb des `TabControl` zurück. Über den angegebenen Index wird festgelegt, für welchen Tab die Informationen ermittelt werden sollen. Der Rückgabewert ist ein `Rectangle`, das die Position sowie die Breite und Höhe des

entsprechenden Tab-Headers enthält. Die zurückgegebenen Koordinaten beziehen sich auf das TabControl selbst.

Die Methode wird häufig verwendet, um Mausklicks auf einzelne Tabs zu erkennen oder um eigene Zeichnungslogik mit `OwnerDrawFixed` umzusetzen.

TabControl.TabPages

Add()

```
$_.TabPages.Add( $tabPage )
# → vorhandenes TabPage hinzufügen
```

Die Methode `Add()` fügt eine `TabPage` zur `TabPages`-Collection hinzu. Das `TabPage` wird dabei am Ende der Sammlung eingefügt und erscheint als neuer Tab im `TabControl`.

Nach dem Hinzufügen kann das `TabPage` über die `TabPages`-Collection, `SelectedIndex` oder `SelectedTab` verwendet werden.

```
$_.TabPages.Add( "Text" )
# → Neues TabPage mit Text erstellen
```

Der übergebene Text wird dabei als Beschriftung des Tabs verwendet.

```
$_.TabPages.Add( "Name", "Text" )
# → Neues TabPage mit Name und Text erstellen
```

Hierbei wird sowohl der interne `Name` als auch die sichtbare Beschriftung (`Text`) festgelegt.

Diese Varianten eignen sich für einfache Tabs, bei denen keine weiteren Eigenschaften unmittelbar gesetzt werden müssen.

AddRange()

```
$_.TabPages.AddRange( $tabPages )
```

Die Methode `AddRange()` fügt mehrere `TabPage`-Instanzen gleichzeitig zur `TabPages`-Collection hinzu.

Die Reihenfolge der Elemente im Array entspricht anschließend der Reihenfolge der Tabs im `TabControl`. Die Tabs werden dabei am Ende der vorhandenen Sammlung eingefügt.

Clear()

```
$_.TabPages.Clear()
```

Die Methode `Clear()` entfernt alle `TabPage`-Instanzen aus der `TabPages`-Collection.

Nach dem Aufruf enthält das `TabControl` keine Tabs mehr. Existiert kein Tab mehr, besitzen Eigenschaften wie `SelectedIndex` den Wert `-1` und `SelectedTab` den Wert `$null`.

Insert()

```
$_.TabPages.Insert( Index, $tabPage )
```

Die Methode `Insert()` fügt eine `TabPage` an einer bestimmten Position innerhalb der `TabPages`-Collection ein.

Bereits vorhandene Einträge werden ab dieser Position um eine Stelle nach hinten verschoben. Dadurch kann die Reihenfolge der Tabs gezielt beeinflusst werden.

Remove()

```
$_.TabPages.Remove( $tabPage )
```

Die Methode `Remove()` entfernt eine bestimmte `TabPage` aus der `TabPages`-Collection.

Das `TabPage` selbst wird dabei nicht gelöscht, sondern lediglich aus dem `TabControl` entfernt. Es kann später erneut einer `TabPages`-Collection hinzugefügt werden.

RemoveAt()

```
$_.TabPages.RemoveAt( Index )
```

Die Methode `RemoveAt()` entfernt die `TabPage` an der angegebenen Position aus der `TabPages`-Collection.

Die verbleibenden Einträge rücken anschließend entsprechend nach vorne auf. Dadurch können sich die Indizes nachfolgender TabPages ändern.

Events

Events

- **Selecting** – Vor dem Wechsel zum TabPage
- **Selected** – Nach dem Wechsel zum TabPage
- **Deselecting** – Vor dem Verlassen vom TabPage
- **Deselected** – Nach dem Verlassen vom TabPage
- **SelectedIndexChanged** – Ausgewählter TabPage hat sich geändert
- **Click** – Mausklick auf das TabControl
- **DoubleClick** – Doppelklick auf das TabControl
- **MouseDown** – Maustaste wurde auf dem TabControl gedrückt
- **MouseMove** – Maus wurde über dem TabControl bewegt
- **MouseUp** – Gedrückte Maustaste wurde auf dem TabControl losgelassen
- **MouseEnter** – Mauszeiger betritt den Bereich des TabControl
- **MouseLeave** – Mauszeiger verlässt den Bereich des TabControl
- **DragDrop** – Element wurde per Drag&Drop auf dem TabControl abgelegt
- **KeyDown** – Taste wurde gedrückt während das TabControl den Fokus hat

- **ControlAdded** – Dem TabControl wurde ein TabPage hinzugefügt
- **ControlRemoved** – Vom TabControl wurde ein TabPage entfernt

- **Resize** – Größe des TabControl hat sich geändert
- **Paint** – TabControl wird neu gezeichnet

```
$tabControl.Add_*
  param($sender, $e)
```

- `$sender` → Das TabControl selbst (= `$this`)
- `$e` *(EventArgs)* → Argumente Tab (TabPage) entsprechend

Tabwechsel

Selecting

Vor dem Wechsel zu einem Tab (TabPage) → kann noch abgebrochen werden

- `$e.TabPage` → der Tab, zu dem gewechselt werden soll

- `$e.TabPageIndex` → Index davon
- `$e.Cancel` → kannst du auf `$true` setzen → Wechsel wird verhindert

Selected

Nach dem Wechsel zu einem Tab (TabPage)

- `$e.TabPage` → der Tab, zu dem gewechselt wurde
- `$e.TabPageIndex` → Index davon

```
$tabControl.Add_Selecting({
    param($sender, $e)

    # Ziel-Tab
    $nextTab = $e.TabPage

    if ($nextTab.Name -eq "PackageTab") {
        Write-Host "Wechsel verhindert!"

        # □□DAS ist der Trick:
        $e.Cancel = $true
    }
})
```

SelectedIndexChanged

Wird ausgelöst, **nachdem sich der ausgewählte Tab geändert hat**

- □ Kein `$e.TabPage`
- □ `$sender.SelectedTab` → aktuell aktiver Tab
- □ `$sender.SelectedIndex` → Index des aktiven Tabs

$sender, $e

`$sender`

- `$sender.SelectedTab` → aktuell aktiver Tab (`TabPage`)
- `$sender.SelectedIndex` → Index davon
- `$sender.TabPages` → alle Tabs (Collection)
- `$sender.TabCount` → Anzahl Tabs
- `$sender.Name` → Name vom Control
- `$sender.Enabled` → ob aktiv

- `$sender.Visible` → sichtbar oder nicht

`$e`

- einfach nur ein Standard-EventArgs-Objekt
- ohne nützliche Zusatzinfos

```
$tabControl.Add_SelectedIndexChanged({
    param($sender, $e)

    # Aktueller Tab
    $currentTab = $sender.SelectedTab

    Write-Host "Aktiver Tab: $($currentTab.Name)"
})
```

Deselecting

Vor dem Verlassen eines Tabs (TabPage) → kann noch abgebrochen werden

```
$tabControl.Add_Deselecting({
    param($sender, $e)

    # Aktueller Tab (der verlassen wird)
    $currentTab = $e.TabPage

    # Beispiel: Verhindere Verlassen wenn noch Auswahl vorhanden
    if ($currentTab.Name -eq "PackageTab" -and $checkedListBox.CheckedItems.Count -gt 0) {
        Write-Host "Du hast noch Auswahl!"

        $e.Cancel = $true
    }
})
```

`$e` *(EventArgs)*
- `$e.TabPage` → der Tab, der verlassen wird
- `$e.Cancel` → `$true` **= Wechsel wird verhindert**

Deselected

Nach dem Verlassen eines Tabs → ideal zum Zurücksetzen von UI

```
$tabControl.Add_Deselected({
    param($sender, $e)

    # Verlassener Tab
    $oldTab = $e.TabPage

    if ($oldTab.Name -eq "PackageTab") {

        # CheckedListBox zurücksetzen
        for ($i = 0; $i -lt $checkedListBox.Items.Count; $i++) {
            $checkedListBox.SetItemChecked($i, $false)
        }

        # ListBox zurücksetzen
        $listBox.ClearSelected()
    }
})
```

`$e` *(EventArgs)*
• `$e.TabPage` → der Tab, der verlassen wurde

□□ Konkreter Ablauf beim Tabwechsel

Wenn du von **Tab A → Tab B** wechselst:

1. **`Deselecting`** *(TabControl)*
   → bevor Tab A verlassen wird
   → **kann abgebrochen werden** (`$_.Cancel = $true`)
2. **`Selecting`** *(TabControl)*
   → bevor Tab B aktiviert wird
   → **kann ebenfalls abgebrochen werden**

> □□ Wenn hier keiner abbricht, geht’s weiter:

3. **`Deselected`** *(TabControl)*
   → Tab A wurde gerade deaktiviert
4. **`SelectedIndexChanged`** *(TabControl)*
   → der Index hat sich geändert
5. **`Selected`** *(TabControl)*
   → Tab B ist jetzt aktiv
6. **`Leave`** *(TabPage A)*
   → Fokus verlässt alten Tab

7. `Enter` *(TabPage B)*
   → Fokus betritt neuen Tab

Click

Klick auf das Control (selten relevant)

MouseDown

Klick einer beliebigen Maustaste auf dem TabControl

- `$e.Button` – gedrückte Maustaste → `[MouseButtons]::Left`
- `$e.Location` – Position des Mausklicks

ControlAdded / ControlRemoved

Wenn TabPages hinzugefügt oder entfernt werden. Wird ausgelöst, wenn ein TabPage zur TabPages-Collection
hinzugefügt oder daraus entfernt wird.

Tipps & Tricks - *TabControl*

Typische Stolperfallen

- **Tab wird nicht angezeigt**
  → nicht zur `TabPages`-Collection hinzugefügt
- **Events greifen nicht**
  → falsches Event verwendet (`Selecting` vs. `SelectedIndexChanged`)
- **Layout wirkt falsch**
  → `Dock` / `Anchor` nicht sauber gesetzt
- **Icons fehlen**
  → `ImageList` nicht gesetzt oder falscher Index

Mentales Modell

Das `TabControl` ist ein **Container mit Umschalter-Logik**.

Es zeigt genau eine `TabPage` gleichzeitig
und verwaltet nur, welche sichtbar ist.

Wann sinnvoll?

- Strukturierung komplexer Inhalte
- Einstellungen / Optionen
- Platz sparen

Wann vermeiden?

- Häufiges Hin- und Herspringen notwendig
- Linearer Workflow
- Stark voneinander abhängige Inhalte

Revision #34
Created 2026-03-18 19:55:55 UTC by John-Andreas Borinas
Updated 2026-06-06 05:50:41 UTC by John-Andreas Borinas