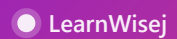


FlowLayoutPanel, TableLayoutPanel, FlexLayoutPanel und erweiterte Layout-Eigenschaften

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Flow-, Table- oder Flex-Container nach dem Verhalten des Inhalts wählen, die erweiterten Child-Eigenschaften FillWeight, FlowBreak, AlignX, AlignY, Row, Column, Spans und Styles, wann Layout-Engines Margins berücksichtigen und fluente Markup-Erweiterungen für programmatische Layouts. Was das für Wisej.NET-Entwickler bedeutet – und wie Sie dieses Modul lesen.



Fokus: Produktivbetrieb

Flow-, Table- oder Flex-Container nach dem Verhalten des Inhalts wählen, die erweiterten Child-Eigenschaften FillWeight, FlowBreak, AlignX, AlignY, Row, Column, Spans und Styles, wann Layout-Engines Margins berücksichtigen und fluente Markup-Erweiterungen für programmatische Layouts.

1. Das Problem im Produktivbetrieb

Modul 4 hat der Adaptive Operations Console eine Shell aus gedockten Bereichen gegeben, doch Docking entscheidet nur, wo ein Bereich sitzt, nicht, wie sich der Inhalt darin verhält, wenn der Browser am Montag 1400 Pixel breit ist und am Dienstag auf einem Tablet 700. Eine Reihe von Kennzahlenkarten soll umbrechen, ein Ticket-Editor soll Labels und Editoren ausgerichtet halten, wie lang die deutsche Übersetzung auch sein mag, und Listen- und Detailbereich sollen sich den Arbeitsbereich in einem Verhältnis teilen, das Sie festlegen. Jedes davon ist ein anderes Inhaltsverhalten, und Wisej.NET liefert für jedes einen Container: `FlowLayoutPanel`, `TableLayoutPanel` und `FlexLayoutPanel`. Wer den falschen wählt, schreibt am Ende wieder Resize-Code.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
FlowLayoutPanel	Ordnet Children in automatischen Zeilen oder Spalten entlang <code>FlowDirection</code> an, bricht um, wenn <code>WrapContents</code> true ist, und ignoriert Location, Dock und Anchor jedes Childs, sodass Chips und Karten sich selbst neu anordnen.
TableLayoutPanel	Platziert Children in Zellen, adressiert über Row und Column, mit Spans für Zellen, die sich erstrecken; so bleiben Labels und Editoren ausgerichtet, während jedes Child innerhalb seiner Zelle weiterhin docken oder verankern kann.
FlexLayoutPanel	Ordnet Children in einer Zeile oder einer Spalte an, füllt stets seinen Client-Bereich und verteilt den verbleibenden Platz nach Gewichtung – passend für Hauptbereiche, die den Arbeitsbereich proportional aufteilen.
FillWeight	Eine erweiterte Child-Eigenschaft, die die Flow- und Flex-Engines lesen: Ein Gewicht über null dehnt das Child im Verhältnis zu den anderen gewichteten Children in den verbleibenden Platz; null behält seine eigene Größe.
FlowBreak	Eine erweiterte Child-Eigenschaft des FlowLayoutPanel, die das nächste Child in eine neue Zeile oder Spalte zwingt, sodass eine Filterleiste eine Zeile bewusst beendet, statt sich auf die Breite zu verlassen.
Zeilen- und Spalten-Styles	<code>RowStyles</code> und <code>ColumnStyles</code> legen fest, ob eine Zeile oder Spalte ihre Größe automatisch anpasst, eine feste Pixelgröße erhält oder sich den Rest prozentual teilt; hier liegen die Proportionen eines Formulars.
Margin und Padding	Padding ist der Innenraum des Containers um alle Children; Margin ist der Abstand des Childs zu seinen Nachbarn, und nur Engines, die Margins berücksichtigen, verwenden ihn.

3. Der rote Faden der Lektion

Den Container nach dem Verhalten des Inhalts wählen

Die Entscheidungsregel ist kurz genug, um sie sich zu merken, und streng genug, um daran zu prüfen. Verwenden Sie `FlowLayoutPanel` für Inhalte, die umbrechen dürfen: Filter-Chips, Kennzahlenkarten, Überlaufbereiche von Toolbars, responsive Befehlsgruppen. Verwenden Sie `TableLayoutPanel` für strukturierte Formulare: Feld-Labels, ausgerichtete Editoren, Zusammenfassungenzeilen und jedes Raster aus Controls, das eine Lokalisierung überstehen muss. Verwenden Sie `FlexLayoutPanel` für Hauptbereiche, die den verfügbaren Platz proportional füllen müssen, etwa eine Navigationsleiste neben einem Inhaltsbereich und einem seitlichen Details-Panel oder gestapelte mobile Abschnitte, die sich die Höhe nach Gewichtung teilen. Docking und Anchoring bleiben richtig für stabile Shell-Bereiche und einfache Formulare; die drei Container sind für den Inhalt innerhalb dieser Bereiche da.

Achten Sie auf das Verhalten, nicht auf das Aussehen. Der Kennzahlenkartenbereich der Konsole sieht aus wie ein Raster, sein Verhalten ist aber „so viele Karten pro Zeile, wie hineinpassen, dann umbrechen“ – das ist Flow. Der Ticket-Editor sieht aus wie eine Liste von Feldern, sein Verhalten ist aber „die linke Kante jedes Editors auf derselben Linie“ – das ist Table. Der Arbeitsbereich sieht aus wie zwei Panels, sein Verhalten ist

aber „die Liste nimmt zwei Drittel, die Details ein Drittel, und keiner fällt zusammen“ – das ist Flex. Das Lab baut alle drei, damit Sie den Browser verkleinern und beobachten können, welcher Container welches Szenario übersteht; dieser schriftliche Vergleich ist das wichtigste Ergebnis.

Die erweiterten Child-Eigenschaften

Die Container arbeiten mit erweiterten Eigenschaften: Werten, die zum Child gehören, aber vom Parent gespeichert und gelesen werden, und die nur im Eigenschaftenraster des Childs erscheinen, solange es in diesem Container sitzt. `FlowLayoutPanel` liest `Margin`, `FillWeight`, `MinimumSize`, `MaximumSize` und `FlowBreak`, im Code über den Container gesetzt: `SetFillWeight(search, 1)` lässt das Suchfeld die gesamte in der Zeile verbleibende Breite nutzen, und `SetFlowBreak(apply, true)` beendet die Zeile, sodass die Karten darunter in einer neuen Zeile beginnen. `FlowDirection` wählt links-nach-rechts, rechts-nach-links, oben-nach-unten oder unten-nach-oben; mit `WrapContents` auf false werden die Children abgeschnitten oder gescrollt statt umbrochen, was eine einzeilige Toolbar mit `AutoScroll` braucht.

`TableLayoutPanel` liest `Row`, `Column`, `RowSpan` und `ColumnSpan` jedes Childs und bemisst das Raster über `RowStyles` und `ColumnStyles`: automatische Größe, ein absoluter Pixelwert oder ein Prozentwert. Der Ticket-Editor verwendet eine absolute Label-Spalte von 140 Pixeln und eine Editor-Spalte mit 100 Prozent, sodass die Editoren mit dem Panel wachsen, während die Labels stehen bleiben, und der Notes-Editor erstreckt sich über beide Spalten in einer Zeile, die die restliche Höhe einnimmt. `GrowStyle` entscheidet, was passiert, wenn ein Control hinzugefügt wird, nachdem alle Zellen belegt sind: Zeilen hinzufügen, Spalten hinzufügen oder das Wachsen verweigern. Weil ein Child innerhalb seiner Zelle weiterhin docken kann, ist ein Editor mit Dock Fill in einer Prozent-Spalte das sauberste ausgerichtete Formular, das Wisej.NET hervorbringt.

`FlexLayoutPanel` liest `Margin`, `MinimumSize`, `MaximumSize`, `FillWeight`, `AlignX` und `AlignY`. Sein `LayoutStyle` ist horizontal (eine Zeile) oder vertikal (eine Spalte); der Container füllt stets seinen Client-Bereich und verteilt den verbleibenden Platz unter den gewichteten Children. Füllt ein Child eine Dimension nicht aus, überschreiben `AlignX` und `AlignY` die Ausrichtung des Parents für genau dieses Child, sodass ein Detailbereich mit `AlignY` top am oberen Rand eines hohen Arbeitsbereichs anliegt, statt zentriert zu werden. Die Regel, die das Lab prüft: Jedes Child mit `FillWeight` erhält eine `MinimumSize`, sonst schrumpft ein schmaler Browser einen Bereich auf null, und der Benutzer sieht einen leeren Streifen, wo der Details-Editor war. `MaximumSize` begrenzt denselben Bereich, damit ein 4K-Monitor ein Formular nicht in Leerraum auseinanderzieht.

```
// DashboardWorkspace : FlexLayoutPanel – die Aufteilung zwischen Liste und Details
this.Dock = DockStyle.Fill;
this.LayoutStyle = FlexLayoutStyle.Horizontal;
this.Padding = new Padding(8);

var list = new Panel { MinimumSize = new Size(300, 200) };
var details = new Panel { MinimumSize = new Size(280, 200),
                          MaximumSize = new Size(520, 0) };

this.Controls.Add(list);
this.Controls.Add(details);

this.SetFillWeight(list, 2); // zwei Drittel der freien Breite
this.SetFillWeight(details, 1); // ein Drittel, nie unter 280 px
this.SetAlignY(details, VerticalAlignment.Top);
```

Wann Layout-Engines Margins berücksichtigen

Margin ist die Eigenschaft, die am häufigsten gesetzt und am seltensten beachtet wird, denn ob sie etwas bewirkt, hängt von der Engine ab, die das Child anordnet. Die Flow- und Flex-Engines setzen Margins durch: Jedes Child eines `FlowLayoutPanel` oder `FlexLayoutPanel` hält den Abstand ein, den sein Margin angibt. Die Standard-Layout-Engine verwendet Margins beim Docking nicht; ein links gedocktes Panel mit einem Margin von 8 liegt trotzdem bündig am Rand, weshalb Modul 4 für Abstände Padding an den gedockten Bereichen verwendet hat. `TableLayoutPanel` berücksichtigt Margins innerhalb der Zelle, sodass der Margin eines Childs zu seinem Abstand zum Zellrand wird, während Padding an der Tabelle den äußeren Rahmen ergänzt.

Behandeln Sie die beiden Eigenschaften so, als gehörten sie verschiedenen Eigentümern: Padding ist die Entscheidung des Containers über den Innenraum um alle seine Children, Margin die Entscheidung des Childs über seinen Abstand zu benachbarten Controls. Ein konsistentes Layout-System für die Konsole definiert Standardwerte einmal, zum Beispiel die Modi dicht, normal und großzügig mit 4, 8 und 16 Pixeln, und wendet sie überall über diese beiden Eigenschaften an. Wenn ein Abstand nicht erscheint, fragen Sie, welche Engine für das Child zuständig ist, bevor Sie irgendeine Zahl ändern.

Fluente Markup-Erweiterungen

Der Namespace `Wisej.Web.Markup` fügt fluente Erweiterungsmethoden hinzu, die sich wie eine Layout-Beschreibung lesen. `FlowLayoutPanelExtensions` können FillWeight und FlowBreak setzen, `FlexLayoutPanelExtensions` FillWeight, AlignX und AlignY, und `TableLayoutPanelExtensions` Spaltenanzahl, Wachstumsverhalten und Layout-Einstellungen. Jede Methode gibt den Container zurück, sodass eine programmatische Shell zu einer einzigen Kette wird: `panel.FillWeight(nav, 0).FillWeight(content, 1).AlignY(nav, VerticalAlignment.Top)`. Das Lab verlangt, einen Bereich auf diese Weise zu bauen; der natürliche Kandidat ist ein Bereich, den Sie ohnehin im Code erzeugen, etwa ein Dashboard, dessen Karten von der Rolle des angemeldeten Benutzers abhängen.

Verwenden Sie den fluenten Stil dort, wo er ein programmatisches Layout lesbar macht, und überlassen Sie im Designer gebaute Bildschirme dem Designer. Mischt man Designer-Serialisierung, imperative `SetFillWeight`-Aufrufe und fluente Ketten in einer Klasse, muss der nächste Entwickler drei Notationen für ein Panel lesen; bleiben Sie also bei einer Notation pro Container. In jedem Fall gilt die Checkliste:

Container statt Resize-Handler, Dock-Reihenfolge geprüft, Padding an gedockten Bereichen, Flow zum Umbrechen, Table für Formulare, Flex für proportionale Bereiche und eine MinimumSize an jedem Control mit FillWeight.



Typische Fehler

Location, Dock oder Anchor an einem Child eines FlowLayoutPanel setzen und sich wundern, warum es ignoriert wird. Bereiche mit FillWeight ohne MinimumSize, die auf dem Smartphone zusammenfallen. Erwarten, dass Margin unter der Standard-Layout-Engine Abstände zwischen gedockten Panels erzeugt. Zum FlexLayoutPanel für ein Formular mit ausgerichteten Labels greifen oder zum TableLayoutPanel für eine Reihe von Karten, die umbrechen soll.



Gewohnheit für das visuelle System

Nutzen Sie den Wisej.NET-Mechanismus, der für das Problem zuständig ist: das Theme für die Identität der Standard-Controls, AppearanceKey für semantische Varianten, CssClass für bereichsbezogenes anwendungsspezifisches CSS, CssStyle nur für einmalige dynamische Werte, Layout-Container statt Resize-Code, responsive Eigenschaften für im Designer gesetzte Profilwerte und ResponsiveProfileChanged für Verhalten, das nicht rein visuell ist.



Lernziele

Am Ende dieses Moduls können Sie:

- ✓ FlowLayoutPanel, TableLayoutPanel oder FlexLayoutPanel anhand des Inhaltsverhaltens und des Zwecks des Bildschirms auswählen.
- ✓ FillWeight, FlowBreak, AlignX, AlignY, Row, Column, RowSpan, ColumnSpan, RowStyles, ColumnStyles, GrowStyle, MinimumSize und MaximumSize verwenden.
- ✓ verstehen, wann Layout-Engines Margins verwenden und wann nicht.
- ✓ Markup-Erweiterungsmethoden, wo sinnvoll, für fluente Layout-Definitionen einsetzen.

Lab- und Prüfungsleitfaden

Was Sie im praktischen Lab bauen, das vorgeschlagene Vorgehen und die geforderten Ergebnisse.



Praktisches Lab

Bauen Sie drei Varianten des Inhaltsbereichs der Adaptive Operations Console: einen Filter- und Kennzahlenkartenbereich mit `FlowLayoutPanel`, der umbricht und `FillWeight` sowie `FlowBreak` nutzt, einen Ticket-Editor mit `TableLayoutPanel`, dessen Labels und Editoren über `RowStyles` und `ColumnStyles` ausgerichtet sind, und eine Dashboard-Shell mit `FlexLayoutPanel`, in der sich Listen- und Detailbereich den Platz per `FillWeight` teilen, mit gesetztem `AlignX`, `AlignY`, `MinimumSize` und `MaximumSize`. Bauen Sie einen der drei Bereiche im Code mit den fluenten Erweiterungen aus `Wisej.Web.Markup` und vergleichen Sie, wie jeder eine Größenänderung des Browsers übersteht.

Ausgangspunkt

Öffnen Sie die Solution `AdaptiveOperationsConsole` so, wie Sie sie nach Modul 4 verlassen haben: Die Haupt-Page ist eine Shell aus verschachtelten Containern mit Toolbar, Navigationsleiste, füllendem Arbeitsbereich, Details-Panel und Statusbereich, Navigations- und Detailbereich sind `UserControls`, und der Arbeitsbereich ist noch ein einfaches `Panel`, dessen Kennzahlenkarten, Ticket-Editor und Liste von Hand positioniert werden. Legen Sie die drei neuen Bereiche in einem Ordner `Layout` ab, damit das Abschlussprojekt den Gewinner jedes Vergleichs übernehmen kann.

Vorgeschlagenes Vorgehen

- ✓ Erstellen Sie eine `FilterBar`, die von `FlowLayoutPanel` abgeleitet ist, mit `FlowDirection` links-nach-rechts und eingeschaltetem `WrapContents`; fügen Sie Suchfeld, Status-Combo und Apply-Button hinzu, geben Sie dem Suchfeld `FillWeight` 1 und eine `MinimumSize`, setzen Sie `FlowBreak` am Apply-Button, platzieren Sie dann die Kennzahlenkarten darunter und beobachten Sie, wie sie umbrechen.
- ✓ Bauen Sie ein `UserControl` `TicketEditor` um ein zweispaltiges `TableLayoutPanel`: eine absolute Label-Spalte von 140 Pixeln, eine Editor-Spalte mit 100 Prozent, eine Zeile mit automatischer Größe pro Feld, eine Notes-Zeile, die sich mit der restlichen Höhe über beide Spalten erstreckt, und jeder Editor mit Dock Fill in seiner Zelle.
- ✓ Ersetzen Sie das Arbeitsbereichs-Panel durch einen `DashboardWorkspace`, der von `FlexLayoutPanel` im horizontalen Layout abgeleitet ist; fügen Sie Listen- und Detailbereich mit `FillWeight` 2 und 1 hinzu, mit einer `MinimumSize` an beiden, einer `MaximumSize` an den Details und `AlignY` top an den Details.
- ✓ Bauen Sie einen der drei Bereiche vollständig im Code mit `using Wisej.Web.Markup;` und den fluenten Erweiterungsmethoden `FillWeight`, `FlowBreak`, `AlignX` und `AlignY`, verkettet am Container, und halten Sie diese Klasse frei von Designer-Serialisierung.
- ✓ Starten Sie die Konsole, verkleinern Sie den Browser für jeden Bereich von 1400 über 1024 auf 700 Pixel, machen Sie bei jeder Breite einen Screenshot und notieren Sie pro Bereich, welcher Container standgehalten hat und welche erweiterte Eigenschaft die Arbeit erledigt hat.

Geforderte Ergebnisse

- ✓ Filter- und Kennzahlenkartenbereich mit FlowLayoutPanel unter Verwendung von WrapContents, FillWeight und FlowBreak
- ✓ Ticket-Editor mit TableLayoutPanel, RowStyles, ColumnStyles und Spans für ausgerichtete Felder
- ✓ Dashboard-Shell mit FlexLayoutPanel, FillWeight, AlignX, AlignY, MinimumSize und MaximumSize
- ✓ Ein Bereich im Code mit den fluenten Erweiterungen aus Wisej.Web.Markup gebaut
- ✓ Screenshots zum Größenänderungsvergleich und eine Notiz, welcher Container zu welchem Bereich passt

Wiederholungsfragen

- ✓ Sie haben dem Detailbereich FillWeight 1, aber keine MinimumSize gegeben, und bei 700 Pixeln ist er verschwunden. Erklären Sie, was die Flex-Engine mit dem Platz gemacht hat und warum die Lösung eine Größenbeschränkung ist und kein anderes Gewicht.
- ✓ Der Apply-Button bricht bei 1024 Pixeln in die zweite Zeile um, obwohl Platz für ihn wäre. Welche erweiterte Eigenschaft hat das bewirkt, und wie würde sich die FilterBar verhalten, wenn Sie sie entfernen und WrapContents auf false setzen?
- ✓ Ein Margin von 8 an den Karten im FlowLayoutPanel hat Abstände erzeugt, derselbe Margin an der gedockten Navigationsleiste hat nichts bewirkt. Welche Engine berücksichtigt Margins im jeweiligen Fall, und welche Eigenschaft erzeugt den Abstand für die Leiste?



Gewohnheit für das visuelle System

Nutzen Sie den Wisej.NET-Mechanismus, der für das Problem zuständig ist: das Theme für die Identität der Standard-Controls, AppearanceKey für semantische Varianten, CssClass für bereichsbezogenes anwendungsspezifisches CSS, CssStyle nur für einmalige dynamische Werte, Layout-Container statt Resize-Code, responsive Eigenschaften für im Designer gesetzte Profilwerte und ResponsiveProfileChanged für Verhalten, das nicht rein visuell ist.