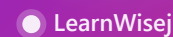


FlowLayoutPanel, TableLayoutPanel, FlexLayoutPanel e proprietà di layout estese

Guida della lezione · Guida al lab / esame



Guida della lezione

Scegliere contenitori flow, table o flex in base al comportamento del contenuto, le proprietà estese dei figli FillWeight, FlowBreak, AlignX, AlignY, Row, Column, span e stili, quando i motori di layout rispettano i margini, e le estensioni di markup fluent per i layout costruiti da codice. Che cosa significa per uno sviluppatore Wisej.NET — e come affrontare questo modulo.



Focus di produzione

Scegliere contenitori flow, table o flex in base al comportamento del contenuto, le proprietà estese dei figli FillWeight, FlowBreak, AlignX, AlignY, Row, Column, span e stili, quando i motori di layout rispettano i margini, e le estensioni di markup fluent per i layout costruiti da codice.

1. Il problema in produzione

Il Modulo 4 ha dato ad Adaptive Operations Console uno shell di regioni in dock, ma il docking decide solo dove sta una regione, non come si comporta il contenuto al suo interno quando il browser è largo 1400 pixel il lunedì e 700 su un tablet il martedì. Una riga di card delle metriche dovrebbe andare a capo, un editor dei ticket dovrebbe mantenere allineati label ed editor per quanto lunga sia la traduzione tedesca, e le regioni della lista e dei dettagli dovrebbero dividersi l'area di lavoro nella proporzione che decidi tu. Ognuno è un comportamento del contenuto diverso, e Wisej.NET fornisce un contenitore per ciascuno: `FlowLayoutPanel`, `TableLayoutPanel` e `FlexLayoutPanel`. Scegliere quello sbagliato è il modo in cui i team finiscono per scrivere di nuovo codice di ridimensionamento.

2. Concetti chiave

Concetto	Perché è importante
FlowLayoutPanel	Dispone i figli in righe o colonne automatiche lungo <code>FlowDirection</code> , va a capo quando <code>WrapContents</code> è true e ignora Location, Dock e Anchor di ogni figlio, così chip e card si ridispongono da soli.
TableLayoutPanel	Colloca i figli in celle indicate da Row e Column, con span per le celle che si estendono, così label ed editor restano allineati mentre ogni figlio può ancora essere messo in dock o ancorato dentro la propria cella.
FlexLayoutPanel	Dispone i figli in un'unica riga o colonna, riempie sempre la sua area client e divide lo spazio rimanente per peso, il che si adatta alle regioni principali che suddividono l'area di lavoro in proporzione.
FillWeight	Una proprietà estesa dei figli letta dai motori flow e flex: un peso maggiore di zero allarga il figlio nello spazio rimanente in proporzione agli altri figli con peso; zero mantiene la sua dimensione.
FlowBreak	Una proprietà estesa dei figli di FlowLayoutPanel che forza il figlio successivo su una nuova riga o colonna, così una barra dei filtri termina una riga in modo deliberato invece di affidarsi alla larghezza.
Stili di riga e di colonna	<code>RowStyles</code> e <code>ColumnStyles</code> dichiarano se una riga o una colonna si dimensiona automaticamente, prende una dimensione fissa in pixel o divide il resto in percentuale; è qui che vivono le proporzioni di un form.
Margin e Padding	Il Padding è lo spazio interno del contenitore attorno a tutti i figli; il Margin è la distanza del figlio dai vicini, e solo i motori che rispettano i margini lo useranno.

3. Il filo della lezione

Scegli il contenitore in base al comportamento del contenuto

La regola di decisione è abbastanza breve da ricordarla e abbastanza rigorosa da usarla in una revisione. Usa `FlowLayoutPanel` per il contenuto che può andare a capo: chip dei filtri, card delle metriche, aree di overflow della toolbar, gruppi di comandi responsive. Usa `TableLayoutPanel` per i form strutturati: label dei campi, editor allineati, righe di riepilogo e qualsiasi griglia di controlli che deve sopravvivere alla localizzazione. Usa `FlexLayoutPanel` per le regioni principali che devono riempire lo spazio disponibile in proporzione, come una barra di navigazione accanto a una regione dei contenuti e un pannello laterale dei dettagli, o sezioni mobili impilate che si dividono l'altezza per peso. Docking e ancoraggio restano giusti per le regioni stabili dello shell e per i form semplici; i tre contenitori servono per il contenuto dentro quelle regioni.

Guarda il comportamento, non l'aspetto. L'area delle card delle metriche della console sembra una griglia, ma il suo comportamento è "tante card per riga quante ne entrano, poi a capo", cioè flow. L'editor dei ticket sembra un elenco di campi, ma il suo comportamento è "il bordo sinistro di ogni editor sulla stessa linea", cioè table. L'area di lavoro sembra due pannelli, ma il suo comportamento è "la lista prende due

terzi, i dettagli un terzo, e nessuno dei due collassa", cioè flex. Il lab li costruisce tutti e tre così puoi ridimensionare il browser e osservare quale contenitore regge quale scenario; quel confronto scritto è il deliverable che conta di più.

Le proprietà estese dei figli

I contenitori lavorano tramite proprietà estese: valori che appartengono al figlio ma sono memorizzati e letti dal genitore, e che compaiono nella griglia delle proprietà del figlio solo mentre si trova dentro quel contenitore. `FlowLayoutPanel` legge `Margin`, `FillWeight`, `MinimumSize`, `MaximumSize` e `FlowBreak`, impostati nel codice tramite il contenitore: `SetFillWeight(search, 1)` permette alla casella di ricerca di usare tutta la larghezza che resta nella riga, e `SetFlowBreak(apply, true)` chiude la riga così che le card sotto inizino su una riga nuova. `FlowDirection` sceglie da sinistra a destra, da destra a sinistra, dall'alto in basso o dal basso in alto; con `WrapContents` false i figli vengono tagliati o fatti scorrere invece di andare a capo, ed è quello che vuole una toolbar su una sola riga con `AutoScroll`.

`TableLayoutPanel` legge `Row`, `Column`, `RowSpan` e `ColumnSpan` da ogni figlio e dimensiona la griglia a partire da `RowStyles` e `ColumnStyles`: dimensionamento automatico, un valore assoluto in pixel o una percentuale. L'editor dei ticket usa una colonna delle label assoluta da 140 pixel e una colonna degli editor al 100 per cento, così gli editor crescono con il pannello mentre le label restano ferme, e l'editor Notes si estende su entrambe le colonne in una riga che prende l'altezza rimanente. `GrowStyle` decide che cosa succede quando si aggiunge un controllo dopo che le celle sono piene: aggiungere righe, aggiungere colonne o rifiutarsi di crescere. Dato che un figlio può ancora essere messo in dock dentro la sua cella, un editor in dock su Fill in una colonna in percentuale è il form allineato più pulito che Wisej.NET possa produrre.

`FlexLayoutPanel` legge `Margin`, `MinimumSize`, `MaximumSize`, `FillWeight`, `AlignX` e `AlignY`. Il suo `LayoutStyle` è orizzontale (una riga) o verticale (una colonna); il contenitore riempie sempre la sua area client e distribuisce lo spazio rimanente tra i figli con peso. Quando un figlio non riempie una dimensione, `AlignX` e `AlignY` ridefiniscono l'allineamento del genitore solo per quel figlio, così una regione dei dettagli con `AlignY` in alto resta attaccata al bordo superiore di un'area di lavoro alta invece di centrarsi. La regola che il lab verifica: ogni figlio con `FillWeight` riceve un `MinimumSize`, altrimenti un browser stretto riduce una regione a zero e l'utente vede una striscia vuota dove prima c'era l'editor dei dettagli. `MaximumSize` limita la stessa regione così che un monitor 4K non allarghi un form in uno spazio vuoto.

```
// DashboardWorkspace : FlexLayoutPanel - la divisione tra lista e dettagli
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); // due terzi della larghezza libera
this.SetFillWeight(details, 1); // un terzo, mai sotto i 280 px
this.SetAlignY(details, VerticalAlignment.Top);
```

Quando i motori di layout rispettano i margini

Margin è la proprietà impostata più spesso e rispettata più di rado, perché il fatto che abbia un effetto dipende dal motore che dispone il figlio. I motori flow e flex applicano i margini: ogni figlio di una `FlowLayoutPanel` o `FlexLayoutPanel` mantiene la distanza dichiarata dal suo Margin. Il motore di layout predefinito non usa i margini per il docking; un Panel in dock a Left con un Margin di 8 resta comunque attaccato al bordo, ed è per questo che il Modulo 4 ha usato il Padding sulle regioni in dock per creare gli spazi. `TableLayoutPanel` tiene conto dei margini dentro la cella, quindi il Margin di un figlio diventa la sua distanza dal bordo della cella mentre il Padding sulla tabella aggiunge la cornice esterna.

Tratta le due proprietà come appartenenti a proprietari diversi: il Padding è la decisione del contenitore sullo spazio interno attorno a tutti i suoi figli, il Margin la decisione del figlio sulla sua distanza dai controlli adiacenti. Un sistema di layout coerente per la console definisce una volta sola dei valori standard, per esempio modalità compatta, normale e spaziosa da 4, 8 e 16 pixel, e li applica ovunque tramite quelle due proprietà. Quando uno spazio non compare, chiediti quale motore possiede il figlio prima di cambiare qualsiasi numero.

Estensioni di markup fluent

Il namespace `Wisej.Web.Markup` aggiunge metodi di estensione fluent che si leggono come una descrizione del layout. `FlowLayoutPanelExtensions` può impostare `FillWeight` e `FlowBreak`, `FlexLayoutPanelExtensions` può impostare `FillWeight`, `AlignX` e `AlignY`, e `TableLayoutPanelExtensions` può impostare il numero di colonne, il grow style e le impostazioni di layout. Ogni metodo restituisce il contenitore, così uno shell costruito da codice diventa un'unica catena: `panel.FillWeight(nav, 0).FillWeight(content, 1).AlignY(nav, VerticalAlignment.Top)`. Il lab ti chiede di costruire così una regione; il candidato naturale è una regione che crei già nel codice, come una dashboard le cui card dipendono dal ruolo dell'utente che ha effettuato l'accesso.

Usa lo stile fluent dove rende leggibile un layout costruito da codice e lascia al designer le schermate costruite nel designer. Mescolare in una classe la serializzazione del designer, chiamate imperative a `SetFillWeight` e catene fluent costringe il prossimo sviluppatore a leggere tre notazioni per un solo pannello, quindi tieni una notazione per contenitore. In ogni caso vale la checklist: contenitori invece di gestori di ridimensionamento, ordine di docking controllato, Padding sulle regioni in dock, flow per andare a capo, table per i form, flex per le regioni proporzionali, e un `MinimumSize` su ogni controllo con `FillWeight`.



Errori tipici

Impostare `Location`, `Dock` o `Anchor` su un figlio di una `FlowLayoutPanel` e chiedersi perché venga ignorato. Regioni con `FillWeight` senza un `MinimumSize` che collassano su un telefono. Aspettarsi che `Margin` crei spazi tra Panel in dock con il motore di layout predefinito. Scegliere `FlexLayoutPanel` per un form con label allineate, o `TableLayoutPanel` per una riga di card che dovrebbe andare a capo.



La regola del sistema visivo

Usa il meccanismo Wisej.NET che possiede il problema: il tema per l'identità dei controlli standard, AppearanceKey per le varianti semantiche, CssClass per il CSS circoscritto e specifico dell'applicazione, CssStyle solo per valori dinamici una tantum, i contenitori di layout invece del codice di ridimensionamento, le proprietà responsive per i valori di profilo impostati nel designer, e ResponsiveProfileChanged per il comportamento che non è semplicemente visivo.



Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Scegliere FlowLayoutPanel, TableLayoutPanel o FlexLayoutPanel in base al comportamento del contenuto e allo scopo della schermata.
- ✓ Usare FillWeight, FlowBreak, AlignX, AlignY, Row, Column, RowSpan, ColumnSpan, RowStyles, ColumnStyles, GrowStyle, MinimumSize e MaximumSize.
- ✓ Capire quando i motori di layout usano i margini e quando no.
- ✓ Usare i metodi di estensione di markup per definizioni di layout fluent dove è appropriato.

Guida al lab / esame

Che cosa costruirai nel lab pratico, l'approccio suggerito e i deliverable richiesti.



Lab pratico

Costruisci tre versioni dell'area contenuti di Adaptive Operations Console: un'area di filtri e card delle metriche con `FlowLayoutPanel` che va a capo e usa `FillWeight` e `FlowBreak`, un editor dei ticket con `TableLayoutPanel` con label ed editor allineati governati da `RowStyles` e `ColumnStyles`, e uno shell della dashboard con `FlexLayoutPanel` in cui le regioni della lista e dei dettagli si dividono lo spazio tramite `FillWeight`, con `AlignX`, `AlignY`, `MinimumSize` e `MaximumSize` impostati. Costruisci una delle tre da codice con le estensioni fluent di `Wisej.Web.Markup` e confronta come ciascuna regge un ridimensionamento del browser.

Punto di partenza

Apri la soluzione `AdaptiveOperationsConsole` come l'hai lasciata dopo il Modulo 4: la Page principale è uno shell di contenitori annidati con una toolbar, una barra di navigazione, un'area di lavoro su Fill, un pannello dei dettagli e una regione di stato, le regioni di navigazione e dettagli sono `UserControl`, e l'area di lavoro è ancora un semplice `Panel` in cui card delle metriche, editor dei ticket e lista sono posizionati a mano. Tieni le tre nuove regioni in una cartella `Layout` così che il capstone possa scegliere la vincitrice di ogni confronto.

Approccio suggerito

- ✓ Crea una `FilterBar` che deriva da `FlowLayoutPanel` con `FlowDirection` da sinistra a destra e `WrapContents` attivo; aggiungi la casella di ricerca, la combo dello stato e il pulsante Apply, dai alla casella di ricerca `FillWeight` 1 e un `MinimumSize`, imposta `FlowBreak` su Apply, poi inserisci sotto le card delle metriche e guardale andare a capo.
- ✓ Costruisci uno `UserControl` `TicketEditor` attorno a una `TableLayoutPanel` a due colonne: una colonna delle label assoluta da 140 pixel, una colonna degli editor al 100 per cento, una riga a dimensionamento automatico per campo, una riga Notes che si estende su entrambe le colonne con l'altezza rimanente, e ogni editor in dock su Fill nella sua cella.
- ✓ Sostituisci il Panel dell'area di lavoro con un `DashboardWorkspace` che deriva da `FlexLayoutPanel` con layout orizzontale; aggiungi le regioni della lista e dei dettagli con `FillWeight` 2 e 1, un `MinimumSize` su entrambe, un `MaximumSize` sui dettagli e `AlignY` in alto sui dettagli.
- ✓ Costruisci una delle tre regioni interamente da codice con `using Wisej.Web.Markup;` e i metodi di estensione fluent `FillWeight`, `FlowBreak`, `AlignX` e `AlignY` concatenati sul contenitore, e tieni quella classe libera dalla serializzazione del designer.
- ✓ Esegui la console, ridimensiona il browser da 1400 a 1024 a 700 pixel per ogni regione, cattura uno screenshot a ogni larghezza e annota, regione per regione, quale contenitore ha retto e quale proprietà estesa ha fatto il lavoro.

Deliverable richiesti

- ✓ Area di filtri e card delle metriche con FlowLayoutPanel che usa WrapContents, FillWeight e FlowBreak
- ✓ Editor dei ticket con TableLayoutPanel con RowStyles, ColumnStyles e span per campi allineati
- ✓ Shell della dashboard con FlexLayoutPanel con FillWeight, AlignX, AlignY, MinimumSize e MaximumSize
- ✓ Una regione costruita da codice con le estensioni fluent di Wisej.Web.Markup
- ✓ Screenshot di confronto al ridimensionamento e una nota su quale contenitore si adatta a ogni regione

Domande di revisione per lo studente

- ✓ Hai dato alla regione dei dettagli FillWeight 1 ma nessun MinimumSize, e a 700 pixel è sparita. Spiega che cosa ha fatto il motore flex con lo spazio e perché la correzione è un vincolo di dimensione e non un peso diverso.
- ✓ Il pulsante Apply va a capo sulla seconda riga a 1024 pixel anche se ci sarebbe spazio. Quale proprietà estesa lo ha causato, e come si comporterebbe diversamente la FilterBar se la togliessi e impostassi WrapContents a false?
- ✓ Un Margin di 8 sulle card nella FlowLayoutPanel ha prodotto degli spazi, ma lo stesso Margin sulla barra di navigazione in dock non ha avuto effetto. Quale motore rispetta i margini in ciascun caso, e quale proprietà crea lo spazio per la barra di navigazione?



La regola del sistema visivo

Usa il meccanismo Wisej.NET che possiede il problema: il tema per l'identità dei controlli standard, AppearanceKey per le varianti semantiche, CssClass per il CSS circoscritto e specifico dell'applicazione, CssStyle solo per valori dinamici una tantum, i contenitori di layout invece del codice di ridimensionamento, le proprietà responsive per i valori di profilo impostati nel designer, e ResponsiveProfileChanged per il comportamento che non è semplicemente visivo.