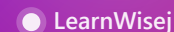


Contenitori, layout, navigazione e riuso

Guida della lezione · Guida al lab / esame



Guida della lezione

Ordine di docking, ancoraggio, layout a flusso e a tabella, proprietà responsive per profilo client, superfici di navigazione e comando, e gli UserControl come unità di riuso. Che cosa significa per uno sviluppatore Wisej.NET — e come affrontare questo modulo.



Focus di produzione

Ordine di docking, ancoraggio, layout a flusso e a tabella, proprietà responsive per profilo client, superfici di navigazione e comando, e gli UserControl come unità di riuso.

1. Il problema in produzione

I contenitori sono l'architettura dell'interfaccia: decidono come i controlli vengono raggruppati, ridimensionati, navigati e riutilizzati. La Operations Console del Modulo 1 ha ancora pannelli segnaposto, e una schermata in cui ogni controllo è posizionato a mano diventa fragile al primo ridimensionamento del browser. I problemi di layout sono quasi sempre problemi di progettazione, non di pixel, e questo modulo sostituisce lo spostamento di pixel con un piano: regioni in dock, uno SplitContainer, un TabControl, un vero modello a comandi e UserControl che nascondono i propri interni.

2. Concetti chiave

Concetto	Perché è importante
ordine di docking	Il docking aggancia un figlio a un lato del contenitore o gli fa riempire quello che resta, e viene applicato nell'ordine dei figli: gli stessi quattro controlli producono shell diversi a seconda dell'ordine in cui sono stati aggiunti.
ancoraggio	L'ancoraggio lega i bordi di un controllo a quelli del contenitore; ancorato a sinistra e a destra si allarga con la larghezza, non ancorato su nessun asse resta "sospeso" mantenendo la posizione relativa.
SplitContainer	Due regioni adiacenti che l'utente può ridimensionare: la scelta giusta ogni volta che deve essere la persona, e non lo sviluppatore, a decidere quanto spazio dare all'albero di navigazione o al pannello di dettaglio.
pannelli di layout	FlowLayoutPanel dispone i figli uno accanto all'altro e va a capo, TableLayoutPanel allinea righe e colonne, e FlexLayoutPanel distribuisce e manda a capo come un layout responsive moderno.
proprietà responsive	Una proprietà che contiene un valore diverso per ciascun profilo client, impostata a design time: un layout desktop può così cambiare una manciata di proprietà sugli schermi stretti senza bisogno di una seconda schermata.
modello a comandi	Una regola coerente su dove sta un comando: MenuBar, ToolBar, RibbonBar, ContextMenu o StatusBar, ciascuno con il proprio compito, tutti che invocano comandi applicativi invece di contenere logica di business.
superficie dello UserControl	L'unità di riuso: un controllo composto che espone poche proprietà, eventi e metodi significativi invece di tutti i controlli figli, così che chi lo usa non si accoppi alla sua implementazione.

3. Il filo della lezione

Scegli il contenitore adatto al compito

Wisej.NET ti offre posizionamento assoluto, docking, ancoraggio e pannelli di layout più ricchi, e la prima decisione è quale superficie ospita la schermata. Una **Page** è la superficie d'ingresso naturale per un'applicazione in stile web come la Operations Console: riempie il browser e non ha cornice di finestra. Un **Form** è adatto a finestre di dialogo, flussi modali e pattern in stile MDI, e un **Desktop** può ospitare più form quando l'applicazione vuole deliberatamente conservare la metafora del desktop. Dentro quella superficie, **Panel** e **GroupBox** raggruppano controlli correlati; l'unica differenza è se il gruppo merita o no un titolo visibile.

Tre contenitori fanno quasi tutto il lavoro strutturale. Uno **SplitContainer** è ideale quando l'utente può ridimensionare due regioni adiacenti, per esempio un albero di navigazione accanto a un pannello di dettaglio. Un **TabControl** organizza pagine di pari livello, così l'utente si sposta tra le sezioni senza aprire nuove finestre. Un **Accordion** impila sezioni richiudibili per navigazione, impostazioni e aree di lavoro

comprimibili. Nel lab la regione sinistra della console diventa il pannello di uno `SplitContainer`, la regione destra un `TabControl`, e i segnaposto del Modulo 1 diventano le sue tab page.

Ordine di docking, ancoraggio e pannelli di layout

Il docking aggancia un controllo figlio a un lato del contenitore (`DockStyle.Top`, `Bottom`, `Left`, `Right`) oppure gli fa riempire lo spazio rimanente (`DockStyle.Fill`). Poiché il docking viene applicato nell'ordine dei figli, l'ordine in cui i controlli vengono aggiunti conta: la ToolBar e la StatusBar devono prendersi i loro bordi prima che lo SplitContainer riempi il resto, altrimenti il controllo in Fill viene disposto per primo e le barre gli finiscono sopra. Quando uno shell sembra sbagliato nel designer, controlla l'ordine dei figli prima di toccare una sola dimensione.

L'ancoraggio lega i bordi di un controllo a quelli del contenitore. Un controllo ancorato a sinistra e a destra cambia larghezza quando cambia quella del contenitore; un controllo non ancorato né in orizzontale né in verticale resta sospeso mantenendo la posizione relativa — giusto per una card di login centrata, sbagliato per una griglia. Usa il docking per le regioni applicative (navigazione, intestazione, piè di pagina o stato, contenuti) e l'ancoraggio dentro quelle regioni per editor e pulsanti.

Per insiemi di controlli, usa un pannello di layout invece delle coordinate. `FlowLayoutPanel` dispone i figli uno accanto all'altro e può andare a capo, così una riga di chip di filtro sopravvive a un browser stretto.

`TableLayoutPanel` dispone il contenuto in righe e colonne, che è esattamente ciò che serve a una maschera con label ed editor allineati. `FlexLayoutPanel` distribuisce lo spazio e va a capo come i layout responsive moderni: la scelta migliore quando l'interfaccia ha bisogno di una distribuzione flessibile più che di una griglia fissa.

In Wisej.NET il design responsive non si limita alle media query CSS. Le proprietà responsive contengono più valori per profili client diversi e si impostano a design time nella property grid; i profili client identificano situazioni di dispositivo o browser, per esempio un telefono fino a 600 pixel di larghezza, un tablet fino a 1024 e un desktop oltre. La regola pratica è costruire prima un buon layout desktop, poi definire la manciata di proprietà che devono cambiare per tablet o schermi stretti: visibilità, dimensione, docking, colonne mostrate, comportamento del pannello laterale e posizione dei comandi. Non creare una schermata completamente separata a meno che il flusso di lavoro non cambi davvero. Quando una proprietà da sola non basta, una logica basata sul profilo scritta in codice fa lo stesso lavoro, come mostra il metodo per il profilo stretto del lab:

```
// Shell di OperationsConsole: aggiungi le regioni in quest'ordine
this.toolBar.Dock = DockStyle.Top;
this.statusBar.Dock = DockStyle.Bottom;
this.splitMain.Dock = DockStyle.Fill; // navigazione | dettaglio TabControl

private void ApplyNarrowProfile(bool narrow)
{
    // schermi stretti: nascondi il pannello di navigazione, mostra il pulsante menu
    this.splitMain.Panel1Collapsed = narrow;
    this.btnMobileMenu.Visible = narrow;
    this.pnlProfile.Text = narrow ? "Profilo stretto" : "Profilo desktop"; // un pannello
    della StatusBar
}
```

Superfici di navigazione e di comando

I controlli di navigazione hanno bisogno di un modello a comandi coerente, e il modello riguarda dove sta un comando, non che aspetto ha. Una `MenuBar` è adatta ai menu applicativi tradizionali. Una `ToolBar` contiene le azioni usate di frequente, ed è per questo che la Operations Console ci mette New, Refresh e Save. Una `RibbonBar` raggruppa i comandi per area di attività quando l'applicazione ne ha molti — eccessiva per una console che ne ha una dozzina. Un `ContextMenu` serve per i comandi legati a un oggetto specifico: una riga, un nodo o la selezione corrente. In ogni caso il controllo solleva un evento e un gestore sottile invoca un comando applicativo; i controlli di navigazione che contengono logica di business sono uno degli errori tipici del modulo.

La `StatusBar` non è una discarica. Deve mostrare informazioni rilevanti come il numero di record selezionati, l'orario dell'ultimo aggiornamento, lo stato della validazione o il profilo client attivo, e il lab chiede stato reale, non la parola "Pronto". Un `TabControl` merita la stessa disciplina. Le tab page non sostituiscono un flusso di lavoro: usale per sezioni di pari livello come Riepilogo, Dettagli, Storico, Documenti e Note, mai come passi obbligati di una procedura guidata. Non nascondere campi obbligatori dentro i tab senza una navigazione chiara della validazione e, se i tab si possono spostare, decidi se il nuovo ordine viene conservato e se riguarda una sola sessione o il profilo dell'utente.

Lo UserControl è l'unità di riuso

Quando lo stesso gruppo di controlli compare due volte, la tentazione è copiarlo, e il conto arriva con la terza copia e la prima modifica. Uno `UserControl` è la principale unità di riuso. Un controllo riutilizzabile deve avere una superficie pubblica ridotta: proprietà come `CustomerId`, `ReadOnly` e `SelectedDate`, eventi come `Saved` o `SelectionChanged`, metodi come `LoadData()` o `ValidateContent()`. Evita di esporre tutti i controlli figli interni, a meno che non si tratti di un controllo avanzato progettato apposta; altrimenti le schermate che lo usano si accoppiano all'implementazione, e rinominare una `TextBox` rompe tre pagine. Il lab sposta la striscia di stato e l'intestazione di record ripetute della console dentro `UserControl`; la domanda di verifica è se chi li usa potrebbe farlo senza sapere che cosa c'è dentro.

Metti per iscritto le decisioni. Il `LayoutNotes.md` del lab è un file breve che tieni nel progetto e che annota quali regioni sono in dock e in quale ordine, quali controlli sono ancorati e perché, e quali proprietà cambiano sul profilo stretto. È quello che il prossimo sviluppatore legge prima di aprire il designer, e deve riflettere queste regole di decisione sul layout:

- Usa il docking per le regioni applicative e uno `SplitContainer` quando deve essere l'utente a controllare la ripartizione dello spazio.
- Usa `TabControl` per sezioni di pari livello, `FlowLayoutPanel` per insiemi che vanno a capo e `TableLayoutPanel` per maschere con label ed editor allineati.
- Usa le proprietà responsive per i cambiamenti legati al profilo e incapsula i layout ripetuti in `UserControl` invece di copiare gruppi di controlli.



Errori tipici

Mescolare posizionamento assoluto, docking e flusso senza un piano. Impostare `AutoSize` su contenitori o griglie di grandi dimensioni e provocare senza accorgersene calcoli di layout enormi. Creare tab che nascondono campi obbligatori e poi mostrare errori di validazione vaghi. Esporre pubblicamente ogni controllo figlio di uno `UserControl`. Rendere i controlli di navigazione responsabili della logica di business invece di far loro invocare comandi applicativi.



La regola da libreria di controlli

Scegli il controllo nativo più semplice che esprime il compito dell'utente prima di scrivere codice personalizzato. I controlli nativi ti danno già supporto a design time, temi, eventi e stato sul server; il passo successivo è un extender o uno `UserControl`, e solo quando nessun controllo nativo è adatto si costruisce un widget JavaScript.



Obiettivi di apprendimento

Alla fine di questo modulo sarai in grado di:

- ✓ Usare correttamente `Form`, `Page`, `Desktop`, `Panel`, `GroupBox`, `SplitContainer`, `TabControl`, `Accordion`, `FlowLayoutPanel`, `TableLayoutPanel` e `UserControl`.
- ✓ Combinare docking, ancoraggio, flusso, tabella e proprietà responsive senza combattere con il motore di layout.
- ✓ Progettare superfici di navigazione e comando usando `MenuBar`, `ToolBar`, `RibbonBar`, `StatusBar`, `ContextMenu` e `TabControl`.
- ✓ Impacchettare i pattern di interfaccia ricorrenti in `UserControl` con proprietà ed eventi pubblici puliti.

Guida al lab / esame

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



Lab pratico

Sostituisci i segnaposto del Modulo 1 con lo shell vero di Operations Console: uno SplitContainer con la navigazione a sinistra, una superficie di dettaglio TabControl a destra, una ToolBar per i comandi frequenti e una StatusBar che mostra lo stato reale. Sposta l'interfaccia ripetuta dentro UserControl che espongono solo proprietà ed eventi pubblici significativi, realizza un layout per il profilo stretto con proprietà responsive o logica basata sul profilo, e documenta le tue scelte di docking e ancoraggio in LayoutNotes.md.

Punto di partenza

Apri la soluzione `OperationsConsole` del Modulo 1 e la sua `Page` principale: ha ancora i pannelli segnaposto di navigazione, contenuti, comandi e stato, le pagine segnaposto per Editors, Layouts, Lists and Trees, DataGridView, Dashboard e Widgets, e lo UserControl `CustomerEditor` del Modulo 2. Tieni il lavoro di questo modulo in una cartella o in un namespace dedicati, così che il capstone possa riutilizzarlo.

Approccio suggerito

- ✓ Abbozza lo shell prima di aprire il designer: ToolBar in dock su Top, StatusBar in dock su Bottom, uno SplitContainer in dock su Fill — e scrivi quell'ordine dei figli in `LayoutNotes.md` come prima cosa.
- ✓ Costruisci lo shell in quell'ordine, metti il controllo di navigazione nel pannello sinistro dello SplitContainer e un TabControl in quello destro, e trasforma ogni pagina segnaposto del Modulo 1 in una tab page per una sezione di pari livello.
- ✓ Aggiungi i pulsanti della ToolBar per New, Refresh e Save e un ContextMenu sull'elenco di navigazione; ogni gestore resta sottile e chiama un metodo di comando con un nome parlante sulla pagina.
- ✓ Estrai l'intestazione di record e la striscia di stato in due UserControl `RecordHeader` e `StatusStrip` che espongono poche proprietà (per esempio `Title`, `RecordCount`) e un evento (per esempio `RefreshRequested`), mai i loro controlli figli.
- ✓ Definisci un profilo client stretto, dai alle proprietà responsive (visibilità della navigazione, chiusura del pannello dello splitter, posizione della ToolBar) un valore per il profilo stretto oppure implementa `ApplyNarrowProfile`, aggiorna la StatusBar con il profilo attivo e cattura screenshot a larghezza desktop e ridotta.

Deliverable richiesti

- ✓ Shell di navigazione costruito con SplitContainer, TabControl, ToolBar e StatusBar
- ✓ Interfaccia ripetuta spostata in UserControl con una superficie pubblica ridotta
- ✓ Layout per profilo stretto con proprietà responsive o logica basata sul profilo
- ✓ LayoutNotes.md che documenta le scelte di docking e ancoraggio

- ✓ Screenshot a larghezza desktop e ridotta

Domande di autovalutazione

- ✓ Se lo SplitContainer fosse aggiunto alla pagina prima della ToolBar e della StatusBar, che cosa vedrebbe l'utente e perché l'ordine conta?
- ✓ Quali membri del tuo UserControl potresti rinominare senza toccare nessuna delle pagine che lo usano, e quali no? Che cosa ti dice questo sulla sua superficie pubblica?
- ✓ Quali proprietà sono cambiate sul profilo stretto, e perché non hai costruito una schermata separata per gli schermi stretti?



La regola da libreria di controlli

Scegli il controllo nativo più semplice che esprime il compito dell'utente prima di scrivere codice personalizzato. I controlli nativi ti danno già supporto a design time, temi, eventi e stato sul server; il passo successivo è un extender o uno UserControl, e solo quando nessun controllo nativo è adatto si costruisce un widget JavaScript.