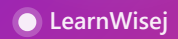


Container, Layouts, Navigation und Wiederverwendung

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Docking-Reihenfolge, Anchoring, Flow- und Table-Layouts, responsive Properties pro Client-Profil, Navigations- und Befehlsflächen sowie UserControls als Einheit der Wiederverwendung. Was das für Wisej.NET-Entwickler bedeutet – und wie Sie dieses Modul lesen.



Fokus: Produktivbetrieb

Docking-Reihenfolge, Anchoring, Flow- und Table-Layouts, responsive Properties pro Client-Profil, Navigations- und Befehlsflächen sowie UserControls als Einheit der Wiederverwendung.

1. Das Problem im Produktivbetrieb

Container sind die Architektur der Benutzeroberfläche: Sie entscheiden, wie Controls gruppiert, in der Größe angepasst, angesteuert und wiederverwendet werden. Die Operations Console aus Modul 1 hat noch Platzhalter-Panels, und ein Bildschirm, auf dem jedes Control von Hand positioniert ist, wird in dem Moment brüchig, in dem sich die Browsergröße ändert. Layout-Probleme sind meist Designprobleme, keine Pixelprobleme, und dieses Modul ersetzt das Pixelschieben durch einen Plan: angedockte Bereiche, einen SplitContainer, ein TabControl, ein echtes Befehlsmodell und UserControls, die ihr Innenleben verbergen.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
Docking-Reihenfolge	Docking heftet ein Child-Control an eine Seite seines Containers oder füllt den verbleibenden Platz, und es wird in der Reihenfolge der Child-Controls angewendet; dieselben vier Controls ergeben also je nach Reihenfolge des Hinzufügens unterschiedliche Shells.
Anchoring	Anchoring bindet die Ränder eines Controls an die Ränder seines Containers; links und rechts verankert dehnt es sich mit der Breite, auf keiner Achse verankert schwebt es und behält seine relative Position.
SplitContainer	Zwei benachbarte Bereiche, deren Größe der Benutzer ändern kann; die richtige Wahl, wann immer der Mensch und nicht der Entwickler entscheiden soll, wie viel Platz Navigationsbaum oder Detail-Panel erhalten.
Layout-Panels	FlowLayoutPanel ordnet Child-Controls nebeneinander an und bricht um, TableLayoutPanel richtet Zeilen und Spalten aus, und FlexLayoutPanel verteilt den Platz und bricht um wie ein modernes responsives Layout.
responsive Properties	Eine Property, die pro Client-Profil einen anderen Wert hat und zur Entwurfszeit gesetzt wird, sodass ein Desktop-Layout auf schmalen Bildschirmen einige wenige Properties ändern kann, ohne einen zweiten Bildschirm zu brauchen.
Befehlsmodell	Eine einheitliche Regel dafür, wo ein Befehl hingehört: MenuBar, ToolBar, RibbonBar, ContextMenu oder StatusBar, jeweils mit eigener Aufgabe, und alle rufen Anwendungsbefehle auf, statt Geschäftslogik zu enthalten.
Schnittstelle eines UserControls	Die Einheit der Wiederverwendung: ein zusammengesetztes Control, das einige aussagekräftige Properties, Events und Methoden bereitstellt statt jedes Child-Control, sodass sich Verwender nie an seine Implementierung koppeln.

3. Der rote Faden der Lektion

Den passenden Container für die Aufgabe wählen

Wisej.NET bietet absolute Positionierung, Docking, Anchoring und leistungsfähigere Layout-Panels, und die erste Entscheidung lautet, welche Oberfläche den Bildschirm überhaupt aufnimmt. Eine **Page** ist die natürliche Einstiegsoberfläche für eine Anwendung im Web-Stil wie die Operations Console: Sie füllt den Browser und hat keinen Fensterrahmen. Ein **Form** eignet sich für dialogartige Fenster, modale Abläufe und MDI-ähnliche Muster, und ein **Desktop** kann mehrere Forms aufnehmen, wenn die Anwendung bewusst eine Desktop-Metapher beibehält. Innerhalb dieser Oberfläche gruppieren **Panel** und **GroupBox** zusammengehörige Controls; der Unterschied besteht nur darin, ob die Gruppe eine sichtbare Überschrift verdient.

Drei Container leisten den Großteil der strukturellen Arbeit. Ein **SplitContainer** ist ideal, wenn der Benutzer die Größe zweier benachbarter Bereiche ändern kann, etwa bei einem Navigationsbaum neben einem Detail-Panel. Ein **TabControl** ordnet gleichrangige Seiten, sodass der Benutzer zwischen

Abschnitten wechselt, ohne neue Fenster zu öffnen. Ein `Accordion` stapelt einklappbare Abschnitte für Navigation, Einstellungen und einklappbare Arbeitsbereiche. Im Lab wird der linke Bereich der Konsole zu einem Panel eines `SplitContainer`, der rechte Bereich zu einem `TabControl`, und die Platzhalter aus Modul 1 werden zu dessen Tab-Seiten.

Docking-Reihenfolge, Anchoring und die Layout-Panels

Docking heftet ein Child-Control an eine Seite seines Containers (`DockStyle.Top`, `Bottom`, `Left`, `Right`) oder lässt es den verbleibenden Platz ausfüllen (`DockStyle.Fill`). Weil Docking in der Reihenfolge der Child-Controls angewendet wird, ist die Reihenfolge des Hinzufügens wichtig: ToolBar und StatusBar müssen ihre Ränder belegen, bevor der SplitContainer den Rest ausfüllt; andernfalls wird das Fill-Control zuerst angeordnet, und die Leisten überlappen es. Wenn eine Shell im Designer falsch aussieht, prüfen Sie die Reihenfolge der Child-Controls, bevor Sie auch nur eine Größe anfassen.

Anchoring hält die Ränder eines Controls in Beziehung zu den Rändern seines Containers. Ein links und rechts verankertes Control ändert seine Breite, wenn sich die Breite des Containers ändert; ein Control, das weder horizontal noch vertikal verankert ist, schwebt und behält seine relative Position – richtig für eine zentrierte Anmeldekarte, falsch für ein Grid. Verwenden Sie Docking für die Anwendungsbereiche (Navigation, Kopfzeile, Fußzeile oder Status, Inhalt) und Anchoring innerhalb dieser Bereiche für Editoren und Buttons.

Für Gruppen von Controls greifen Sie zu einem Layout-Panel statt zu Koordinaten. `FlowLayoutPanel` ordnet Child-Controls nebeneinander an und kann umbrechen, sodass eine Reihe von Filter-Chips auch ein schmales Browserfenster übersteht. `TableLayoutPanel` ordnet Inhalte in Zeilen und Spalten an, und genau das braucht ein Formular mit ausgerichteten Labels und Editoren. `FlexLayoutPanel` verteilt den Platz und bricht um, wie es moderne responsive Layouts tun; es ist die bessere Wahl, wenn die Oberfläche eine flexible Verteilung statt eines festen Rasters braucht.

Responsives Design ist in Wisej.NET nicht auf CSS-Media-Queries beschränkt. Responsive Properties enthalten mehrere Werte für verschiedene Client-Profile und werden zur Entwurfszeit im Eigenschaftenfenster gesetzt; Client-Profile beschreiben Geräte- oder Browsersituationen, etwa ein Smartphone bis 600 Pixel Breite, ein Tablet bis 1024 und einen Desktop darüber. Die praktische Regel lautet: Bauen Sie zuerst ein gutes Desktop-Layout und legen Sie dann die wenigen Properties fest, die sich für Tablets oder schmale Bildschirme ändern müssen: Sichtbarkeit, Größe, Docking, Spaltenanzeige, Verhalten der Seitenleiste und Platzierung der Befehle. Erstellen Sie keinen völlig eigenen Bildschirm, solange sich der Arbeitsablauf nicht wirklich ändert. Wenn eine Property allein nicht genügt, erledigt profilabhängige Logik im Code dieselbe Aufgabe, wie die Methode für das schmale Profil aus dem Lab zeigt:

```
// OperationsConsole-Shell: die Bereiche in dieser Reihenfolge hinzufügen
this.toolBar.Dock = DockStyle.Top;
this.statusBar.Dock = DockStyle.Bottom;
this.splitMain.Dock = DockStyle.Fill; // Navigation | TabControl-Detail

private void ApplyNarrowProfile(bool narrow)
{
    // schmale Bildschirme: Navigations-Panel ausblenden, Menü-Button einblenden
    this.splitMain.Panel1Collapsed = narrow;
    this.btnMobileMenu.Visible = narrow;
    this.pnlProfile.Text = narrow ? "Narrow profile" : "Desktop profile"; // ein
    StatusBar-Panel
}
}
```

Navigations- und Befehlsflächen

Navigations-Controls brauchen ein einheitliches Befehlsmodell, und bei diesem Modell geht es darum, wohin ein Befehl gehört, nicht wie er aussieht. Eine **MenuBar** eignet sich für klassische Anwendungsmenüs. Eine **ToolBar** enthält die häufig genutzten Aktionen; deshalb legt die Operations Console New, Refresh und Save dort ab. Eine **RibbonBar** gruppiert Befehle nach Aufgabenbereichen, wenn die Anwendung viele Befehle hat – für eine Konsole mit einem Dutzend Befehlen ist das zu viel des Guten. Ein **ContextMenu** ist für Befehle gedacht, die an ein bestimmtes Objekt gebunden sind: eine Zeile, einen Knoten oder die aktuelle Auswahl. In jedem Fall löst das Control ein Event aus, und ein schlanker Handler ruft einen Anwendungsbefehl auf; Navigations-Controls, die Geschäftslogik enthalten, sind einer der typischen Fehler dieses Moduls.

Die **StatusBar** ist keine Ablage für alles Übrige. Sie sollte relevanten Status zeigen, etwa die Anzahl ausgewählter Datensätze, den Zeitpunkt der letzten Aktualisierung, den Validierungszustand oder das aktive Client-Profil, und das Lab verlangt echten Status statt des Wortes „Ready“. Ein **TabControl** verdient dieselbe Disziplin. Tab-Seiten sind kein Ersatz für einen Arbeitsablauf: Verwenden Sie sie für gleichrangige Abschnitte wie Summary, Details, History, Documents und Notes, niemals als Pflichtschritte eines Assistenten. Verstecken Sie keine Pflichtfelder auf Tabs, ohne dass die Validierung klar dorthin führt, und wenn Tabs verschoben werden können, entscheiden Sie, ob die neue Reihenfolge gespeichert wird und ob sie für eine Session oder für das Benutzerprofil gilt.

UserControl ist die Einheit der Wiederverwendung

Wenn dieselbe Gruppe von Controls zweimal auftaucht, liegt es nahe, sie zu kopieren, und die Kosten kommen mit der dritten Kopie und der ersten Änderung. Ein **UserControl** ist die wichtigste Einheit der Wiederverwendung. Ein wiederverwendbares Control sollte eine kleine öffentliche Schnittstelle haben: Properties wie **CustomerId**, **ReadOnly** und **SelectedDate**, Events wie **Saved** oder **SelectionChanged** und Methoden wie **LoadData()** oder **ValidateContent()**. Legen Sie nicht alle internen Child-Controls offen, es sei denn, es handelt sich bewusst um ein fortgeschrittenes Control; sonst koppeln sich die verwendenden Bildschirme an die Implementierung, und das Umbenennen einer TextBox bricht drei Seiten. Das Lab verlagert die wiederholte Statusleiste und den Datensatz-Kopf der Konsole in UserControls; die Kontrollfrage lautet, ob ein Verwender sie nutzen könnte, ohne zu wissen, was darin steckt.

Halten Sie die Entscheidungen schriftlich fest. Die `LayoutNotes.md` aus dem Lab ist eine kurze Datei im Projekt, die festhält, welche Bereiche in welcher Reihenfolge angedockt sind, welche Controls verankert sind und warum, und welche Properties sich im schmalen Profil ändern. Sie ist das, was der nächste Entwickler liest, bevor er den Designer öffnet, und sie sollte diese Layout-Entscheidungsregeln widerspiegeln:

- Verwenden Sie Docking für Anwendungsbereiche und einen `SplitContainer`, wenn der Benutzer die Platzaufteilung steuern soll.
- Verwenden Sie `TabControl` für gleichrangige Abschnitte, `FlowLayoutPanel` für Gruppen, die umbrechen, und `TableLayoutPanel` für Formulare mit ausgerichteten Labels und Editoren.
- Verwenden Sie responsive Properties für profilspezifische Änderungen und kapseln Sie wiederholte Layouts als UserControls, statt Gruppen von Controls zu kopieren.



Typische Fehler

Absolute Positionierung, Docking und Flow ohne Plan mischen. `AutoSize` auf großen Containern oder Grids setzen und damit versehentlich riesige Layout-Berechnungen erzwingen. Tabs anlegen, die Pflichtfelder verstecken, und dann vage Validierungsfehler anzeigen. Jedes Child-Control eines UserControls öffentlich machen. Navigations-Controls für Geschäftslogik verantwortlich machen, statt Anwendungsbefehle aufzurufen.



Gewohnheit für die Control-Bibliothek

Wählen Sie das einfachste native Control, das die Aufgabe des Benutzers abbildet, bevor Sie eigenen Code schreiben. Native Controls bringen Designer-Unterstützung, Theming, Events und Serverzustand bereits mit; greifen Sie als Nächstes zu einem Extender oder einem UserControl und bauen Sie ein JavaScript-Widget erst, wenn kein natives Control passt.



Lernziele

Am Ende dieses Moduls können Sie:

- ✓ Form, Page, Desktop, Panel, GroupBox, SplitContainer, TabControl, Accordion, FlowLayoutPanel, TableLayoutPanel und UserControl korrekt einsetzen.
- ✓ Docking, Anchoring, Flow-, Table-Layouts und responsive Properties kombinieren, ohne gegen die Layout-Engine anzukämpfen.
- ✓ Navigations- und Befehlsflächen mit MenuBar, ToolBar, RibbonBar, StatusBar, ContextMenu und TabControl gestalten.
- ✓ wiederkehrende UI-Muster in UserControls mit sauberen öffentlichen Properties und Events verpacken.

Lab- und Prüfungsleitfaden

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



Praktisches Lab

Ersetzen Sie die Platzhalter aus Modul 1 durch die echte Shell der Operations Console: einen SplitContainer mit der Navigation links, einer TabControl-Detailfläche rechts, einer ToolBar für häufige Befehle und einer StatusBar, die den tatsächlichen Status anzeigt. Verlagern Sie wiederholte Oberflächenteile in UserControls, die nur aussagekräftige öffentliche Properties und Events bereitstellen, setzen Sie ein Layout für das schmale Profil mit responsiven Properties oder profilabhängiger Logik um und dokumentieren Sie Ihre Docking- und Anchoring-Entscheidungen in LayoutNotes.md.

Ausgangspunkt

Öffnen Sie die Projektmappe `OperationsConsole` aus Modul 1 und ihre Haupt- `Page` : Sie enthält noch die Platzhalter-Panels für Navigation, Inhalt, Befehle und Status, die Platzhalterseiten für Editors, Layouts, Lists and Trees, DataGridView, Dashboard und Widgets sowie das UserControl `CustomerEditor` aus Modul 2. Legen Sie die Arbeit dieses Moduls in einen eigenen Ordner oder Namespace, damit das Abschlussprojekt sie wiederverwenden kann.

Vorgeschlagenes Vorgehen

- ✓ Skizzieren Sie die Shell, bevor Sie den Designer öffnen: ToolBar mit Dock Top, StatusBar mit Dock Bottom, ein SplitContainer mit Dock Fill, und schreiben Sie diese Reihenfolge der Child-Controls zuerst in `LayoutNotes.md` .
- ✓ Bauen Sie die Shell in dieser Reihenfolge auf, legen Sie das Navigations-Control in das linke Panel des SplitContainers und ein TabControl in das rechte Panel, und machen Sie aus jeder Platzhalterseite von Modul 1 eine Tab-Seite für einen gleichrangigen Abschnitt.
- ✓ Fügen Sie der ToolBar Buttons für New, Refresh und Save sowie der Navigationsliste ein ContextMenu hinzu; jeder Handler bleibt schlank und ruft eine benannte Befehlsmethode der Seite auf.
- ✓ Lagern Sie den Datensatz-Kopf und die Statusleiste in die UserControls `RecordHeader` und `StatusStrip` aus, die einige Properties (zum Beispiel `Title` , `RecordCount`) und ein Event (zum Beispiel `RefreshRequested`) bereitstellen, niemals ihre Child-Controls.
- ✓ Definieren Sie ein schmales Client-Profil, geben Sie den responsiven Properties (Sichtbarkeit der Navigation, Einklappen des Splitter-Panels, Platzierung der ToolBar) einen Wert für das schmale Profil oder implementieren Sie `ApplyNarrowProfile` , zeigen Sie das aktive Profil in der StatusBar an und erstellen Sie Screenshots in Desktop- und schmaler Breite.

Geforderte Ergebnisse

- ✓ Navigations-Shell aus SplitContainer, TabControl, ToolBar und StatusBar
- ✓ Wiederholte Oberflächenteile in UserControls mit kleiner öffentlicher Schnittstelle verlagert
- ✓ Layout für das schmale Profil mit responsiven Properties oder profilabhängiger Logik

- ✓ LayoutNotes.md mit den Docking- und Anchoring-Entscheidungen
- ✓ Screenshots in Desktop- und schmaler Breite

Wiederholungsfragen

- ✓ Wenn der SplitContainer vor ToolBar und StatusBar zur Seite hinzugefügt würde: Was würde der Benutzer sehen, und warum ist die Reihenfolge wichtig?
- ✓ Welche Member Ihres UserControls könnten Sie umbenennen, ohne eine verwendende Seite anzufassen, und welche nicht? Was sagt Ihnen das über seine öffentliche Schnittstelle?
- ✓ Welche Properties haben sich im schmalen Profil geändert, und warum haben Sie keinen eigenen Bildschirm für schmale Breiten gebaut?



Gewohnheit für die Control-Bibliothek

Wählen Sie das einfachste native Control, das die Aufgabe des Benutzers abbildet, bevor Sie eigenen Code schreiben. Native Controls bringen Designer-Unterstützung, Theming, Events und Serverzustand bereits mit; greifen Sie als Nächstes zu einem Extender oder einem UserControl und bauen Sie ein JavaScript-Widget erst, wenn kein natives Control passt.