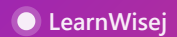


KI-Coding-Übung

Erstellen Sie das Beispielprojekt AdaptiveOps dieses Moduls mit ChatGPT oder Claude aus einem fertigen Prompt, der das Modell auf die Spezifikation des Moduls verweist – und prüfen, starten und erweitern Sie dann, was Sie erhalten, auch im Vergleich mit unserem eigenen Referenz-Build auf GitHub.



Den Code zu schreiben ist heute die leichtere Hälfte.

Funktionierenden Code für „Layout-Grundlagen: Docking, Anchoring, AutoSize, AutoScroll und Shell-Aufbau“ bekommen Sie von einem modernen Assistenten mit einem guten Prompt. Ihn zu beurteilen, zu korrigieren und zu erweitern, erfordert alles, was dieses Modul behandelt hat. Genau auf dieser Aufteilung baut die Übung auf: Sie liefern Spezifikation und Review, der Assistent die Tipparbeit.

Worum Sie bitten

Bauen Sie die Shell der Adaptive Operations Console per Refactoring in verschachtelte Container um: eine Toolbar oben, eine Navigationsleiste links, ein füllender Arbeitsbereich, ein Details-Panel rechts und ein Statusbereich, jeweils ein Panel oder UserControl mit eigenem lokalen Layout. Löschen Sie jeden Resize-Handler, der Bounds setzt, korrigieren Sie die Dock-Reihenfolge im Designer, verwenden Sie Padding für Abstände zwischen gedockten Bereichen, geben Sie Arbeitsbereich und Grid eine MinimumSize und lassen Sie den Details-Editor bei schmalen Breiten mit AutoScroll und ausgeblendeten Scrollbars scrollen.

Das Modul verlangt diese Ergebnisse, und der Prompt unten fordert genau diese vom Assistenten an:

- ✓ Shell aus verschachtelten Containern mit Toolbar-, Navigations-, Arbeitsbereichs-, Details- und Statusbereich neu aufgebaut
- ✓ Sämtlicher manueller Resize-Code, der Bounds setzte, entfernt
- ✓ Dock-Reihenfolge, Padding und MinimumSize in jedem Bereich bewusst gesetzt
- ✓ Details-Editor scrollt bei schmalen Breiten über AutoScroll mit ausgeblendeten Scrollbars
- ✓ Navigations- und Detailbereich in UserControls ausgelagert

Warum es so aufgebaut ist

Die Adaptive Operations Console hat jetzt ein Theme und bereichsbezogenes CSS, doch ihre Shell verschiebt Controls noch von Hand: Ein Resize -Handler berechnet bei jeder Größenänderung des Browsers die Bounds für Toolbar, Navigationsleiste, Ticket-Grid und Details-Editor neu. Das funktioniert bei der Breite, die der Entwickler getestet hat, bricht bei jeder anderen und übersteht die Tablet- und Smartphone-Profile nicht, die Modul 6 einführt. Adaptive Design in Wisej.NET beruht auf Containern und Layout-Regeln, nicht auf Koordinaten: Bereiche, die Ränder beanspruchen, ein Arbeitsbereich, der den Rest füllt, und Editoren, die scrollen, statt sich zu überlappen.

Diese Begriffe muss der Assistent richtig verwenden. Geht die Antwort nachlässig mit ihnen um, hat er das Modul nicht verstanden:

Begriff	Was er hier bedeutet
Standard-Layout-Engine	Jedes Wisej.NET-Control hat eine Layout-Engine; die Standard-Engine unterstützt absolute Positionierung, Docking und Anchoring, sodass ein Bildschirm präzise Platzierung im Designer mit Bereichen kombinieren kann, die ihre Größe selbst anpassen.
Docking-Priorität	Gedockte Children belegen Platz in der Reihenfolge, in der sie in der Control-Collection des Parents stehen; deshalb ergeben dieselben fünf Panels je nach Child-Reihenfolge eine korrekte oder eine überlappende Shell.
DisplayRectangle und Padding	Gedockte Controls werden innerhalb des DisplayRectangle des Parents angeordnet, das Padding verkleinert; die Standard-Engine ignoriert Margin bei gedockten Children, Abstände entstehen also durch Padding oder Platzhalter-Controls.
Anchoring	Ein Anchor hält den Abstand eines Controls zu ausgewählten Seiten des Parents konstant, was zu formularartigen Bereichen passt: Labels oben links, Textfelder links-rechts, Befehls-Buttons unten rechts.
AutoSize und zirkuläres Layout	Ein Container, der seine Größe aus Children ableitet, die wiederum an ihm gedockt oder verankert sind, hat keine stabile Antwort; diese Schleife ist die zentrale AutoSize-Warnung des Moduls.
AutoScroll mit ausgeblendeten Leisten	AutoScroll lässt einen Container scrollen, wenn der Inhalt nicht hineinpasst, und ScrollBars.Hidden erhält das Touch-Scrollen auf Smartphones, ohne Scrollbars zu zeichnen.
UserControl-Bereich	Ein Shell-Bereich mit eigenem lokalen Layout und kleiner öffentlicher Oberfläche, sodass die Shell den ganzen Bereich pro Profil anzeigt, ausblendet, dockt oder verschiebt, statt seine internen Koordinaten zu kennen.

Selbst bauen – wählen Sie Ihren Assistenten

Beide Buttons öffnen einen neuen Tab, in dem **derselbe fertige Prompt** bereits eingetragen ist. Der Prompt verweist das Modell auf die Spezifikation dieses Moduls – eine einzelne Markdown-Datei mit der Lektion, der Lab-Aufgabe und den geforderten Ergebnissen –, sodass es mit dem echten Material arbeitet, statt zu raten, worum es in Modul 4 geht. Sie können [module-04.md](#) selbst öffnen, um genau zu sehen, was es lesen wird.



ChatGPT verwenden

Öffnet [chatgpt.com](#) mit dem Prompt dieses Moduls. Funktioniert mit GPT-5 oder



Claude verwenden

Öffnet [claude.ai](#) mit demselben Prompt. Wenn Sie Claude Code nutzen, können Sie

Codex; jeder Tarif mit Websuche ruft die Spezifikation ab.

ihn stattdessen in einem leeren Ordner in ein Terminal einfügen.



Unverändert abgeschickt, erhalten Sie ein Projekt, kein Code-Fragment

Der Prompt verlangt am Ende die gesamte Projektmappe als einzelne `.zip`-Datei mit Anleitung zum Kompilieren und Starten. Laden Sie sie herunter, öffnen Sie die Projektmappe in Visual Studio, drücken Sie F5, und AdaptiveOps startet auf `localhost`. Sie benötigen das .NET SDK und das NuGet-Paket `Wisej-4`; alles andere soll der Assistent laut Prompt selbst schreiben.

Ich arbeite gerade Modul 4 – „Layout-Grundlagen: Docking, Anchoring, AutoSize, AutoScroll und Shell-Aufbau“ – des LearnWisej-Kurses „Theming und responsive Layouts“ durch und möchte das Beispielprojekt des Moduls gemeinsam mit dir erstellen. Bitte antworte auf Deutsch.

Lies zuerst die Modulspezifikation. Sie ist eine einzelne Markdown-Datei (auf Englisch) mit der vollständigen Lektion, der Lab-Aufgabe und den geforderten Ergebnissen:

<https://learnwisej.com/assets/courses/theming-and-responsive-layouts/specs/module-04.md>

Erstelle dann ein vollständiges Wisej.NET-4-Projekt, das ich in Visual Studio öffnen kann:

1. Projekt „AdaptiveOps“, C#, net8.0 oder neuer, NuGet-Paket Wisej-4 4.1.x, Kestrel-Host, der `app.UseWisej()` aufruft.
2. Setze um, was das Lab der Spezifikation verlangt: Bauen Sie die Shell der Adaptive Operations Console per Refactoring in verschachtelte Container um: eine Toolbar oben, eine Navigationsleiste links, ein füllender Arbeitsbereich, ein Details-Panel rechts und ein Statusbereich, jeweils ein Panel oder UserControl mit eigenem lokalen Layout. Löschen Sie jeden Resize-Handler, der Bounds setzt, korrigieren Sie die Dock-Reihenfolge im Designer, verwenden Sie Padding für Abstände zwischen gedockten Bereichen, geben Sie Arbeitsbereich und Grid eine `MinimumSize` und lassen Sie den Details-Editor bei schmalen Breiten mit `AutoScroll` und ausgeblendeten Scrollbars scrollen.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Shell aus verschachtelten Containern mit Toolbar-, Navigations-, Arbeitsbereichs-, Details- und Statusbereich neu aufgebaut
 - Sämtlicher manueller Resize-Code, der Bounds setzte, entfernt
 - Dock-Reihenfolge, Padding und `MinimumSize` in jedem Bereich bewusst gesetzt
 - Details-Editor scrollt bei schmalen Breiten über `AutoScroll` mit ausgeblendeten Scrollbars
 - Navigations- und Detailbereich in UserControls ausgelagert
4. Halte dich an die Spezifikation, wo immer sie eine Wisej.NET-Klasse, -Eigenschaft oder ein -Event nennt – verwende genau diese. Wo sie eine Wahl offenlässt, sag mir, welche API du gewählt hast und warum.
5. Der Server bleibt maßgeblich: Zustand und Validierung leben in .NET, der Browser stellt dar und meldet zurück. Verlagere nie eine Entscheidung nach JavaScript, nur weil es dort einfacher ist.
6. Mach das Ergebnis vorführbar: Ich muss es starten und das Verhalten sehen können, um das es in diesem Modul geht – auch, was passiert, wenn etwas schiefgeht.

Bevor du Code schreibst, fasse in eigenen Worten zusammen, was du bauen wirst und wo jedes Teil liegen wird, damit ich den Entwurf korrigieren kann, solange das noch wenig kostet.

Wenn der Code fertig ist, packe die gesamte Projektmappe in eine einzelne .zip-Datei zum Herunterladen und nenne mir die genauen Schritte, um sie zu kompilieren und auszuführen.

Beantworte zum Schluss diese Fragen aus der Spezifikation und nutze dabei deine eigene Implementierung als Beleg:

- Wäre das Arbeitsbereichs-Panel in der Child-Reihenfolge vor dem Status-Panel gedockt: Was sähe der Benutzer am unteren Rand des Grids, und welcher Designer-Befehl behebt das, ohne eine einzige Größe zu ändern?
- Sie haben an der Navigationsleiste einen Margin von 12 Pixeln gesetzt, und der Abstand ist nicht erschienen. Warum nicht, und welche zwei Mechanismen erzeugen tatsächlich einen Abstand zwischen gedockten Bereichen?
- Der Details-Editor ist ein UserControl mit `AutoScroll`. Welche seiner Member muss die Shell

kennen, um ihn auf dem Tablet unter das Grid zu verschieben, und welche darf sie nie benötigen?

Frag mich nach allem, was unklar ist, bevor du anfängst.

Prüfen Sie dann, was zurückkommt

Code, der kompiliert, ist noch lange kein richtiger Code. Die spannenden Fragen sind Designfragen: Ist das wirklich das Wisej.NET-Control, das die Aufgabe verlangt? Folgt der Code dem Lebenszyklus, den das Framework vorgibt? Liegt der maßgebliche Zustand auf dem Server – oder wurde er im Browser abgelegt, weil das bequemer war? Lesen Sie das Ergebnis als Reviewer, nicht als Kunde.

Die Checkliste liefert Ihnen dieses Modul. Beantworten Sie jeden Punkt anhand des Codes, den Sie gerade erhalten haben:

- ✓ Wäre das Arbeitsbereichs-Panel in der Child-Reihenfolge vor dem Status-Panel gedockt: Was sähe der Benutzer am unteren Rand des Grids, und welcher Designer-Befehl behebt das, ohne eine einzige Größe zu ändern?
- ✓ Sie haben an der Navigationsleiste einen Margin von 12 Pixeln gesetzt, und der Abstand ist nicht erschienen. Warum nicht, und welche zwei Mechanismen erzeugen tatsächlich einen Abstand zwischen gedockten Bereichen?
- ✓ Der Details-Editor ist ein UserControl mit AutoScroll. Welche seiner Member muss die Shell kennen, um ihn auf dem Tablet unter das Grid zu verschieben, und welche darf sie nie benötigen?

Alles, was Sie korrigieren mussten, sollten Sie festhalten. Eine kurze Notiz zu den Konventionen und Entscheidungen, auf die Sie sich festgelegt haben – eingefügt in die erste Nachricht Ihrer nächsten Sitzung –, ist mehr wert als jeder noch so clevere Prompt.



Dieses Beispiel haben wir schon gebaut – vergleichen Sie Ihres mit unserem

Die Referenzimplementierung zu Modul 4 liegt im öffentlichen Beispiel-Repository. Bauen Sie zuerst Ihre eigene Version und öffnen Sie dann unsere: Die Unterschiede sind der interessante Teil.



[Beispiel zu Modul 4 auf GitHub öffnen](#)

Oder klonen Sie alle Beispiele mit `git clone https://github.com/iceteagroup/LearnWisej-Samples` und öffnen Sie [Theming and Responsive Layouts Course/Module 4](#).



Unser Beispiel ist ein Ausgangspunkt, kein fertiges Produkt

Es ist genauso entstanden wie Ihres – per Prompt und anschließend korrigiert – und endet bewusst dort, wo das Modul endet. Rechnen Sie mit Platzhaltern, wo Produktivcode hingehört, mit knapper Fehlerbehandlung und ohne Tests. Verstehen Sie das als Einladung: Nutzen Sie, was Sie in diesem Modul gelernt haben, und dazu den Assistenten, um es weiterzuentwickeln. Code zu verbessern, den Sie nicht selbst geschrieben haben, ist die Fähigkeit, die dieser Kurs eigentlich vermittelt.

Drei Prompts für den nächsten Schritt

Richten Sie Ihren Assistenten auf das Projekt, das Sie gerade gebaut haben – oder auf unseres –, und probieren Sie diese Prompts aus. Jeder ist eine Übung zum Prüfen und Erweitern, kein Neuschreiben.

- ✓ **Tauschen Sie ein Teil aus, ohne die Schnittstellen zu verschieben.** *"Wähle den Teil dieses AdaptiveOps-Projekts, der am offensichtlichsten nur ein Platzhalter für etwas Echtes ist – eine Fake-Datenquelle, ein In-Memory-Speicher, ein vereinfachter Service – und ersetze ihn durch eine praxisnahe Implementierung. Der Vertrag auf beiden Seiten darf sich nicht ändern: dieselben Typen, dieselben Aufrufe, dasselbe Verhalten im Fehlerfall. Sag mir, was du durch die echte Implementierung behandeln musstest, was der Platzhalter nicht behandelt hat."*
- ✓ **Machen Sie es fit für einen schlechten Tag.** *"Untersuche dieses AdaptiveOps-Projekt auf Fehlerszenarien, die es noch nicht behandelt – fehlerhafte oder fehlende Eingaben, eine Abhängigkeit, die langsam oder nicht erreichbar ist, Arbeit, die eintrifft, nachdem der Benutzer die Seite verlassen hat, und zwei Aktionen, die sich gegenseitig in die Quere kommen. Sag mir für jedes zuerst, ob es in Wisej.NET 4 ein echtes Risiko ist und warum, und schreib erst dann die Korrektur. Wo du entscheidest, dass nichts behandelt werden muss, sag das und begründe es, statt defensiven Code hinzuzufügen."*
- ✓ **Ergänzen Sie den zweiten Fall.** *"Erweitere dieses AdaptiveOps-Projekt um einen zweiten Fall, der demselben Muster folgt wie der erste – eine weitere Entität, Maske oder ein Control derselben Form, auf dieselbe Weise verdrahtet. Lagere wirklich Gemeinsames in eine Basisklasse oder einen Service aus, aber nur dort, wo es sich lohnt; zwei ähnliche Dinge brauchen nicht immer eine Abstraktion. Zeig mir den Diff und begründe jede Auslagerung."*



Nehmen Sie es mit ins Lab

Das praktische Lab am Ende dieses Moduls verlangt dieselben Ergebnisse, die Sie gerade vom Assistenten haben erstellen lassen. Nutzen Sie, was Sie hier haben, als Gerüst für Ihre Antwort und als Beleg dafür – nicht als die Antwort selbst. Bewertet wird im Lab Ihre Argumentation, und dieser Teil muss von Ihnen stammen.