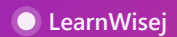


KI-Coding-Übung

Erstellen Sie das Beispielprojekt OperationsConsole 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.



Instruieren, lesen, dann auf die Probe stellen.

Diese Übung hat drei Teile, und nur der erste dreht sich um den Prompt. Sie instruieren einen Assistenten mit der Modulspezifikation und erhalten eine OperationsConsole-Projektmappe zurück; Sie lesen, was er geschrieben hat, und entscheiden, ob es wirklich tut, was das Modul verlangt; dann treiben Sie es weiter, als das Modul verlangt hat. Im zweiten und dritten Teil steckt das eigentliche Lernen.

Worum Sie bitten

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.

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

- ✓ 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

Warum es so aufgebaut ist

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.

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
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 UserControl	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.

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 3 geht. Sie können [module-03.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 OperationsConsole 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 3 – „Container, Layouts, Navigation und Wiederverwendung“ – des LearnWisej-Kurses „Die Control-Bibliothek meistern“ 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/mastering-the-control-library/specs/module-03.md>

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

1. Projekt „OperationsConsole“, 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: 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.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - 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
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:

- 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?

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:

- ✓ 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?

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 3 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 3 auf GitHub öffnen](#)

Oder klonen Sie alle Beispiele mit `git clone https://github.com/iceteagroup/LearnWisej-Samples` und öffnen Sie `Mastering the Control Library Course/Module 3`.



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.

- ✓ **Beweisen Sie, dass es wirklich funktioniert.** *"Geh dieses Projekt Ergebnis für Ergebnis anhand der Modulspezifikation unter <https://learnwisej.com/assets/courses/mastering-the-control-library/specs/module-03.md> durch und zeig mir den Beleg, dass jedes wirklich erledigt ist – den Codepfad, worauf ich klicken muss und was ich sehen sollte. Wo ein Ergebnis nur teilweise erfüllt ist, sag das klar, statt es schönzureden. Ergänze dann, was fehlt."*
- ✓ **Prüfen Sie es wie einen Pull-Request.** *"Führ mich durch diesen OperationsConsole-Code, als würdest du den Pull-Request eines Kollegen prüfen und hättest ihn nicht selbst geschrieben. Wo lebt der Zustand, was passiert bei einer einzelnen Benutzeraktion von Anfang bis Ende, was würdest du zur Überarbeitung zurückgeben, und welche Teile würdest du nicht guten Gewissens einem Kunden vorlegen? Sei konkret und zitiere die Zeilen, die du meinst – fasse mir nicht den Entwurf zusammen."*
- ✓ **Tauschen Sie ein Teil aus, ohne die Schnittstellen zu verschieben.** *"Wähle den Teil dieses OperationsConsole-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."*



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.