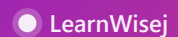


KI-Coding-Übung

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.



Den Code zu schreiben ist heute die leichtere Hälfte.

Funktionierenden Code für „Listen, Bäume, Repeater und hierarchische Daten“ 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

Erstellen Sie den Dokument-Explorer der Operations Console: eine TreeView mit Kategorien oder Ordnern, bei der an jedem Knoten eine stabile ID hängt, eine ListView mit den Dokumenten des ausgewählten Knotens, verzögertes Laden oder Virtual Mode auf mindestens einem Listen- oder Baumpfad, Icons oder Image-Keys für die verschiedenen Elementtypen und ein Detail-UserControl, das bei jedem Auswahlwechsel über eine Service- oder ViewModel-Schicht geladen wird.

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

- ✓ TreeView mit Kategorien und einer stabilen ID an jedem Knoten
- ✓ ListView mit den Dokumenten des ausgewählten Baumknotens
- ✓ Verzögertes Laden oder Virtual Mode auf mindestens einem Listen- oder Baumpfad
- ✓ Detail-UserControl, das bei der Auswahl über eine Service- oder ViewModel-Schicht geladen wird
- ✓ Icons oder Image-Keys für die verschiedenen Elementtypen

Warum es so aufgebaut ist

Listen- und Baum-Controls werden oft als einfache Auswahl-Widgets behandelt, doch in einer Geschäftsanwendung sind sie die Landkarte des Benutzers: Ordner, Kunden, Aufträge, Verfahren, Rollen, Aufgaben, Dokumente, Produktkataloge und die Module der Anwendung selbst. Der Dokument-Explorer der Operations Console, den Sie in diesem Modul bauen, muss schnell bleiben, wenn der Kategoriebaum tief und die Dokumentliste lang ist, und jeder Klick muss auf einen echten Geschäftsdatensatz verweisen statt auf ein Stück Anzeigetext.

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
Datenform	Die erste Entwurfsfrage: Sind die Daten flach, gruppiert, hierarchisch, als Karten wiederholt oder eine Menge von Properties? Die Antwort bestimmt das Control.
ListView-Ansichten	Ein Control, das dieselbe Sammlung als LargeIcon, SmallIcon, Details oder Tile anzeigt; die natürliche Wahl für dateiähnliche und katalogartige Bildschirme.
Virtual Mode	Ein ListView-Modus, in dem die Anwendung VirtualListSize setzt und RetrieveVirtualItem beantwortet, sodass Einträge nur für die Zeilen existieren, nach denen das Control fragt.
verzögertes Laden	Beim Start nur die oberen Ebenen eines Baums laden und Kinder erst beim Aufklappen eines Knotens holen, damit der Start günstig und die Oberfläche reaktionsschnell bleibt.
virtuelles Scrollen und Vorabladen	Im Browser nur die sichtbaren Zeilen einer großen Liste oder eines großen Baums rendern, wobei PrefetchItems einige zusätzliche Zeilen vorab rendert, damit das Scrollen flüssig bleibt.
stabile Identität	Eine eindeutige fachliche ID in Tag oder Name des Knotens oder Eintrags oder in einer ViewModel-Zuordnung, niemals aus dem angezeigten Text abgeleitet.
Auswahl als Event	Ein Auswahlwechsel ermittelt eine ID und fordert die Daten bei einem Service an; im Code, der Knoten oder Einträge aufbaut, läuft niemals eine Abfrage.

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 Codex; jeder Tarif mit Websuche ruft die Spezifikation ab.



Claude verwenden

Öffnet claude.ai mit demselben Prompt. Wenn Sie Claude Code nutzen, können Sie 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 4 – „Listen, Bäume, Repeater und hierarchische Daten“ – 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-04.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: Erstellen Sie den Dokument-Explorer der Operations Console: eine TreeView mit Kategorien oder Ordnern, bei der an jedem Knoten eine stabile ID hängt, eine ListView mit den Dokumenten des ausgewählten Knotens, verzögertes Laden oder Virtual Mode auf mindestens einem Listen- oder Baumpfad, Icons oder Image-Keys für die verschiedenen Elementtypen und ein Detail-UserControl, das bei jedem Auswahlwechsel über eine Service- oder ViewModel-Schicht geladen wird.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - TreeView mit Kategorien und einer stabilen ID an jedem Knoten
 - ListView mit den Dokumenten des ausgewählten Baumknotens
 - Verzögertes Laden oder Virtual Mode auf mindestens einem Listen- oder Baumpfad
 - Detail-UserControl, das bei der Auswahl über eine Service- oder ViewModel-Schicht geladen wird
 - Icons oder Image-Keys für die verschiedenen Elementtypen
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:

- Zwei Kategorien unter verschiedenen Kunden heißen beide „Contracts“. Was genau verwendet Ihr Auswahl-Handler, um sie zu unterscheiden, und wo liegt dieser Wert?
- Wenn die Anzahl der Dokumente in einer Kategorie von 300 auf 300.000 wüchse: Welcher Teil Ihres Explorers würde zuerst brechen, und welche Property oder welches Event würden Sie zur Behebung ändern?
- Welche Zeilen Ihres Auswahl-Handlers ließen sich ohne Browser in einen Unit-Test verschieben, und welche Zeilen gehören stattdessen in das Detail-UserControl?

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:

- ✓ Zwei Kategorien unter verschiedenen Kunden heißen beide „Contracts“. Was genau verwendet Ihr Auswahl-Handler, um sie zu unterscheiden, und wo liegt dieser Wert?
- ✓ Wenn die Anzahl der Dokumente in einer Kategorie von 300 auf 300.000 wüchse: Welcher Teil Ihres Explorers würde zuerst brechen, und welche Property oder welches Event würden Sie zur Behebung ändern?
- ✓ Welche Zeilen Ihres Auswahl-Handlers ließen sich ohne Browser in einen Unit-Test verschieben, und welche Zeilen gehören stattdessen in das Detail-UserControl?

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 `Mastering the Control Library 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.

- ✓ **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."*
- ✓ **Machen Sie es fit für einen schlechten Tag.** *"Untersuche dieses OperationsConsole-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."*



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.