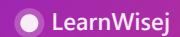


KI-Coding-Übung

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.



Dasselbe Modul – so gebaut, wie Sie in der Praxis wirklich bauen.

Alle anderen Abschnitte dieses Moduls erklären, wie „DataGridView meistern“ funktioniert. Dieser fordert Sie auf, es herzustellen: indem Sie einen Assistenten mit der Spezifikation des Moduls steuern und dann prüfen, ob das Ergebnis läuft und tut, was die Lab-Aufgabe beschreibt. Es ist der Arbeitsablauf, den Sie in echten Projekten nutzen werden – geübt an etwas, dessen richtige Lösung Sie bereits kennen.

Worum Sie bitten

Erstellen Sie das Orders-Grid der Operations Console: eine DataGridView, gebunden an eine BindingSource, mit expliziten Spalten und kontrolliertem AutoGenerateColumns, eine Statuszelle, die mit AllowHtml und CellFormatting dargestellt wird, eine eigene Editor- oder Befehlsspalte, eine Variante mit Virtual Mode oder Cache für große Datenmengen sowie eine zusammengesetzte Filter- und Statusleiste, die innerhalb oder um das Grid angedockt ist.

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

- ✓ Orders-DataGridView mit expliziten Spalten, gebunden an eine BindingSource
- ✓ Statuszelle, dargestellt mit AllowHtml und CellFormatting
- ✓ Eine eigene Editor- oder Befehlsspalte
- ✓ Variante des Grids für große Datenmengen mit Virtual Mode oder Cache
- ✓ Zusammengesetzte Filter- und Statusleiste, innerhalb oder um das Grid angedockt

Warum es so aufgebaut ist

DataGridView ist das Control, das am ehesten darüber entscheidet, ob sich eine Geschäftsanwendung ausgereift anfühlt. Aufträge, Rechnungen, Fristen, Logs, Benutzer und Dokumente landen alle in einem Grid, und die Benutzer beurteilen die Anwendung danach, wie dieses Grid lädt, scrollt, Bearbeitungen annimmt und Status meldet. Wisej.NET bietet alles vom manuellen Befüllen bis zu Virtual Mode und Zell-Painting; Meisterschaft heißt, die richtige Technik für die Datenmenge und die Aufgabe des Benutzers zu wählen.

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
BindingSource	Ein serverseitiger Vermittler zwischen ViewModel-Zeilen und dem Grid, der ihm eine stabile Datenquelle, eine aktuelle Position und Sortierung gibt, ohne den Bildschirm an die Datenbank zu koppeln.
AutoGenerateColumns	Legt für jede öffentliche Property des gebundenen Typs eine Spalte an; praktisch für einen Prototyp, gefährlich im Produktivbetrieb, weil eine Modelländerung den Bildschirm unbemerkt umstellt.
explizite Spalte	Eine Spalte, die Sie mit Feldbindung, Überschrift, Breite, Ausrichtung, Format, Schreibschutz, Sortiermodus und Editortyp definieren, damit die Oberfläche stabil bleibt, wenn sich das Modell ändert.
Virtual Mode	Mit eingeschaltetem VirtualMode kennt das Grid nur seine Zeilenanzahl und fragt jeden sichtbaren Wert über CellValueNeeded ab, sodass für Zeilen, die niemand sieht, nichts erzeugt wird.
serverseitiger Cache	Ein Puffer in Seitengröße, der CellValueNeeded speist, damit das Scrollen nicht pro Zelle die Datenbank trifft; das DataRead -Event ist der dokumentierte Einstiegspunkt, um ihn aufzubauen und zu pflegen.
eigener Zell-Editor	Ein Control, das der Property Editor einer Spalte zugewiesen ist; es muss seinen Wert in die Zelle zurückschreiben, über CellBeginEdit und CellEndEdit, wenn es kein passendes Change-Event hat.
Zell-HTML vs. Zell-Control	In CellFormatting mit AllowHtml erzeugtes HTML ist reines Anzeige-Markup und nahezu kostenlos; ein eingebettetes Control ist ein echtes Server-Control und Browser-Widget mit Kosten pro Zelle.

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 5 geht. Sie können [module-05.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 5 – „DataGridView meistern“ – des LearnWisej-Kurses „Die Control-Bibliothek meistern“ durch und möchte das Beispielpjekt 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-05.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 das Orders-Grid der Operations Console: eine DataGridView, gebunden an eine BindingSource, mit expliziten Spalten und kontrolliertem `AutoGenerateColumns`, eine Statuszelle, die mit `AllowHtml` und `CellFormatting` dargestellt wird, eine eigene Editor- oder Befehlsspalte, eine Variante mit Virtual Mode oder Cache für große Datenmengen sowie eine zusammengesetzte Filter- und Statusleiste, die innerhalb oder um das Grid angedockt ist.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Orders-DataGridView mit expliziten Spalten, gebunden an eine BindingSource
 - Statuszelle, dargestellt mit `AllowHtml` und `CellFormatting`
 - Eine eigene Editor- oder Befehlsspalte
 - Variante des Grids für große Datenmengen mit Virtual Mode oder Cache
 - Zusammengesetzte Filter- und Statusleiste, innerhalb oder um das Grid angedockt
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 OrderRow im nächsten Sprint zwei neue Properties erhält: Was ändert sich auf dem Bildschirm mit generierten Spalten, und was mit Ihren expliziten Spalten?
- Ihr CellValueNeeded -Handler funktioniert mit 200 Aufträgen. Was passiert mit der Datenbank, wenn ein Benutzer den Scrollbalken durch 200.000 zieht, und welcher Teil Ihres Entwurfs fängt das ab?
- Warum gehört das Status-Badge in CellFormatting und nicht in das OrderRow -Modell, und was würde schiefgehen, wenn ein Kundenname ohne Kodierung in dieses HTML gelangte?

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 OrderRow im nächsten Sprint zwei neue Properties erhält: Was ändert sich auf dem Bildschirm mit generierten Spalten, und was mit Ihren expliziten Spalten?
- ✓ Ihr CellValueNeeded -Handler funktioniert mit 200 Aufträgen. Was passiert mit der Datenbank, wenn ein Benutzer den Scrollbalken durch 200.000 zieht, und welcher Teil Ihres Entwurfs fängt das ab?
- ✓ Warum gehört das Status-Badge in CellFormatting und nicht in das OrderRow -Modell, und was würde schiefgehen, wenn ein Kundenname ohne Kodierung in dieses HTML gelangte?

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 5 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 5 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 5`.



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 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."*
- ✓ **Ergänzen Sie den zweiten Fall.** *"Erweitere dieses OperationsConsole-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.