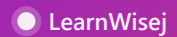


KI-Coding-Übung

Erstellen Sie das Beispielprojekt ValidationClinic 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.



Dieses Beispiel tippen Sie nicht ab. Sie steuern es.

Kaum jemand schreibt Wisej.NET-Code heute noch in einer leeren Datei: Ein Assistent liefert den ersten Entwurf, und der Entwickler entscheidet, ob er stimmt. Das funktioniert aber nur, wenn Sie den Stoff beherrschen – und genau den hat Ihnen der Rest dieses Moduls vermittelt. Hier geben Sie dieses Wissen an ChatGPT oder Claude weiter, erhalten das Beispielprojekt des Moduls als lauffähige Visual-Studio-Projektmappe zurück und prüfen es so, wie Sie den Pull-Request eines Kollegen prüfen würden.

Worum Sie bitten

Sorgen Sie dafür, dass das Kontakt-Grid von ValidationClinic fehlerhafte Daten zurückweist. Behandeln Sie `gridContacts.CellValidating`: Lesen Sie `e.FormattedValue` statt des Zellwerts, setzen Sie den `ErrorText` der Zelle zurück und weisen Sie dann einen leeren `EmailColumn`-Wert oder einen ohne `@` sowie einen `AgeColumn`-Wert außerhalb von 0 bis 120 mit `e.Cancel = true` und einem `ErrorText` der Zelle ab. Ergänzen Sie eine Methode `ValidateRow`, die den `ErrorText` der Zeile und den Zellfehler von `ClosedDateColumn` setzt, wenn Status auf `Closed` steht, aber kein Abschlussdatum vorhanden ist, entfernen Sie veraltete Fehler in `CellValidated` und behandeln Sie `DataError` mit `ThrowException = false`, einer verständlichen Zellmeldung und einem `Debug`-Log der Exception. Überspringen Sie die Platzhalterzeile für neue Einträge, bis sie Benutzerdaten enthält, und blockieren Sie das Speichern, solange noch eine Zeile einen Fehler hat.

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

- ✓ `CellValidating`-Handler, der `e.FormattedValue` liest und die Spalten für E-Mail und Alter validiert
- ✓ Methode `ValidateRow`, die für geschlossene Kontakte den `ErrorText` der Zeile und den Zellfehler für das Abschlussdatum setzt
- ✓ `CellValidated`-Handler, der veraltete Zellfehler entfernt
- ✓ `DataError`-Handler mit `ThrowException = false`, einer verständlichen Meldung und protokollierter Exception
- ✓ Schutz beim Speichern, der die Platzhalterzeile überspringt und das Speichern verweigert, solange eine Zeile einen Fehler hat

Warum es so aufgebaut ist

Die Module 1 bis 5 haben das Aufnahmeformular von ValidationClinic geschützt, einen Kontakt nach dem anderen. Das Kontakt-Grid ist anders: Ein Benutzer kann mit Tab durch zwanzig Zeilen springen, eine E-Mail-Adresse ohne @ eingeben, „abc“ als Alter eintragen und einen Kontakt ohne Abschlussdatum auf Closed setzen, alles ohne ein Formular zu öffnen. Eine Grid-Zelle ist keine TextBox, daher erreichen die Validating -Handler und der ErrorProvider, die Sie bereits geschrieben haben, sie nicht. Dieses Modul gibt dem Grid eigene Feldregeln, Zeilenregeln und Regeln für die Datenquelle und stellt sicher, dass keine davon einen normalen Tippfehler in einen Absturz oder eine blockierte Maske verwandelt.

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
CellValidating	Das Grid-Event, das ausgelöst wird, wenn die aktuelle Zelle gleich den Fokus verliert; es ist die Grid-Variante des Validating -Events eines Controls und der Ort, um einen einzelnen Wert abzulehnen.
e.FormattedValue	Der Wert genau so, wie der Benutzer ihn im Editor eingegeben hat, bevor er in die Zelle oder die Datenquelle übernommen wird; lesen Sie stattdessen Value der Zelle, erhalten Sie den alten Wert.
e.Cancel	Steht es auf true, wird das Abschließen der Bearbeitung verweigert, sodass der Cursor in der fehlerhaften Zelle bleibt, bis der Benutzer sie korrigiert oder die Bearbeitung abbricht.
CellValidated	Wird ausgelöst, nachdem eine Zelle die Validierung bestanden hat; ein sicherer Ort, um einen nicht mehr zutreffenden Fehler zu entfernen oder eine Zusammenfassung zu aktualisieren.
ErrorText der Zelle	Eine Meldung an einer einzelnen Zelle, angezeigt mit einem Fehler-Icon in dieser Zelle; verwenden Sie sie für ein Problem mit einem einzelnen Wert.
ErrorText der Zeile	Eine Meldung an der ganzen Zeile, angezeigt im Zeilenkopf; verwenden Sie sie für eine Regel, die von mehreren Zellen derselben Zeile abhängt.
DataError	Wird ausgelöst, wenn das Grid einen Wert nicht konvertieren, formatieren oder übernehmen kann; mit e.ThrowException = false ersetzen Sie die Exception durch eine verständliche Meldung.

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 6 geht. Sie können [module-06.md](#) selbst öffnen, um genau zu sehen, was es lesen wird.



ChatGPT verwenden



Claude verwenden

Öffnet chatgpt.com mit dem Prompt dieses Moduls. Funktioniert mit GPT-5 oder Codex; jeder Tarif mit Websuche ruft die Spezifikation ab.

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 ValidationClinic 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 6 – „Zellvalidierung und Zeilenfehler im DataGridView“ – des LearnWisej-Kurses „Validierung in Wisej.NET“ 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/validation-in-wisej-net/specs/module-06.md>

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

1. Projekt „ValidationClinic“, 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: Sorgen Sie dafür, dass das Kontakt-Grid von `ValidationClinic` fehlerhafte Daten zurückweist. Behandeln Sie `gridContacts.CellValidating`: Lesen Sie `e.FormattedValue` statt des Zellwerts, setzen Sie den `ErrorText` der Zelle zurück und weisen Sie dann einen leeren `EmailColumn`-Wert oder einen ohne `@` sowie einen `AgeColumn`-Wert außerhalb von 0 bis 120 mit `e.Cancel = true` und einem `ErrorText` der Zelle ab. Ergänzen Sie eine Methode `ValidateRow`, die den `ErrorText` der Zeile und den Zellfehler von `ClosedDateColumn` setzt, wenn Status auf `Closed` steht, aber kein Abschlussdatum vorhanden ist, entfernen Sie veraltete Fehler in `CellValidated` und behandeln Sie `DataError` mit `ThrowException = false`, einer verständlichen Zellmeldung und einem Debug-Log der Exception. Überspringen Sie die Platzhalterzeile für neue Einträge, bis sie Benutzerdaten enthält, und blockieren Sie das Speichern, solange noch eine Zeile einen Fehler hat.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - `CellValidating`-Handler, der `e.FormattedValue` liest und die Spalten für E-Mail und Alter validiert
 - Methode `ValidateRow`, die für geschlossene Kontakte den `ErrorText` der Zeile und den Zellfehler für das Abschlussdatum setzt
 - `CellValidated`-Handler, der veraltete Zellfehler entfernt
 - `DataError`-Handler mit `ThrowException = false`, einer verständlichen Meldung und protokollierter Exception
 - Schutz beim Speichern, der die Platzhalterzeile überspringt und das Speichern verweigert, solange eine Zeile einen Fehler hat
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:

- Die E-Mail-Regel eines Kommilitonen liest `grid.Rows[e.RowIndex].Cells[e.ColumnIndex].Value`

und „funktioniert beim Testen“. Beschreiben Sie genau die Bearbeitung, die durchrutscht, und warum.

- Ihre Altersregel prüft den Bereich 0 bis 120. Was sollte sie tun, wenn der Text überhaupt keine Zahl ist, und warum wird ein `ValidationError`-Handler trotzdem gebraucht, selbst wenn die Regel „abc“ selbst ablehnt?
- Die `Required`-Regel könnte als Zellfehler, als Zeilenfehler oder als beides angezeigt werden. Wofür haben Sie sich entschieden, und was würde ein Benutzer, der nie mit der Maus über ein Icon fährt, trotzdem sehen?

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:

- ✓ Die E-Mail-Regel eines Kommilitonen liest `grid.Rows[e.RowIndex].Cells[e.ColumnIndex].Value` und „funktioniert beim Testen“. Beschreiben Sie genau die Bearbeitung, die durchrutscht, und warum.
- ✓ Ihre Altersregel prüft den Bereich 0 bis 120. Was sollte sie tun, wenn der Text überhaupt keine Zahl ist, und warum wird ein `ValidationError`-Handler trotzdem gebraucht, selbst wenn die Regel „abc“ selbst ablehnt?
- ✓ Die `Required`-Regel könnte als Zellfehler, als Zeilenfehler oder als beides angezeigt werden. Wofür haben Sie sich entschieden, und was würde ein Benutzer, der nie mit der Maus über ein Icon fährt, trotzdem sehen?

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 6 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 6 auf GitHub öffnen](#)

Oder klonen Sie alle Beispiele mit `git clone https://github.com/iceteagroup/LearnWisej-Samples` und öffnen Sie [Validation in Wisej.NET Course/Module 6](#).



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 ValidationClinic-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 ValidationClinic-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 ValidationClinic-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.