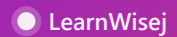


KI-Coding-Übung

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.



Der Assistent liefert die schnellen Hände. Der Architekt bleiben Sie.

Ein Assistent erstellt das Beispielprojekt dieses Moduls in wenigen Minuten. Was er nicht kann, ist zu entscheiden, ob das Ergebnis der richtige Entwurf für Ihre Anwendung ist – diese Entscheidung bleibt bei Ihnen. Das Urteilsvermögen bringen Sie mit: Instruieren Sie ihn sorgfältig und prüfen Sie dann, was zurückkommt, anhand dessen, was Ihnen Modul 2 gerade vermittelt hat.

Worum Sie bitten

Machen Sie aus dem Kontaktformular von ValidationClinic ein Formular mit programmatischer Validierung. Legen Sie für das Formular einen ErrorProvider mit gesetztem ContainerControl, NeverBlink und rechtsbündigem Icon an, schreiben Sie einen FieldValidation-Record sowie die Hilfsmethoden ValidateName, ValidateEmail und ValidatePhone, die eine Meldung zurückgeben, statt die UI anzufassen, und eine Apply-Methode, die für jedes Ergebnis SetError aufruft und ein validationSummaryLabel füllt, das nur sichtbar ist, solange Fehler bestehen. Verwenden Sie InvalidMessage am Feld für die Kundennummer, um die beiden Feedback-Varianten zu vergleichen, und stellen Sie sicher, dass das Korrigieren eines Feldes nur dessen eigenen Fehler entfernt.

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

- ✓ ErrorProvider auf Formularebene mit konfiguriertem ContainerControl, BlinkStyle und Icon-Ausrichtung
- ✓ FieldValidation-Record und die Hilfsmethoden ValidateName, ValidateEmail und ValidatePhone
- ✓ Apply-Methode, die den Fehler jedes Controls setzt oder entfernt und zurückgibt, ob alle Felder gültig sind
- ✓ validationSummaryLabel, das jedes aktuelle Problem auflistet und ausgeblendet ist, wenn es keine gibt
- ✓ Ein Feld, das InvalidMessage statt ErrorProvider verwendet, mit einer Notiz, die beide vergleicht

Warum es so aufgebaut ist

Modul 1 hat dem Aufnahmeformular von ValidationClinic zwei Validating -Handler gegeben, und für zwei Felder funktioniert das. Ein echtes Kontaktformular hat einen Namen, eine E-Mail-Adresse, eine Telefonnummer, eine Kundennummer und mehr, und wenn jedes Feld seinen eigenen handgeschriebenen

Handler mit seiner eigenen Art bekommt, eine Meldung anzuzeigen, wird das Formular schnell uneinheitlich: Ein Feld zeigt ein Icon, ein anderes gar nichts, ein drittes lässt einen alten Fehler stehen, obwohl der Benutzer ihn schon korrigiert hat. Dieses Modul ersetzt diesen Wildwuchs durch ein einziges Muster: Kleine Hilfsmethoden entscheiden, was falsch ist, eine Methode zeigt das Ergebnis an, und eine sichtbare Zusammenfassung sagt jedem Benutzer, was zu korrigieren ist, ohne dass er mit der Maus über etwas fahren muss.

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
ErrorProvider	Eine Extender-Komponente, die neben jedem Control, dem Sie eine Meldung geben, ein Fehler-Icon zeichnet und diese Meldung als Tooltip anzeigt; eine Instanz bedient das gesamte Formular.
ContainerControl	Das Formular oder User-Control, zu dem der Provider gehört; setzen Sie es auf this, weiß der Provider, auf welcher Oberfläche er seine Icons zeichnet.
setError(control, message)	Zeigt message neben control an; ein Aufruf mit einem leeren String oder null entfernt den Fehler nur für dieses eine Control.
getError, hasErrors, clear()	Die aktuelle Meldung eines Controls lesen, abfragen, ob im Formular noch ein Fehler besteht, oder alle Fehler auf einmal entfernen, wenn das gesamte Formular zurückgesetzt wird.
InvalidMessage	Eine Eigenschaft an Editor-Controls, die das Interface IValidation unterstützen; das Control zeigt die Meldung selbst als Ungültig-Tooltip an, ohne dass ein separater Provider beteiligt ist.
FieldValidation -Record	Ein winziger Ergebnistyp mit Control, Feldname und Meldung, wobei eine leere Meldung „gültig“ bedeutet; damit liefern Regeln Antworten, statt den Bildschirm zu verändern.
validationSummaryLabel	Ein einfaches Label, das jedes aktuelle Problem als Text auflistet und nur sichtbar ist, solange Fehler bestehen, damit Feedback nie davon abhängt, ein Icon zu sehen oder mit der Maus darüberzufahren.

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 2 geht. Sie können [module-02.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 2 – „Programmatische Validierung und Feedback mit ErrorProvider“ – 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-02.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: Machen Sie aus dem Kontaktformular von ValidationClinic ein Formular mit programmatischer Validierung. Legen Sie für das Formular einen ErrorProvider mit gesetztem ContainerControl, NeverBlink und rechtsbündigem Icon an, schreiben Sie einen FieldValidation-Record sowie die Hilfsmethoden `ValidateName`, `ValidateEmail` und `ValidatePhone`, die eine Meldung zurückgeben, statt die UI anzufassen, und eine Apply-Methode, die für jedes Ergebnis `SetError` aufruft und ein `validationSummaryLabel` füllt, das nur sichtbar ist, solange Fehler bestehen. Verwenden Sie `InvalidMessage` am Feld für die Kundennummer, um die beiden Feedback-Varianten zu vergleichen, und stellen Sie sicher, dass das Korrigieren eines Feldes nur dessen eigenen Fehler entfernt.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - ErrorProvider auf Formularebene mit konfiguriertem ContainerControl, BlinkStyle und Icon-Ausrichtung
 - FieldValidation-Record und die Hilfsmethoden `ValidateName`, `ValidateEmail` und `ValidatePhone`
 - Apply-Methode, die den Fehler jedes Controls setzt oder entfernt und zurückgibt, ob alle Felder gültig sind
 - `validationSummaryLabel`, das jedes aktuelle Problem auflistet und ausgeblendet ist, wenn es keine gibt
 - Ein Feld, das `InvalidMessage` statt ErrorProvider verwendet, mit einer Notiz, die beide vergleicht
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:

- Der `txtName_Validating`-Handler eines Teamkollegen ruft `errorProvider.Clear()` auf, sobald der Name gültig wird. Was geht für einen Benutzer schief, der außerdem eine ungültige E-Mail-Adresse eingegeben hat, und was sollte der Handler stattdessen aufrufen?

- Das Feld für die Kundennummer zeigt seine Meldung über `InvalidMessage` an. Was müsste das Zusammenfassungs-Label über dieses Feld wissen, um es ebenfalls aufzulisten, und was sagt Ihnen das darüber, wann ein gemeinsamer Provider die bessere Wahl ist?
- Ein Benutzer gibt eine korrekt aufgebaute E-Mail-Adresse ein, die bereits zu einem anderen Kontakt gehört. Ist das Feld-Feedback oder Feedback zu einer Geschäftsregel, wann sollte die Prüfung laufen, und wo sollte die Meldung erscheinen?

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:

- ✓ Der `txtName_Validating` -Handler eines Teamkollegen ruft `errorProvider.Clear()` auf, sobald der Name gültig wird. Was geht für einen Benutzer schief, der außerdem eine ungültige E-Mail-Adresse eingegeben hat, und was sollte der Handler stattdessen aufrufen?
- ✓ Das Feld für die Kundennummer zeigt seine Meldung über `InvalidMessage` an. Was müsste das Zusammenfassungs-Label über dieses Feld wissen, um es ebenfalls aufzulisten, und was sagt Ihnen das darüber, wann ein gemeinsamer Provider die bessere Wahl ist?
- ✓ Ein Benutzer gibt eine korrekt aufgebaute E-Mail-Adresse ein, die bereits zu einem anderen Kontakt gehört. Ist das Feld-Feedback oder Feedback zu einer Geschäftsregel, wann sollte die Prüfung laufen, und wo sollte die Meldung erscheinen?

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 2 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 2 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 2](#).



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.

- ✓ **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."*
- ✓ **Beweisen Sie, dass es wirklich funktioniert.** *"Geh dieses Projekt Ergebnis für Ergebnis anhand der Modulspezifikation unter <https://learnwisej.com/assets/courses/validation-in-wisej-net/specs/module-02.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."*



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.