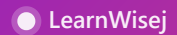


KI-Coding-Übung

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

Legen Sie in ValidationClinic das Formular zur Kundenaufnahme mit den TextBoxen txtName und txtEmail, einem Speichern-Button und einem Abbrechen-Button an. Hängen Sie Validating-Handler an, die e.Cancel = true setzen und eine ErrorProvider-Meldung ausgeben, wenn der Name leer ist oder die E-Mail-Adresse kein @ enthält, die Meldung bei einem gültigen Wert wieder entfernen und den Namen in seinem Validated-Handler trimmen. Legen Sie den AutoValidate-Modus des Formulars fest, sorgen Sie dafür, dass Abbrechen keine Validierung auslöst, damit der Benutzer das Formular jederzeit verlassen kann, und rufen Sie ValidateChildren() am Anfang von btnSave_Click auf, damit das Speichern abbricht, sobald ein Feld ungültig ist. Schließen Sie mit einer kurzen Lab-Notiz ab, die das gewählte Fokusverhalten erklärt.

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

- ✓ Formular zur Kundenaufnahme mit txtName und txtEmail, einem Speichern-Button und einem Abbrechen-Button
- ✓ Validating-Handler, die für einen leeren Namen und eine ungültige E-Mail-Adresse e.Cancel = true und eine Fehlermeldung setzen
- ✓ Validated-Handler, der den akzeptierten Namen trimmt
- ✓ Speichern-Handler, der ValidateChildren() aufruft und abbricht, wenn es false zurückgibt
- ✓ Lab-Notiz, die den gewählten AutoValidate-Modus nennt und erklärt, warum Abbrechen auch bei einem ungültigen Formular funktioniert

Warum es so aufgebaut ist

Das Formular zur Kundenaufnahme in ValidationClinic akzeptiert derzeit alles: einen leeren Namen, eine E-Mail-Adresse ohne @ oder einen Klick auf Speichern in einem Formular, das niemand ausgefüllt hat. Die erste Idee der meisten Einsteiger ist eine if -Anweisung im Speichern-Button, aber damit erhält der

Benutzer während der Arbeit keinerlei Feedback, und Speichern wird ganz am Ende zu einer Wand aus Fehlermeldungen. Dieses Modul bringt die Validierung dorthin, wo sie in einem Wisej.NET-Formular hingehört: an jeden Editor, wenn der Benutzer ihn verlässt, und noch einmal, wenn der Benutzer auf Speichern klickt.

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
Validierung	Die Entscheidung, ob ein Wert akzeptabel ist, bevor die Anwendung ihn verwendet, gefolgt von sichtbarem Feedback und, wenn der Wert falsch ist, der Weigerung, fortzufahren.
Validating	Das Event, das ein Control auslöst, wenn der Benutzer es verlassen will (oder wenn Code eine Validierung anfordert); sein Handler liest den Wert und ist die einzige Stelle, an der über Annehmen oder Ablehnen entschieden wird.
CancelEventArgs.Cancel	Das Flag, das ein Validating -Handler auf true setzt, um den Wert abzulehnen: Das Control gilt als ungültig, der Fokus kann darin gehalten werden, und ValidateChildren() meldet einen Fehlschlag.
Validated	Das Event, das erst ausgelöst wird, nachdem Validating ohne Abbruch beendet wurde; der Ort für Aufräumarbeiten wie das Trimmen von Text, niemals für das Speichern.
AutoValidate	Eine Eigenschaft eines Formulars, einer Page oder eines User-Controls, die festlegt, was mit dem Fokus passiert, wenn ein untergeordnetes Control die Validierung nicht besteht.
CausesValidation	Ein Schalter pro Control; steht er bei einem Button wie Abbrechen auf false, validiert ein Klick darauf das Feld nicht, das der Benutzer gerade verlässt.
ValidateChildren()	Eine Methode eines Containers, die die Validierung für seine untergeordneten Controls ausführt, auch für Felder, die der Benutzer nie besucht hat, und false zurückgibt, wenn irgendein Handler abgebrochen hat.

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 1 geht. Sie können [module-01.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 1 – „Grundlagen der Validierung in Wisej.NET“ – des LearnWisej-Kurses „Validierung in Wisej.NET“ 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/validation-in-wisej-net/specs/module-01.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: Legen Sie in ValidationClinic das Formular zur Kundenaufnahme mit den TextBoxen `txtName` und `txtEmail`, einem Speichern-Button und einem Abbrechen-Button an. Hängen Sie Validating-Handler an, die `e.Cancel = true` setzen und eine ErrorProvider-Meldung ausgeben, wenn der Name leer ist oder die E-Mail-Adresse kein @ enthält, die Meldung bei einem gültigen Wert wieder entfernen und den Namen in seinem Validated-Handler trimmen. Legen Sie den AutoValidate-Modus des Formulars fest, sorgen Sie dafür, dass Abbrechen keine Validierung auslöst, damit der Benutzer das Formular jederzeit verlassen kann, und rufen Sie `ValidateChildren()` am Anfang von `btnSave_Click` auf, damit das Speichern abbricht, sobald ein Feld ungültig ist. Schließen Sie mit einer kurzen Lab-Notiz ab, die das gewählte Fokusverhalten erklärt.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Formular zur Kundenaufnahme mit `txtName` und `txtEmail`, einem Speichern-Button und einem Abbrechen-Button
 - Validating-Handler, die für einen leeren Namen und eine ungültige E-Mail-Adresse `e.Cancel = true` und eine Fehlermeldung setzen
 - Validated-Handler, der den akzeptierten Namen trimmt
 - Speichern-Handler, der `ValidateChildren()` aufruft und abbricht, wenn es `false` zurückgibt
 - Lab-Notiz, die den gewählten AutoValidate-Modus nennt und erklärt, warum Abbrechen auch bei einem ungültigen Formular funktioniert
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:

- Sie entfernen `e.Cancel = true` aus `txtEmail_Validating`, behalten aber den Aufruf von `SetError`. Beschreiben Sie genau, was der Benutzer sieht, wenn er „`maria.example.com`“ mit Tab verlässt und dann auf Speichern klickt.

- Welche Ihrer Prüfungen laufen, während der Benutzer bearbeitet, und welche laufen nur, weil Speichern `ValidateChildren()` aufgerufen hat? Nennen Sie eine Benutzeraktion, die Speichern mit einem Feld erreicht, dessen Handler nie gelaufen ist.
- Sie stellen das Formular von `EnablePreventFocusChange` auf `EnableAllowFocusChange` um. Was ändert sich für den Benutzer, was bleibt gleich, und spielt `btnCancel.CausesValidation = false` dann noch eine Rolle?

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:

- ✓ Sie entfernen `e.Cancel = true` aus `txtEmail_Validating`, behalten aber den Aufruf von `SetError`. Beschreiben Sie genau, was der Benutzer sieht, wenn er „`maria.example.com`“ mit Tab verlässt und dann auf Speichern klickt.
- ✓ Welche Ihrer Prüfungen laufen, während der Benutzer bearbeitet, und welche laufen nur, weil Speichern `ValidateChildren()` aufgerufen hat? Nennen Sie eine Benutzeraktion, die Speichern mit einem Feld erreicht, dessen Handler nie gelaufen ist.
- ✓ Sie stellen das Formular von `EnablePreventFocusChange` auf `EnableAllowFocusChange` um. Was ändert sich für den Benutzer, was bleibt gleich, und spielt `btnCancel.CausesValidation = false` dann noch eine Rolle?

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 1 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 1 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 1](#).



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.