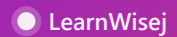


KI-Coding-Übung

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.



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

Alle anderen Abschnitte dieses Moduls erklären, wie „Validierung gebundener Daten und Modellfehler“ 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

Binden Sie das Aufnahmeformular von ValidationClinic über eine `contactBindingSource` an ein `ContactEditModel` und versehen Sie dessen Eigenschaften `Name`, `Email` und `Age` mit den Annotationen `Required`, `StringLength`, `EmailAddress` und `Range`. Rufen Sie in `btnSave_Click` zuerst `contactBindingSource.EndEdit()` auf, führen Sie `Validator.TryValidateObject` mit `validateAllProperties` auf `true` aus und ordnen Sie den Member-Namen jedes `ValidationResult` über eine Feldzuordnung und die Validierungszusammenfassung seinem Control zu. Implementieren Sie dann `IDataErrorInfo` auf einem zweiten Modell, richten Sie den `ErrorProvider` mit `BindToDataAndErrors` auf die Binding-Source aus und zeigen Sie, dass dieselben Fehler über Data Binding erscheinen, ohne handgeschriebenen `SetError`-Aufruf.

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

- ✓ `ContactEditModel` mit `DataAnnotations`, über eine `BindingSource` an das Formular gebunden
- ✓ Speichern-Handler, der `EndEdit` aufruft, bevor `Validator.TryValidateObject` läuft
- ✓ Feldzuordnung, die Member-Namen aus `ValidationResult` in Control-Fehler und Zeilen der Zusammenfassung umsetzt
- ✓ `IDataErrorInfo`-Modell, dessen Fehler über einen datengebundenen `ErrorProvider` erscheinen
- ✓ Aufruf von `BindToDataAndErrors`, wenn sich die Datenquelle des Providers zur Laufzeit ändert

Warum es so aufgebaut ist

Alles, was ValidationClinic bisher prüft, steckt im Aufnahmeformular: hier ein `Validating` -Handler, dort eine Regel am `Validation` -Extender, ein Aufruf von `ContactValidator` in `btnSave_Click`. Ein Kontakt kommt aber nicht nur über dieses Formular herein. Er kann im Kontakt-Grid bearbeitet, aus einer Datei importiert oder von einem Service angelegt werden, und keiner dieser Wege führt die Handler des Formulars aus. Dieses

Modul verlagert die grundlegenden Regeln auf das ContactEditModel selbst, sodass die Daten ihre eigene Definition von „gültig“ mitbringen und jede Maske ihnen dieselbe Frage stellen kann.

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	Eine Komponente zwischen den Controls und Ihrem Objekt: Ihre DataSource ist das Modell, und jedes Control bindet sich über sie an eine Eigenschaft, statt von Hand kopiert zu werden.
EndEdit()	Überträgt den Wert, den der Benutzer noch bearbeitet, aus dem Control in das Modell, sodass die Validierung liest, was auf dem Bildschirm steht, und nicht den zuletzt übernommenen Wert.
DataAnnotations	Attribute wie [Required], [StringLength], [EmailAddress] und [Range] auf Modelleigenschaften, jeweils mit einer ErrorMessage, die für den Benutzer geschrieben ist.
Validator.TryValidateObject	Die.NET-Methode, die diese Attribute auf ein Objekt anwendet und eine Liste von ValidationResult -Objekten füllt; sie weiß nichts über Controls oder Wisej.NET.
MemberNames	Die Eigenschaftsnamen, zu denen ein ValidationResult gehört; sie sind die einzige Verbindung zwischen einer Fehlermeldung und dem Control, das sie anzeigen soll.
IDataErrorInfo	Ein Interface, über das ein Modell per Indexer die Frage „Was stimmt mit dieser Eigenschaft nicht?“ beantwortet, sodass die Binding-Schicht Fehler abfragen kann, statt dass Ihr Code sie hineinschiebt.
Datengebundener ErrorProvider	Ein ErrorProvider, dessen DataSource dieselbe Binding-Source ist wie die der Controls, sodass er die Fehler des Modells neben den gebundenen Controls anzeigt, ganz ohne SetError -Aufruf.

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



Claude verwenden

Öffnet [claude.ai](#) mit demselben Prompt. Wenn Sie Claude Code nutzen, können Sie

Codex; jeder Tarif mit Websuche ruft die Spezifikation ab.

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 5 – „Validierung gebundener Daten und Modellfehler“ – 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-05.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: Binden Sie das Aufnahmeformular von ValidationClinic über eine `contactBindingSource` an ein `ContactEditModel` und versehen Sie dessen Eigenschaften Name, Email und Age mit den Annotationen `Required`, `StringLength`, `EmailAddress` und `Range`. Rufen Sie in `btnSave_Click` zuerst `contactBindingSource.EndEdit()` auf, führen Sie `Validator.TryValidateObject` mit `validateAllProperties` auf `true` aus und ordnen Sie den Member-Namen jedes `ValidationResult` über eine Feldzuordnung und die Validierungszusammenfassung seinem Control zu. Implementieren Sie dann `IDataErrorInfo` auf einem zweiten Modell, richten Sie den `ErrorProvider` mit `BindToDataAndErrors` auf die `BindingSource` aus und zeigen Sie, dass dieselben Fehler über Data Binding erscheinen, ohne handgeschriebenen `SetError`-Aufruf.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - `ContactEditModel` mit `DataAnnotations`, über eine `BindingSource` an das Formular gebunden
 - Speichern-Handler, der `EndEdit` aufruft, bevor `Validator.TryValidateObject` läuft
 - Feldzuordnung, die Member-Namen aus `ValidationResult` in Control-Fehler und Zeilen der Zusammenfassung umsetzt
 - `IDataErrorInfo`-Modell, dessen Fehler über einen datengebundenen `ErrorProvider` erscheinen
 - Aufruf von `BindToDataAndErrors`, wenn sich die Datenquelle des Providers zur Laufzeit ändert
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:

- Eine Testerin tippt eine neue E-Mail-Adresse ein und drückt zum Speichern die Eingabetaste, geprüft und gespeichert wird aber die alte, gültige Adresse. Welche Zeile fehlt, und warum sieht das Formular auf dem Bildschirm richtig aus, während das Modell falsch ist?

- Sie entfernen `validateAllProperties: true`, und jedes Speichern „funktioniert“ weiterhin. Welche Ihrer vier Annotationen laufen nicht mehr, und welche fehlerhaften Daten können jetzt das Repository erreichen?
- Der Import-Service legt Kontakte an, ohne das Formular zu öffnen. Welche Regeln dieses Moduls schützen diesen Weg weiterhin, und welche laufen nur, weil ein Benutzer auf Speichern geklickt hat?

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:

- ✓ Eine Testerin tippt eine neue E-Mail-Adresse ein und drückt zum Speichern die Eingabetaste, geprüft und gespeichert wird aber die alte, gültige Adresse. Welche Zeile fehlt, und warum sieht das Formular auf dem Bildschirm richtig aus, während das Modell falsch ist?
- ✓ Sie entfernen `validateAllProperties: true`, und jedes Speichern „funktioniert“ weiterhin. Welche Ihrer vier Annotationen laufen nicht mehr, und welche fehlerhaften Daten können jetzt das Repository erreichen?
- ✓ Der Import-Service legt Kontakte an, ohne das Formular zu öffnen. Welche Regeln dieses Moduls schützen diesen Weg weiterhin, und welche laufen nur, weil ein Benutzer auf Speichern geklickt hat?

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 [Validation in Wisej.NET 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.

- ✓ **Prüfen Sie es wie einen Pull-Request.** *"Führ mich durch diesen ValidationClinic-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 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."*



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.