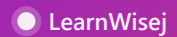


KI-Coding-Übung

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.



Instruieren, lesen, dann auf die Probe stellen.

Diese Übung hat drei Teile, und nur der erste dreht sich um den Prompt. Sie instruieren einen Assistenten mit der Modulspezifikation und erhalten eine ValidationClinic-Projektmappe zurück; Sie lesen, was er geschrieben hat, und entscheiden, ob es wirklich tut, was das Modul verlangt; dann treiben Sie es weiter, als das Modul verlangt hat. Im zweiten und dritten Teil steckt das eigentliche Lernen.

Worum Sie bitten

Ersetzen Sie die handgeschriebenen Feld-Handler in ValidationClinic durch den Validation-Extender. Fügen Sie dem Aufnahmeformular eine Validation-Komponente und einen ErrorProvider hinzu, setzen Sie ContainerControl und validation.ErrorProvider und rufen Sie dann SetValidationRules in einer Methode ConfigureValidationRules auf: Required und danach Email auf txtEmail, Required und danach Integer auf txtAge, Required und danach Telephone auf txtPhone, Currency auf txtCreditLimit und eine Regex-Regel für die Kundennummer. Setzen Sie Required überall an die erste Stelle, damit die hilfreichste Meldung gewinnt, hängen Sie sich an das Validating-Event des Extenders, um die Zusammenfassung zu aktualisieren, und verschieben Sie die Regel-Arrays in eine kleine wiederverwendbare Hilfsklasse, damit ein anderes Formular dieselbe Konfiguration anwenden kann.

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

- ✓ Validation-Komponente, die mit dem Formular und einem ErrorProvider als Anzeigeziel verbunden ist
- ✓ Methode ConfigureValidationRules, die SetValidationRules für E-Mail, Alter, Telefon, Kreditlimit und Kundennummer aufruft
- ✓ Required-Regel an erster Stelle auf jedem Pflichtfeld, mit verständlichem InvalidMessage-Text
- ✓ Validating-Handler des Extenders, der die Validierungszusammenfassung mit den Regeln synchron hält
- ✓ Wiederverwendbare Hilfsklasse für die Regelkonfiguration, die ein zweites Formular aufrufen kann

Warum es so aufgebaut ist

Am Ende von Modul 2 funktioniert das Aufnahmeformular von ValidationClinic, aber sehen Sie sich an, wie viel Code dafür nötig war: eine Hilfsmethode für den Namen, eine für die E-Mail-Adresse, eine für die Telefonnummer, und jede wiederholt dieselben Prüfungen „Ist es leer, hat es die richtige Form?“. Kommen Alter, Kreditlimit und Kundennummer hinzu, wächst das Formular um eine weitere Handvoll fast identischer Methoden. Die Prüfungen sind nicht falsch; sie werden nur für jedes Feld und jedes Formular erneut abgetippt. Dieses Modul ersetzt diese Wiederholung durch den Validation -Extender: Sie beschreiben, wie ein gültiges Feld aussieht, und Wisej.NET führt die Prüfungen für Sie aus.

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
Validation -Extender	Eine nicht sichtbare Komponente, die Sie einem Formular hinzufügen; sie hängt sich an die Events Validating und Validated jedes Controls, das Regeln hat, sodass Sie keinen Handler pro Feld mehr schreiben.
erweiterte Eigenschaft	Eine Eigenschaft, die eine Komponente anderen Controls hinzufügt; sobald der Extender auf dem Formular liegt, zeigt jedes compatible Control im Eigenschaftenfenster einen Eintrag ValidationRules.
SetValidationRules	Das Gegenstück zum Designer im Code: validation.SetValidationRules(txtEmail, new ValidationRule[] { ... }) gibt einem Control zur Laufzeit seine Regelliste.
integrierte Regeln	Fertige Prüfungen für gängige Eingaben: RequiredValidationRule, EmailValidationRule, IntegerValidationRule, DecimalValidationRule, CurrencyValidationRule, TelephoneValidationRule und RegexValidationRule.
Regelreihenfolge	Regeln laufen in der Reihenfolge, in der Sie sie auflisten; die erste, die fehlschlägt, stoppt die übrigen und bricht das Validating -Event des Controls ab.
Anzeigeziel	Wo die InvalidMessage der fehlgeschlagenen Regel erscheint: am Control selbst, über ein ErrorProvider -Icon oder als Text in einem Label, jeweils erreicht über validation.ErrorProvider.
Hilfsklasse für die Regelkonfiguration	Eine kleine Methode, die die Regel-Arrays für gängige Felder aufbaut, sodass ein zweites Formular durch ihren Aufruf dieselben Prüfungen und dieselben Formulierungen erhält.

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 3 geht. Sie können [module-03.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 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 3 – „Integrierte Validierungsregeln und der Validation-Extender“ – 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-03.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: Ersetzen Sie die handgeschriebenen Feld-Handler in ValidationClinic durch den Validation-Extender. Fügen Sie dem Aufnahmeformular eine Validation-Komponente und einen ErrorProvider hinzu, setzen Sie ContainerControl und validation.ErrorProvider und rufen Sie dann SetValidationRules in einer Methode ConfigureValidationRules auf: Required und danach Email auf txtEmail, Required und danach Integer auf txtAge, Required und danach Telephone auf txtPhone, Currency auf txtCreditLimit und eine Regex-Regel für die Kundennummer. Setzen Sie Required überall an die erste Stelle, damit die hilfreichste Meldung gewinnt, hängen Sie sich an das Validating-Event des Extenders, um die Zusammenfassung zu aktualisieren, und verschieben Sie die Regel-Arrays in eine kleine wiederverwendbare Hilfsklasse, damit ein anderes Formular dieselbe Konfiguration anwenden kann.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Validation-Komponente, die mit dem Formular und einem ErrorProvider als Anzeigeziel verbunden ist
 - Methode ConfigureValidationRules, die SetValidationRules für E-Mail, Alter, Telefon, Kreditlimit und Kundennummer aufruft
 - Required-Regel an erster Stelle auf jedem Pflichtfeld, mit verständlichem InvalidMessage-Text
 - Validating-Handler des Extenders, der die Validierungszusammenfassung mit den Regeln synchron hält
 - Wiederverwendbare Hilfsklasse für die Regelkonfiguration, die ein zweites Formular aufrufen kann
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 vertauschen die Reihenfolge, sodass auf txtEmail die Email-Regel vor Required läuft.

Beschreiben Sie genau, was ein Benutzer sieht, nachdem er das Feld leer verlassen hat, und warum diese Meldung weniger nützlich ist.

– Die Hilfsmethoden aus Modul 2 und der Extender enden beide mit einem Icon neben dem Feld.

Was schreiben Sie nicht mehr von Hand, und was mussten Sie trotzdem selbst entscheiden?

– Eine Kollegin möchte „Das Enddatum muss nach dem Startdatum liegen“ in ClinicRules aufnehmen. Warum ist der Extender der falsche Ort für diese Prüfung, und wann würde sie nicht laufen?

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 vertauschen die Reihenfolge, sodass auf txtEmail die Email-Regel vor Required läuft.
Beschreiben Sie genau, was ein Benutzer sieht, nachdem er das Feld leer verlassen hat, und warum diese Meldung weniger nützlich ist.
- ✓ Die Hilfsmethoden aus Modul 2 und der Extender enden beide mit einem Icon neben dem Feld.
Was schreiben Sie nicht mehr von Hand, und was mussten Sie trotzdem selbst entscheiden?
- ✓ Eine Kollegin möchte „Das Enddatum muss nach dem Startdatum liegen“ in ClinicRules aufnehmen. Warum ist der Extender der falsche Ort für diese Prüfung, und wann würde sie nicht laufen?

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 3 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 3 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 3](#).



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.

- ✓ **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-03.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."*
- ✓ **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."*



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.