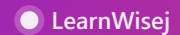


KI-Coding-Übung

Erstellen Sie das Beispielprojekt GlobalDesk 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.



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

Erstellen Sie die GlobalDesk-Dashboard-Seite mit einer Willkommensüberschrift, einer Navigationsschaltfläche Customers und einem kleinen Vorschaubereich für die Kultur. Fügen Sie Resources/Strings.resx als neutrale Ressource mit mindestens zwölf stabilen semantischen Schlüsseln wie Navigation.Customers und CustomerEditor.Save hinzu, schreiben Sie einen statischen Texts-Helper über Wisej.Resources.ResourceManager, der bei einem fehlenden Schlüssel eine in Klammern gesetzte Markierung zurückgibt, und ersetzen Sie jede fest verdrahtete Beschriftung auf der Seite durch einen Texts.Get-Aufruf. Füllen Sie anschließend den Vorschaubereich mit echten Werten: Formatieren Sie das heutige Datum, eine Menge und einen Währungsbetrag mit Application.CurrentCulture, statt die Zeichenfolgen von Hand zusammenzusetzen, und schließen Sie mit einer kurzen Lab-Notiz ab, die den gespeicherten Wert von seiner kultursensiblen Darstellung trennt.

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

- ✓ GlobalDesk-Dashboard-Seite, deren Beschriftungen alle aus Ressourcenschlüsseln stammen, ohne fest verdrahteten sichtbaren Text
- ✓ Neutrale Datei Resources/Strings.resx mit mindestens zwölf stabilen semantischen Schlüsseln
- ✓ Texts-Helper über Wisej.Resources.ResourceManager, der einen fehlenden Schlüssel markiert, statt eine leere Zeichenfolge zurückzugeben
- ✓ Vorschaubereich für die Kultur, der ein Datum, eine Zahl und einen Währungsbetrag mit Application.CurrentCulture formatiert
- ✓ Lab-Notiz, die erklärt, welche Werte Daten sind und welche kultursensible Darstellung

Warum es so aufgebaut ist

GlobalDesk ist ein kleiner Service Desk, geschrieben für ein einziges Büro. Jede Überschrift, jede Schaltflächenbeschriftung und jede Bestätigungsmeldung ist ein englisches Literal, direkt in den C#-Code getippt, und das Dashboard gibt Beträge und Datumsangaben so aus, wie eine US-Maschine es zufällig tut. Wenn eine deutsche Niederlassung eröffnet, hat dieser Text nirgendwohin zu gehen — und wer damit beginnt, Beschriftungen von Hand zu ändern, trifft auf die zweite Hälfte des Problems: Eine deutsche Beschriftung über einem Betrag, der weiterhin 1,850.75 lautet, ergibt noch keinen deutschen Bildschirm.

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
Lokalisierung	Eine Anwendung an Sprache und Kultur des Benutzers anpassen: übersetzter Text, korrekt formatierte Datumsangaben, Zahlen und Währungen, Platz für längere Wörter und ein Layout, das sich für Rechts-nach-links-Sprachen spiegeln lässt.
Kulturname	Ein Bezeichner wie en, de-DE oder fr-CA, der eine Sprache und optional eine Region benennt; der Sprachteil wählt, welche Ressourcen bevorzugt werden, der vollständige Name steuert die Formatierungskonventionen.
neutrale Ressource	Der vollständige Rückfallsatz an Ressourcen, abgelegt in einer Datei wie Resources/Strings.resx ohne Kultur im Namen; jeder Schlüssel, nach dem die Anwendung fragt, muss hier vorhanden sein.
Wisej.Resources.ResourceManager	Das Objekt, das einen Ressourcen-Basisnamen und eine Assembly entgegennimmt und den Wert für einen Schlüssel in der am besten passenden Kultur zurückgibt, sodass Ihr Code nie selbst eine Datei öffnet.
Kulturrückfall	Die.NET-Regel zum Auffinden eines Werts: erst die vollständige Kultur versuchen, dann die reine Sprachkultur, dann die neutrale Ressource — und den ersten Treffer nehmen, der den Schlüssel hat.
semantischer Ressourcenschlüssel	Ein stabiler Name, der die Bedeutung beschreibt statt der Formulierung, etwa CustomerEditor.Save oder Validation.Required, sodass der englische Text umformuliert werden kann, ohne eine andere Sprache anzufassen.
Application.CurrentCulture	Die Kultur der aktuellen Wisej.NET-Sitzung; zwei gleichzeitig angemeldete Benutzer können zwei verschiedene Werte halten, und genau deshalb darf die Kultur niemals in einem statischen Feld leben.

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

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 GlobalDesk 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 Lokalisierung: Kultur, Sprache und Ressourcen“ – des LearnWisej-Kurses „Lokalisierung 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/localization-in-wisej-net/specs/module-01.md>

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

1. Projekt „GlobalDesk“, 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: Erstellen Sie die GlobalDesk-Dashboard-Seite mit einer Willkommensüberschrift, einer Navigationsschaltfläche Customers und einem kleinen Vorschaubereich für die Kultur. Fügen Sie `Resources/Strings.resx` als neutrale Ressource mit mindestens zwölf stabilen semantischen Schlüsseln wie `Navigation.Customers` und `CustomerEditor.Save` hinzu, schreiben Sie einen statischen Texts-Helper über `Wisej.Resources.ResourceManager`, der bei einem fehlenden Schlüssel eine in Klammern gesetzte Markierung zurückgibt, und ersetzen Sie jede fest verdrahtete Beschriftung auf der Seite durch einen `Texts.Get`-Aufruf. Füllen Sie anschließend den Vorschaubereich mit echten Werten: Formatieren Sie das heutige Datum, eine Menge und einen Währungsbetrag mit `Application.CurrentCulture`, statt die Zeichenfolgen von Hand zusammenzusetzen, und schließen Sie mit einer kurzen Lab-Notiz ab, die den gespeicherten Wert von seiner kultursensiblen Darstellung trennt.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - GlobalDesk-Dashboard-Seite, deren Beschriftungen alle aus Ressourcenschlüsseln stammen, ohne fest verdrahteten sichtbaren Text
 - Neutrale Datei `Resources/Strings.resx` mit mindestens zwölf stabilen semantischen Schlüsseln
 - Texts-Helper über `Wisej.Resources.ResourceManager`, der einen fehlenden Schlüssel markiert, statt eine leere Zeichenfolge zurückzugeben
 - Vorschaubereich für die Kultur, der ein Datum, eine Zahl und einen Währungsbetrag mit `Application.CurrentCulture` formatiert
 - Lab-Notiz, die erklärt, welche Werte Daten sind und welche kultursensible Darstellung
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 ändern die englische Formulierung der Schaltfläche Save von „Save“ zu „Save customer“.

Welche Dateien müssen sich ändern, wenn Ihr Schlüssel `CustomerEditor.Save` lautet, und welche hätten sich ändern müssen, wenn der Schlüssel der englische Satz selbst gewesen wäre?

- Ihr Texts -Helper gibt für ein Label `[Preview.Amount]` zurück. Nennen Sie zwei verschiedene Fehler, die genau diese Markierung erzeugen, und sagen Sie, wie Sie sie ohne Debugger auseinanderhalten würden.
- Ein Kollege schlägt vor, die Auftragssumme als Zeichenfolge „1.850,75 €“ in der Datenbank zu speichern, damit deutsche Benutzer sie immer korrekt sehen. Erklären Sie, was dabei kaputtgeht, und welcher Teil dieser Zeichenfolge Daten sind und welcher Darstellung.

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 ändern die englische Formulierung der Schaltfläche Save von „Save“ zu „Save customer“. Welche Dateien müssen sich ändern, wenn Ihr Schlüssel `CustomerEditor.Save` lautet, und welche hätten sich ändern müssen, wenn der Schlüssel der englische Satz selbst gewesen wäre?
- ✓ Ihr Texts -Helper gibt für ein Label `[Preview.Amount]` zurück. Nennen Sie zwei verschiedene Fehler, die genau diese Markierung erzeugen, und sagen Sie, wie Sie sie ohne Debugger auseinanderhalten würden.
- ✓ Ein Kollege schlägt vor, die Auftragssumme als Zeichenfolge „1.850,75 €“ in der Datenbank zu speichern, damit deutsche Benutzer sie immer korrekt sehen. Erklären Sie, was dabei kaputtgeht, und welcher Teil dieser Zeichenfolge Daten sind und welcher Darstellung.

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.



Für diesen Kurs gibt es noch kein Referenzbeispiel – Ihres ist das erste

Das [LearnWisej-Beispiel-Repository](#) enthält ausgearbeitete Builds für die meisten Kurse, für diesen aber noch nicht. Es gibt also nichts zum Vergleichen und keine Abkürzung: Die Spezifikation und Ihr eigenes Review sind das ganze Sicherheitsnetz. Behalten Sie, was Sie bauen – und wenn es gut wird, würden wir es gern sehen.

Drei Prompts für den nächsten Schritt

Richten Sie Ihren Assistenten auf das Projekt, das Sie gerade gebaut haben, und probieren Sie diese Prompts aus. Jeder ist eine Übung zum Prüfen und Erweitern, kein Neuschreiben.

- ✓ **Ergänzen Sie den zweiten Fall.** *"Erweitere dieses GlobalDesk-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/localization-in-wisej-net/specs/module-01.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 GlobalDesk-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.