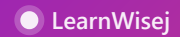


KI-Coding-Übung

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.



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 7 gerade vermittelt hat.

Worum Sie bitten

Bringen Sie das GlobalDesk-Abschlussprojekt zu Ende. Kapseln Sie jede Ressourcensuche in einem LocalizationService mit den Methoden Text, Date, Number und Currency, geben Sie ihm eine Richtlinie für fehlende Schlüssel, die in der Entwicklung einen in Klammern gesetzten Schlüssel anzeigt und ihn im Produktivbetrieb protokolliert, und machen Sie diesen Service zur einzigen Stelle, an der die Oberfläche den ResourceManager berührt. Halten Sie die Domäne heraus: Das Ticket-Modell stellt einen TicketStatus-Wert bereit, und die Oberfläche bildet ihn auf eine lokalisierte Beschriftung ab. Führen Sie dann die Lokalisierungs-Testmatrix über die fertige Anwendung aus – die neutrale Kultur, Deutsch, eine Sprach-Regions-Kultur, die zurückfällt, ein absichtlich fehlender Schlüssel, formatierte Datumsangaben und Währungsbeträge, ein Wechsel zur Laufzeit, ein Durchgang mit langen Zeichenfolgen und die RTL-Kultur – und nutzen Sie dafür einen pseudo-lokalisierten Ressourcensatz, der alles aufdeckt, was noch fest verdrahtet ist. Vergewissern Sie sich, dass die Kulturordner eine Veröffentlichung überstehen, und reichen Sie die QA-Checkliste als Ergebnis des Abschlussprojekts ein.

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

- ✓ LocalizationService mit den Methoden Text, Date, Number und Currency und einer dokumentierten Richtlinie für fehlende Schlüssel
- ✓ Domänenencode, der einen TicketStatus-Wert zurückgibt, den die Oberfläche auf eine lokalisierte Beschriftung abbildet
- ✓ Pseudo-lokalisierter Ressourcensatz, mit dem jede verbliebene fest verdrahtete Zeichenfolge gefunden wird
- ✓ Vollständige Lokalisierungs-Testmatrix für Rückfall, Formatierung, Laufzeitwechsel und RTL
- ✓ Veröffentlichte Ausgabe, auf ihre Kulturordner geprüft, dazu die GlobalDesk-Lokalisierungs-QA-Checkliste

Warum es so aufgebaut ist

Nach sechs Modulen spricht GlobalDesk zwei Sprachen, doch die Mechanik ist verstreut: Designer-Ressourcen auf dem Kundeneditor, ein Helper über den gemeinsamen.resx -Dateien, eine Sprachauswahl, ein CultureChanged -Handler und eine RTL-Vorführung. Noch sagt nichts, was geschieht, wenn ein Schlüssel keine Übersetzung hat, und niemand hat belegt, dass der deutsche Build noch in ein schmales Fenster passt. Ein Lokalisierungssystem für den Produktivbetrieb ist vorhersehbar: Es weiß, woher jede Zeichenfolge kommt, welche Kultur in dieser Sitzung aktiv ist, wie eine fehlende Übersetzung aussieht und was ein Sprachwechsel mit geöffneten Bildschirmen macht. Dieses Modul macht aus diesen vier Antworten einen kleinen Service, eine schriftliche Testmatrix und das fertige Abschlussprojekt.

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
LocalizationService	Eine kleine Klasse, die die Entscheidungen Ihrer Anwendung rund um die.NET-Lokalisierung bündelt, statt sie zu ersetzen: ein Ressourcen-Basisname, eine Richtlinie für fehlende Schlüssel, ein Satz Formatierungsmethoden.
Richtlinie für fehlende Schlüssel	Das gewählte, getestete Verhalten, wenn GetString nichts liefert: eine sichtbare Markierung [Key] während der Entwicklung, ein protokollierter Schlüssel plus neutraler Rückfalltext im Produktivbetrieb.
Produkt-, System- und Diagnosetext	Drei Familien mit drei Eigentümern: Ihre eigenen.resx -Zeichenfolgen, die Beschriftungen von Wisej.NET über Resources-[LANG].resx und Protokoll- oder Ausnahmetext, der in einer einzigen Entwicklersprache bleibt.
kulturfreie Domäne	Das Modell stellt einen Wert wie TicketStatus.Closed bereit und die Oberfläche bildet ihn auf eine lokalisierte Beschriftung ab, sodass Geschäftslogik und ihre Tests nie von einer Sprache abhängen.
Lokalisierungs-Testmatrix	Eine schriftliche Liste von Läufen — neutral, übersetzt, mit Regionsrückfall, fehlender Schlüssel, formatierte Werte, Wechsel zur Laufzeit, lange Zeichenfolgen, RTL, Systembeschriftungen —, sodass Abdeckung eine Checkliste ist und keine Erinnerung.
Pseudo-Lokalisierung	Ein Testressourcensatz, dessen Werte sichtbar verändert und verlängert sind, etwa [!!! Save customer record!!!], gebaut, um Abschneidungen und fest verdrahtete Zeichenfolgen aufzudecken, und nicht, um etwas zu übersetzen.
Satellitenressourcen	.NET kompiliert kulturspezifische Ressourcen in Satellitenassemblies, deshalb müssen die vom Build erzeugten Kulturordner die Anwendung begleiten, wenn Sie sie veröffentlichen oder paketieren.

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 7 geht. Sie können [module-07.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 7 – „Lokalisierungsarchitektur für den Produktivbetrieb, Tests und Abschlussprojekt“ – 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-07.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: Bringen Sie das GlobalDesk-Abschlussprojekt zu Ende. Kapseln Sie jede Ressourcensuche in einem `LocalizationService` mit den Methoden `Text`, `Date`, `Number` und `Currency`, geben Sie ihm eine Richtlinie für fehlende Schlüssel, die in der Entwicklung einen in Klammern gesetzten Schlüssel anzeigt und ihn im Produktivbetrieb protokolliert, und machen Sie diesen Service zur einzigen Stelle, an der die Oberfläche den `ResourceManager` berührt. Halten Sie die Domäne heraus: Das Ticket-Modell stellt einen `TicketStatus`-Wert bereit, und die Oberfläche bildet ihn auf eine lokalisierte Beschriftung ab. Führen Sie dann die Lokalisierungs-Testmatrix über die fertige Anwendung aus – die neutrale Kultur, Deutsch, eine Sprach-Regions-Kultur, die zurückfällt, ein absichtlich fehlender Schlüssel, formatierte Datumsangaben und Währungsbeträge, ein Wechsel zur Laufzeit, ein Durchgang mit langen Zeichenfolgen und die RTL-Kultur – und nutzen Sie dafür einen pseudo-lokalisierten Ressourcensatz, der alles aufdeckt, was noch fest verdrahtet ist. Vergewissern Sie sich, dass die Kulturordner eine Veröffentlichung überstehen, und reichen Sie die QA-Checkliste als Ergebnis des Abschlussprojekts ein.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - `LocalizationService` mit den Methoden `Text`, `Date`, `Number` und `Currency` und einer dokumentierten Richtlinie für fehlende Schlüssel
 - Domänenencode, der einen `TicketStatus`-Wert zurückgibt, den die Oberfläche auf eine lokalisierte Beschriftung abbildet
 - Pseudo-lokalisierter Ressourcensatz, mit dem jede verbliebene fest verdrahtete Zeichenfolge gefunden wird
 - Vollständige Lokalisierungs-Testmatrix für Rückfall, Formatierung, Laufzeitwechsel und RTL
 - Veröffentlichte Ausgabe, auf ihre Kulturordner geprüft, dazu die GlobalDesk-Lokalisierungs-QA-Checkliste
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:

- Ein Reviewer öffnet GlobalDesk auf Französisch, einer Kultur, die Sie nie übersetzt haben, und sieht englische Beschriftungen mit korrekt formatierten französischen Datumsangaben. Erklären Sie, welcher Mechanismus jede Hälfte dieses Bildschirms hervorgebracht hat, und sagen Sie, ob irgendetwas kaputt ist.
- Ihr pseudo-lokalisierter Lauf zeigt ein Label, das weiterhin schlicht englisch „Save“ liest. Nennen Sie die drei Stellen, aus denen dieser Text stammen könnte, und beschreiben Sie, wie Sie sie ohne Raten auseinanderhalten würden.
- Ein Kollege schlägt vor, die aufgelöste CultureInfo in einem statischen Feld auf LocalizationService zwischenzuspeichern, „weil es verschwenderisch ist, sie bei jedem Aufruf nachzuschlagen“. Beschreiben Sie, was ein zweiter angemeldeter Benutzer erleben würde, und was Sie stattdessen vorschlagen würden.

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:

- ✓ Ein Reviewer öffnet GlobalDesk auf Französisch, einer Kultur, die Sie nie übersetzt haben, und sieht englische Beschriftungen mit korrekt formatierten französischen Datumsangaben. Erklären Sie, welcher Mechanismus jede Hälfte dieses Bildschirms hervorgebracht hat, und sagen Sie, ob irgendetwas kaputt ist.
- ✓ Ihr pseudo-lokalisierter Lauf zeigt ein Label, das weiterhin schlicht englisch „Save“ liest. Nennen Sie die drei Stellen, aus denen dieser Text stammen könnte, und beschreiben Sie, wie Sie sie ohne Raten auseinanderhalten würden.
- ✓ Ein Kollege schlägt vor, die aufgelöste CultureInfo in einem statischen Feld auf LocalizationService zwischenzuspeichern, „weil es verschwenderisch ist, sie bei jedem Aufruf nachzuschlagen“. Beschreiben Sie, was ein zweiter angemeldeter Benutzer erleben würde, und was Sie stattdessen vorschlagen würden.

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.

- ✓ **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-07.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."*
- ✓ **Tauschen Sie ein Teil aus, ohne die Schnittstellen zu verschieben.** *"Wähle den Teil dieses GlobalDesk-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."*



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.