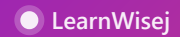


KI-Coding-Übung

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.



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 GlobalDesk-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

Erweitern Sie die gemeinsamen Ressourcen von GlobalDesk. Legen Sie Strings.de.resx neben Resources/Strings.resx an, mit exakt denselben Schlüsseln und übersetzten Werten, fügen Sie einen Eintrag Customer.LastOrder mit zwei Platzhaltern hinzu und bauen Sie den Satz mit string.Format und Application.CurrentCulture auf, statt Fragmente aneinanderzuhängen. Leiten Sie jede Meldung, die der Kundeneditor anzeigt, über den Texts-Helper, einschließlich der Speicherbestätigung und der beiden Validierungsmeldungen, und lassen Sie den Save-Service einen stabilen Ergebnisschlüssel wie CustomerEditor.Saved zurückgeben statt eines englischen Satzes, den die Oberfläche direkt ausgibt. Fügen Sie anschließend eine Datei Resources-de.resx hinzu, die eine Systembeschriftung von Wisej überschreibt, weisen Sie das in einer MessageBox nach und schreiben Sie eine kurze Notiz zum Unterschied zwischen Ihrem Produkttext und dem Systemtext von Wisej.

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

- ✓ Strings.de.resx mit denselben Schlüsseln wie die neutrale Datei und ausschließlich übersetzten Werten
- ✓ Ressource Customer.LastOrder, mit string.Format und der aktiven Kultur aufgebaut, niemals zusammengehängt
- ✓ Save-Service, der einen Ressourcenschlüssel oder Ergebniscode statt eines englischen Satzes zurückgibt
- ✓ Resources-de.resx, die mindestens eine Systembeschriftung von Wisej überschreibt, gezeigt in einer MessageBox
- ✓ Lab-Notiz, die Produkttext, Wisej-Systemtext und Diagnosetext voneinander trennt

Warum es so aufgebaut ist

Modul 2 hat den GlobalDesk-Kundeneditor auf Deutsch überzeugend hinterlassen: Jede Beschriftung, die der Designer besitzt, wechselt mit der Kultur, und das Layout hält stand. Dann drückt der Benutzer Save und liest Customer saved. auf Englisch, denn dieser Satz ist keine Eigenschaft irgendeines Controls — er ist ein Zeichenfolgenliteral in einem Click-Handler, gleich neben This field is required. und der Statuszeile, die die geladenen Kontakte zählt.

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
gemeinsame Anwendungsressourcendatei	Resources/Strings.resx enthält den für den Benutzer sichtbaren Text, den kein einzelner entworfener Bildschirm besitzt: Speicherbestätigungen, Validierungsmeldungen, Statuszeilen, Berichtsüberschriften und Serviceergebnisse.
kulturspezifische Geschwisterdatei	Strings.de.resx und Strings.it.resx liegen neben der neutralen Datei und tragen dieselben Schlüssel; ein Übersetzer ändert Werte, niemals Schlüsselnamen.
stabiler semantischer Schlüssel	Ein Name wie CustomerEditor.Saved beschreibt die Bedeutung des Textes, sodass eine Umformulierung des englischen Satzes nie eine Codeänderung erzwingt oder die Übersetzungen zerstört.
Wisej.Resources.ResourceManager	Das Suchobjekt, das dem.NET-Ressourcenmodell folgt: Mit Basisname und Assembly findet es den Wert für die aktive Kultur und fällt in Richtung neutraler Datei zurück.
Markierung für fehlende Schlüssel	Ein Helper, der bei einer fehlgeschlagenen Suche [Key] zurückgibt, macht aus einem unsichtbaren Defekt etwas, das Sie sehen und suchen können, statt eines leeren Labels.
Formatzeichenfolge mit Platzhaltern	Der gesamte Satz mit seinen {0} und {1} lebt in der Ressource und string.Format füllt ihn, denn Sprachen ordnen Name, Datum und Verb unterschiedlich an.
Resources-[LANG].resx	Eine separate Wisej.NET-Systemressourcendatei im Projektstamm oder in einem Ordner /Resources, die Framework-Beschriftungen wie \$Ok, \$Cancel und \$Offline überschreibt oder ergänzt.

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 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 3 – „Anwendungsressourcen, ResourceManager und Wisej-Systemtexte“ – 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-03.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: Erweitern Sie die gemeinsamen Ressourcen von GlobalDesk. Legen Sie `Strings.de.resx` neben `Resources/Strings.resx` an, mit exakt denselben Schlüsseln und übersetzten Werten, fügen Sie einen Eintrag `Customer.LastOrder` mit zwei Platzhaltern hinzu und bauen Sie den Satz mit `string.Format` und `Application.CurrentCulture` auf, statt Fragmente aneinanderzuhängen. Leiten Sie jede Meldung, die der Kundeneditor anzeigt, über den Texts-Helper, einschließlich der Speicherbestätigung und der beiden Validierungsmeldungen, und lassen Sie den Save-Service einen stabilen Ergebnisschlüssel wie `CustomerEditor.Saved` zurückgeben statt eines englischen Satzes, den die Oberfläche direkt ausgibt. Fügen Sie anschließend eine Datei `Resources-de.resx` hinzu, die eine Systembeschriftung von Wisej überschreibt, weisen Sie das in einer `MessageBox` nach und schreiben Sie eine kurze Notiz zum Unterschied zwischen Ihrem Produkttext und dem Systemtext von Wisej.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - `Strings.de.resx` mit denselben Schlüsseln wie die neutrale Datei und ausschließlich übersetzten Werten
 - Ressource `Customer.LastOrder`, mit `string.Format` und der aktiven Kultur aufgebaut, niemals zusammengehängt
 - Save-Service, der einen Ressourcenschlüssel oder Ergebniscode statt eines englischen Satzes zurückgibt
 - `Resources-de.resx`, die mindestens eine Systembeschriftung von Wisej überschreibt, gezeigt in einer `MessageBox`
 - Lab-Notiz, die Produkttext, Wisej-Systemtext und Diagnosetext voneinander trennt
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:

- Löschen Sie einen Schlüssel aus `Strings.de.resx` und führen Sie die Anwendung auf Deutsch

aus. Beschreiben Sie, was der Benutzer sieht, warum nichts abstürzt und wie Sie das in einer echten Anwendung mit zweihundert Schlüsseln bemerken würden.

- Ihr Save-Service gibt derzeit CustomerEditor.Saved zurück. Was ginge in den Unit-Tests und auf dem deutschen Bildschirm kaputt, wenn er stattdessen den Satz selbst zurückgäbe?
- Ein Kollege fügt das Wort „Cancel“ unter einem selbst erfundenen Schlüssel in Resources-de.resx ein. Erklären Sie, warum die Schaltfläche der MessageBox sich nicht ändert und wohin diese Zeichenfolge tatsächlich gehört hätte.

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:

- ✓ Löschen Sie einen Schlüssel aus Strings.de.resx und führen Sie die Anwendung auf Deutsch aus. Beschreiben Sie, was der Benutzer sieht, warum nichts abstürzt und wie Sie das in einer echten Anwendung mit zweihundert Schlüsseln bemerken würden.
- ✓ Ihr Save-Service gibt derzeit CustomerEditor.Saved zurück. Was ginge in den Unit-Tests und auf dem deutschen Bildschirm kaputt, wenn er stattdessen den Satz selbst zurückgäbe?
- ✓ Ein Kollege fügt das Wort „Cancel“ unter einem selbst erfundenen Schlüssel in Resources-de.resx ein. Erklären Sie, warum die Schaltfläche der MessageBox sich nicht ändert und wohin diese Zeichenfolge tatsächlich gehört hätte.

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.

- ✓ **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."
- ✓ **Machen Sie es fit für einen schlechten Tag.** "Untersuche dieses GlobalDesk-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.