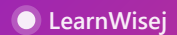


KI-Coding-Übung

Erstellen Sie das Beispielprojekt IconDesk 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 IconDesk.Icons als Klassenbibliothek neben der Anwendung, legen Sie mindestens zwölf SVG-Icons in ein Verzeichnis Resources und markieren Sie sie als Embedded Resource. Fügen Sie einen statischen Katalog Applcons hinzu, der jedes Icon als benannten resource.wx-String bereitstellt, sodass niemand mehr einen Pfad tippt, referenzieren Sie die Bibliothek aus IconDesk und weisen Sie dann einige Icons über den Designer und andere in C# aus dem Katalog zu. Prüfen Sie das Farbsuffix an drei Icons und notieren Sie, welche SVG-Bereinigung sie dafür gebraucht haben. Schließen Sie mit den Regeln für Benennung und Versionierung ab, die bestimmen, wie ein neues Icon ins Pack kommt.

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

- ✓ Klassenbibliothek IconDesk.Icons mit mindestens zwölf SVG-Dateien in einem Verzeichnis Resources, alle als Embedded Resource
- ✓ Statischer Katalog Applcons, der jedes Icon als benannten resource.wx-String bereitstellt
- ✓ IconDesk-Seite, die einige Icons über den Designer und andere aus dem Katalog Applcons zuweist
- ✓ Drei mit dem Farbsuffix geprüfte Icons samt Notiz zur nötigen SVG-Bereinigung
- ✓ Regeln für Benennung und Versionierung, die beschreiben, wie ein neues Icon ins Pack kommt und wie eine Anwendung eines überschreibt

Warum es so aufgebaut ist

Beim fünften Modul zeigt IconDesk ein eingebettetes Logo, ein Theme-Bild und Icons aus zwei offiziellen Packs – und genau da beginnt das Durcheinander: Ressourcen-Strings stehen direkt in Event-Handletern, zwei Bildschirme schreiben dieselbe Datei unterschiedlich, und niemand kann sagen, welche Icons dem Produkt gehören. Ein eigenes Pack ist die günstige Lösung, und es bringt eine Grenze: Eine Assembly hält die Grafiken, ein C#-Katalog benennt sie, und der Rest der Anwendung tippt nie wieder einen Pfad.

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
Ressourcen-Assembly	Eine schlichte.NET-Klassenbibliothek, deren einzige Aufgabe es ist, Icon-Dateien als eingebettete Ressourcen zu tragen; sie braucht keine Referenz auf die Anwendung und keinen eigenen Wisej.NET-Code.
Das vierstufige Rezept	Eine Klassenbibliothek erstellen, ein Verzeichnis namens Resources anlegen, die Icon-Dateien hineinlegen, ihre Build Action auf Embedded Resource setzen. Alles danach ist Komfort.
Icon-Katalog	Eine statische Klasse wie Applcons, die jedes Icon als benannte String-Konstante bereitstellt, sodass Compiler und IntelliSense kopierte Pfade ersetzen.
Semantischer Name gegenüber Dateiname	Der Katalog zeigt das Vokabular Ihres Produkts – save, delete, customer, warning –, während die Datei einen einfachen Namen behält; sie umzubenennen ändert also eine Zeile und nicht vierzig Bildschirme.
Qualifizierte Ressourcenadresse	Ein Icon in einer eigenen Assembly wird als resource.wx/ plus Assembly-Name plus Ressourcenpfad adressiert – deshalb teilen sich alle Katalogeinträge eine private Wurzelkonstante.
Vertrag beim Umfärben	Das Suffix?color= färbt nur geeignete einfarbige Grafiken um; ein SVG mit fest verdrahteten fill -Werten ist der übliche Grund, warum ein Icon seine Farbe nicht ändert.
Theme-Sätze und Übersreibungen	Parallele Pakete mit derselben Katalog-API über anderen Grafiken sowie ausgelieferte Dateien, die Vorrang vor einer eingebetteten Ressource haben, ersetzen Icons, ohne Anwendungscode anzufassen.

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 6 geht. Sie können [module-06.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 IconDesk 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 6 – „Ein eigenes Wisej.NET-Icon-Pack bauen“ – des LearnWisej-Kurses „Icons und Bilder 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/icons-and-images/specs/module-06.md>

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

1. Projekt „IconDesk“, 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 `IconDesk.Icons` als Klassenbibliothek neben der Anwendung, legen Sie mindestens zwölf SVG-Icons in ein Verzeichnis `Resources` und markieren Sie sie als `Embedded Resource`. Fügen Sie einen statischen Katalog `AppIcons` hinzu, der jedes Icon als benannten `resource.wx-String` bereitstellt, sodass niemand mehr einen Pfad tippt, referenzieren Sie die Bibliothek aus `IconDesk` und weisen Sie dann einige Icons über den Designer und andere in C# aus dem Katalog zu. Prüfen Sie das Farbsuffix an drei Icons und notieren Sie, welche SVG-Bereinigung sie dafür gebraucht haben. Schließen Sie mit den Regeln für Benennung und Versionierung ab, die bestimmen, wie ein neues Icon ins Pack kommt.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Klassenbibliothek `IconDesk.Icons` mit mindestens zwölf SVG-Dateien in einem Verzeichnis `Resources`, alle als `Embedded Resource`
 - Statischer Katalog `AppIcons`, der jedes Icon als benannten `resource.wx-String` bereitstellt
 - `IconDesk`-Seite, die einige Icons über den Designer und andere aus dem Katalog `AppIcons` zuweist
 - Drei mit dem Farbsuffix geprüfte Icons samt Notiz zur nötigen SVG-Bereinigung
 - Regeln für Benennung und Versionierung, die beschreiben, wie ein neues Icon ins Pack kommt und wie eine Anwendung eines überschreibt
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:

- Ihr Pack baut, und der Designer zeigt im Image Selector nichts an. Welchen Schritt des Rezepts würden Sie zuerst prüfen, und welcher Beleg würde Ihnen sagen, dass die Ressourcen wirklich in der Assembly liegen?
- Zwei Ihrer drei Test-Icons lassen sich umfärben, das dritte bleibt grau. Was steckt in

diesem SVG, und wann würden Sie ein Icon absichtlich nicht umfärbbar lassen?

– Ein Kunde möchte, dass nur in seinem Deployment drei Icons ersetzt werden. Welchen der Überschreibungswege würden Sie wählen, und was kostet Sie diese Wahl bei der nächsten Pack-Version?

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:

- ✓ Ihr Pack baut, und der Designer zeigt im Image Selector nichts an. Welchen Schritt des Rezepts würden Sie zuerst prüfen, und welcher Beleg würde Ihnen sagen, dass die Ressourcen wirklich in der Assembly liegen?
- ✓ Zwei Ihrer drei Test-Icons lassen sich umfärben, das dritte bleibt grau. Was steckt in diesem SVG, und wann würden Sie ein Icon absichtlich nicht umfärbbar lassen?
- ✓ Ein Kunde möchte, dass nur in seinem Deployment drei Icons ersetzt werden. Welchen der Überschreibungswege würden Sie wählen, und was kostet Sie diese Wahl bei der nächsten Pack-Version?

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](https://learnwisej.com/assets/courses/icons-and-images/specs/module-06.md) 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/icons-and-images/specs/module-06.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 IconDesk-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 IconDesk-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.