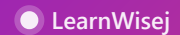


# KI-Coding-Übung

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.



## Dasselbe Modul – so gebaut, wie Sie in der Praxis wirklich bauen.

Alle anderen Abschnitte dieses Moduls erklären, wie „Eingebettete Ressourcen und resource.wx“ funktioniert. Dieser fordert Sie auf, es herzustellen: indem Sie einen Assistenten mit der Spezifikation des Moduls steuern und dann prüfen, ob das Ergebnis läuft und tut, was die Lab-Aufgabe beschreibt. Es ist der Arbeitsablauf, den Sie in echten Projekten nutzen werden – geübt an etwas, dessen richtige Lösung Sie bereits kennen.

## Worum Sie bitten

Legen Sie in der Anwendung IconDesk einen Ordner Resources mit logo.svg, status-ok.svg, status-warning.svg, photo.png und badge.gif an und setzen Sie für jede dieser Dateien Build Action auf Embedded Resource. Bauen Sie eine Seite ResourceLab, die alle fünf über resource.wx-Bildquellen anzeigt, schreiben Sie eine dieser Quellen mit dem Assembly-Namen der Anwendung vor dem Dateinamen, damit die Auflösung eindeutig ist, und lassen Sie die übrigen unqualifiziert. Legen Sie dann eine Ersatzdatei neben die ausgelieferte Anwendung, um eine eingebettete Ressource zu überschreiben, und halten Sie den Zustand davor und danach fest. Schreiben Sie zum Abschluss eine resource.wx-URL absichtlich falsch und notieren Sie die Schritte, die den Fehler gefunden haben.

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

- ✓ Ordner Resources mit logo.svg, status-ok.svg, status-warning.svg, photo.png und badge.gif, alle als Embedded Resource markiert
- ✓ Seite ResourceLab, die jedes eingebettete Asset über eine resource.wx-Bildquelle anzeigt
- ✓ Eine Bildquelle, die mit dem Assembly-Namen der Anwendung qualifiziert ist, während die übrigen unqualifiziert bleiben
- ✓ Ausgelieferte Datei, die eine eingebettete Ressource überschreibt, dokumentiert davor und danach
- ✓ Fehlersuche-Notiz mit den Schritten, die eine falsch geschriebene Ressourcen-URL identifiziert haben

## Warum es so aufgebaut ist

Bis hierher war jedes Asset von IconDesk etwas, das die Anwendung auf der Festplatte gefunden oder beim Theme angefragt hat. Das trägt, bis jemand die Site ausliefert und einen Ordner vergisst: Der Browser

fordert ein Icon an, der Server antwortet mit 404, und die Benutzerin sieht ein leeres Rechteck, wo das Status-Badge stehen sollte. Im Review fällt es niemandem auf, weil der Entwicklungsrechner die Datei noch hat.

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                                                                                                                                                                                                        |
|-------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Build Action = Embedded Resource          | Die Dateieigenschaft in Visual Studio, die eine Datei in das Manifest der Assembly kompiliert, statt sie neben die ausführbare Datei zu kopieren – das Asset wird also innerhalb der DLL ausgeliefert.                      |
| resource.wx-Handler                       | Der HTTP-Handler von Wisej.NET, der eingebettete Ressourcen an den Browser ausliefert; zugewiesen wird ein Bildquellen-String, nie ein serverseitiges Bildobjekt.                                                           |
| Unqualifizierte Ressourcen-URL            | resource.wx/company-logo.svg bittet Wisej.NET, die geladenen Assemblys nach einer Ressource dieses Namens zu durchsuchen – bequem und ein wenig mehrdeutig.                                                                 |
| Mit Assembly qualifizierte Ressourcen-URL | resource.wx/MyCompany.Branding/company-logo.svg benennt die Assembly (ohne.dll), sodass die Auflösung nicht bei der gleichnamigen Datei eines anderen landen kann.                                                          |
| Erlaubte Ressourcentypen                  | Der Handler dokumentiert genau neun Erweiterungen:.css,.js,.html,.jpg,.jpeg,.png,.gif,.svg und.bmp; alles andere kommt als 404 zurück.                                                                                      |
| Überschreiben durch ausgelieferte Datei   | Eine mit der Anwendung ausgelieferte Datei kann unter demselben Pfad anstelle der eingebetteten Ressource ausgeliefert werden – so kann eine Installation Grafiken umbranden oder hotfixen, ohne die Assembly neu zu bauen. |
| Manifest-Ressourcenname                   | Der Name, den.NET tatsächlich speichert, abgeleitet aus Assembly-Name, Standardnamespace und dem Ordner der Datei; ihn zu raten ist der übliche Grund, warum eine Ressourcen-URL nichts liefert.                            |

## 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 5 geht. Sie können [module-05.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



### Claude verwenden

Öffnet [claude.ai](#) mit demselben Prompt. Wenn Sie Claude Code nutzen, können Sie

Codex; jeder Tarif mit Websuche ruft die Spezifikation ab.

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 5 – „Eingebettete Ressourcen und resource.wx“ – 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-05.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: Legen Sie in der Anwendung IconDesk einen Ordner Resources mit `logo.svg`, `status-ok.svg`, `status-warning.svg`, `photo.png` und `badge.gif` an und setzen Sie für jede dieser Dateien Build Action auf Embedded Resource. Bauen Sie eine Seite ResourceLab, die alle fünf über resource.wx-Bildquellen anzeigt, schreiben Sie eine dieser Quellen mit dem Assembly-Namen der Anwendung vor dem Dateinamen, damit die Auflösung eindeutig ist, und lassen Sie die übrigen unqualifiziert. Legen Sie dann eine Ersatzdatei neben die ausgelieferte Anwendung, um eine eingebettete Ressource zu überschreiben, und halten Sie den Zustand davor und danach fest. Schreiben Sie zum Abschluss eine resource.wx-URL absichtlich falsch und notieren Sie die Schritte, die den Fehler gefunden haben.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
  - Ordner Resources mit `logo.svg`, `status-ok.svg`, `status-warning.svg`, `photo.png` und `badge.gif`, alle als Embedded Resource markiert
  - Seite ResourceLab, die jedes eingebettete Asset über eine resource.wx-Bildquelle anzeigt
  - Eine Bildquelle, die mit dem Assembly-Namen der Anwendung qualifiziert ist, während die übrigen unqualifiziert bleiben
  - Ausgelieferte Datei, die eine eingebettete Ressource überschreibt, dokumentiert davor und danach
  - Fehlersuche-Notiz mit den Schritten, die eine falsch geschriebene Ressourcen-URL identifiziert haben
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:

- Zwei referenzierte Assemblys betten beide eine Datei namens `logo.svg` ein. Worauf löst eine unqualifizierte Quelle auf, und was würden Sie ändern, damit die Antwort nicht mehr von der

Ladereihenfolge abhängt?

- Ihre Überschreibung funktioniert auf dem Testserver, aber nach dem nächsten Release nicht mehr. Welche Teile des Deployment-Prozesses könnten sie entfernt haben, und wo hätte die Überschreibung dokumentiert werden müssen?
- Ein Teammitglied bettet icon.ico ein und meldet, resource.wx sei kaputt. Was passiert tatsächlich, und was ist die kleinste Änderung, die das behebt?

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:

- ✓ Zwei referenzierte Assemblys betten beide eine Datei namens logo.svg ein. Worauf löst eine unqualifizierte Quelle auf, und was würden Sie ändern, damit die Antwort nicht mehr von der Ladereihenfolge abhängt?
- ✓ Ihre Überschreibung funktioniert auf dem Testserver, aber nach dem nächsten Release nicht mehr. Welche Teile des Deployment-Prozesses könnten sie entfernt haben, und wo hätte die Überschreibung dokumentiert werden müssen?
- ✓ Ein Teammitglied bettet icon.ico ein und meldet, resource.wx sei kaputt. Was passiert tatsächlich, und was ist die kleinste Änderung, die das behebt?

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 IconDesk-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/icons-and-images/specs/module-05.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."*



### **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.