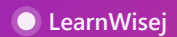


KI-Coding-Übung

Erstellen Sie das Beispielprojekt OperationsConsole 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, auch im Vergleich mit unserem eigenen Referenz-Build auf GitHub.



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

Kapseln Sie ein kleines JavaScript-Bewertungs- oder Anzeige-Widget mit dem Widget-Control oder konfigurieren Sie ein Wisej.NET-Erweiterungspaket: Laden Sie seine JavaScript- und CSS-Pakete, lösen Sie mit dem Widget-Event-Muster vom Client aus ein serverseitiges Event aus, rufen Sie mit CallAsync oder Eval vom Server aus eine clientseitige Funktion auf, passen Sie das Theme an die Konsole an, feilen Sie dann am vollständigen Abschlussprojekt Operations Console und bereiten Sie eine 5-minütige Demo vor.

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

- ✓ Bildschirm auf Basis eines Widgets oder einer Erweiterung mit geladenen JavaScript- und CSS-Paketen
- ✓ Event vom Client zum Server über das Widget-Event-Muster oder ein natives Erweiterungs-Event
- ✓ Aufruf vom Server zum Client über CallAsync, Eval oder die API der Erweiterung
- ✓ Dokumentierter Client/Server-Vertrag: Pakete, Properties, Events und Payload-Format
- ✓ Ausgefeiltes Abschlussprojekt Operations Console mit einem Skript für eine 5-minütige Demo

Warum es so aufgebaut ist

Früher oder später braucht ein Produkt etwas, das die native Bibliothek nicht mitbringt: eine spezialisierte Browserfunktion, eine JavaScript-Bibliothek eines Drittanbieters, eine eigene Visualisierung oder eine wiederverwendbare Integration. Die Versuchung ist groß, in JavaScript abzutauchen und dabei das Wisej.NET-Modell zu verlieren: Serverzustand, Theming, den Designer und einen Event-Fluss, dem ein anderer Entwickler folgen kann. Dieses Modul integriert ein Client-Widget, ohne auf irgendetwas davon zu verzichten, und schließt dann das Abschlussprojekt Operations Console ab.

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
natives Control	Ein Control, das sowohl die Serverseite als auch sein Browser-Widget implementiert und daher bereits Designer-Unterstützung, Theming, Events und Serverzustand mitbringt; immer die erste Anlaufstelle.
Erweiterungsarten	Control, Widget, Component, Extender Provider und Icon Pack: Wer die Art benennt, verwechselt „wiederverwendbar“ nicht mehr mit „neues natives Control“.
Widget-Control	Ein universeller Client-Container, der JavaScript- und CSS-Pakete lädt, ein Initialisierungsskript ausführt und Events und Daten mit dem Servercode austauscht.
Packages und InitScript	Die Paketliste deklariert die Client-Dateien, von denen das Widget abhängt; das InitScript erzeugt das JavaScript-Objekt, verknüpft seine Client-Events und stellt nur die Funktionen bereit, die der Server braucht.
WidgetEvent	Das Server-Event, das ausgelöst wird, wenn Client-Code fireWidgetEvent aufruft; Typ und Daten-Payload kommen aus dem Browser und müssen wie jede andere Benutzereingabe validiert werden.
CallAsync / Eval	Die Richtung vom Server zum Client dieser Verbindung: eine vom Widget bereitgestellte Funktion aufrufen oder ein Skript auswerten, ohne C# zu verlassen.
Widget-Vertrag	Die dokumentierten Properties, Events, das Payload-Format, die Paketabhängigkeiten und Theming-Regeln, mit denen ein anderer Entwickler oder ein fundierter KI-Assistent das Widget pflegen kann.

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 OperationsConsole 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 – „Eigene Widgets, Erweiterungen, Theming und Abschlussprojekt“ – des LearnWisej-Kurses „Die Control-Bibliothek meistern“ 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/mastering-the-control-library/specs/module-07.md>

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

1. Projekt „OperationsConsole“, 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: Kapseln Sie ein kleines JavaScript-Bewertungs- oder Anzeige-Widget mit dem Widget-Control oder konfigurieren Sie ein Wisej.NET-Erweiterungspaket: Laden Sie seine JavaScript- und CSS-Pakete, lösen Sie mit dem Widget-Event-Muster vom Client aus ein serverseitiges Event aus, rufen Sie mit `CallAsync` oder `Eval` vom Server aus eine clientseitige Funktion auf, passen Sie das Theme an die Konsole an, feilen Sie dann am vollständigen Abschlussprojekt Operations Console und bereiten Sie eine 5-minütige Demo vor.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Bildschirm auf Basis eines Widgets oder einer Erweiterung mit geladenen JavaScript- und CSS-Paketen
 - Event vom Client zum Server über das Widget-Event-Muster oder ein natives Erweiterungs-Event
 - Aufruf vom Server zum Client über `CallAsync`, `Eval` oder die API der Erweiterung
 - Dokumentierter Client/Server-Vertrag: Pakete, Properties, Events und Payload-Format
 - Ausgefeiltes Abschlussprojekt Operations Console mit einem Skript für eine 5-minütige Demo
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:

- Was würde passieren, wenn ein Benutzer die `ratingChanged` -Payload in den Entwicklerwerkzeugen des Browsers verändert, bevor sie Ihren Handler erreicht, und welche Zeile Ihres Codes hält das auf?
- Ihr Widget erscheint leer, nachdem der Tab Widgets aus- und wieder eingeblendet wurde: Welche drei Dinge prüfen Sie in welcher Reihenfolge, und warum ist das Start-HTML nicht die Lösung?
- Welche Regel Ihres Widgets könnten Sie als Optimierung nach JavaScript verlagern, und

welche darf den Server niemals verlassen?

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:

- ✓ Was würde passieren, wenn ein Benutzer die `ratingChanged` -Payload in den Entwicklerwerkzeugen des Browsers verändert, bevor sie Ihren Handler erreicht, und welche Zeile Ihres Codes hält das auf?
- ✓ Ihr Widget erscheint leer, nachdem der Tab Widgets aus- und wieder eingeblendet wurde: Welche drei Dinge prüfen Sie in welcher Reihenfolge, und warum ist das Start-HTML nicht die Lösung?
- ✓ Welche Regel Ihres Widgets könnten Sie als Optimierung nach JavaScript verlagern, und welche darf den Server niemals verlassen?

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.



Dieses Beispiel haben wir schon gebaut – vergleichen Sie Ihres mit unserem

Die Referenzimplementierung zu Modul 7 liegt im öffentlichen Beispiel-Repository. Bauen Sie zuerst Ihre eigene Version und öffnen Sie dann unsere: Die Unterschiede sind der interessante Teil.



[Beispiel zu Modul 7 auf GitHub öffnen](#)

Oder klonen Sie alle Beispiele mit `git clone https://github.com/iceteagroup/LearnWisej-Samples` und öffnen Sie `Mastering the Control Library Course/Module 7`.



Unser Beispiel ist ein Ausgangspunkt, kein fertiges Produkt

Es ist genauso entstanden wie Ihres – per Prompt und anschließend korrigiert – und endet bewusst dort, wo das Modul endet. Rechnen Sie mit Platzhaltern, wo Produktivcode hingehört, mit knapper Fehlerbehandlung und ohne Tests. Verstehen Sie das als Einladung: Nutzen Sie, was Sie in diesem Modul gelernt haben, und dazu den Assistenten, um es weiterzuentwickeln. Code zu verbessern, den Sie nicht selbst geschrieben haben, ist die Fähigkeit, die dieser Kurs eigentlich vermittelt.

Drei Prompts für den nächsten Schritt

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

- ✓ **Ergänzen Sie den zweiten Fall.** *"Erweitere dieses OperationsConsole-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/mastering-the-control-library/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 OperationsConsole-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.