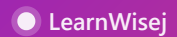


KI-Coding-Übung

Erstellen Sie das Beispielprojekt SupportDesk 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.



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

Legen Sie die Projektmappe Support Desk Data Console mit den Projekten Web, Data, Services und Tests an, fügen Sie dem Datenprojekt die EF-Core-Provider-Pakete hinzu, erstellen Sie SupportDeskContext mit einer Design-Time-Factory, registrieren Sie AddDbContextFactory im ASP.NET-Core-Startup und stellen Sie Wisej.NET den IServiceProvider von Microsoft bereit. Lösen Sie dann aus einer Wisej.NET-Page einen Abfrage-Service auf und führen Sie eine erste asynchrone Zählerabfrage hinter einem Lade-Flag und einer Fehlermeldung aus – noch ohne Binding.

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

- ✓ Projektmappe mit den Projekten Web, Data, Services und Tests und den EF-Core-Provider-Paketen
- ✓ SupportDeskContext über AddDbContextFactory registriert, mit einer Design-Time-Factory
- ✓ IServiceProvider von Microsoft für Wisej.NET bereitgestellt und ein Abfrage-Service aus einer Page aufgelöst
- ✓ Erste asynchrone Zählerabfrage mit Lade-Schutz und Fehlermeldung
- ✓ Nirgends in der Projektmappe ein statischer DbContext

Warum es so aufgebaut ist

Eine Wisej.NET-Anwendung fühlt sich wie WinForms an, weil Formulare, Pages, Controls, BindingSource - Komponenten und Event-Handler alle auf dem Server leben – und genau diese Vertrautheit verleitet zu einer gefährlichen Abkürzung: einen DbContext als ein weiteres langlebiges Serverobjekt zu behandeln. EF Core läuft zwar ebenfalls serverseitig, ist aber kein Container für UI-Zustand, sondern eine kurze Unit of Work, deren Change Tracker weder threadsicher noch zum Teilen gedacht ist. Viele Browser-Sessions

öffnen die Support Desk Data Console gleichzeitig; die erste Fähigkeit dieses Kurses besteht deshalb darin, die Lebensdauern der UI und die Lebensdauer der Datenbankzugriffe auseinanderzuhalten, bevor überhaupt ein Grid gebunden wird.

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
Session-Zustand	Der serverseitige Zustand pro Browser-Session, den Wisej.NET am Leben hält: offene Pages, die ausgewählte Zeile, Filter, Validierungsmeldungen und ausstehende Änderungen, erreichbar über Application.Session und den aktiven Control-Baum.
Lebensdauer von UI-Objekten	Wie lange eine Page, eine Form oder eine BindingSource existiert – von der Konstruktion, bis der Benutzer sie schließt oder die Session endet; oft Stunden, weit länger als jeder Datenbankaufruf.
Request-/Thread-Lebensdauer	Ein Browser-Event oder eine await -Fortsetzung auf dem Server; der Thread, der einen Click-Handler begonnen hat, ist nicht unbedingt derjenige, der ihn beendet.
Unit of Work	Die Lebensdauer eines DbContext: erzeugen, Entitäten abfragen oder ändern, SaveChangesAsync aufrufen, freigeben; Sekunden, nicht Sessions.
DbContextFactory	Der von AddDbContextFactory registrierte Service IDbContextFactory<T>, der einen frischen, korrekt konfigurierten Context liefert, sobald eine Operation einen braucht.
Anbindung an die Microsoft-DI	Die Registrierung des IServiceProvider von ASP.NET Core bei Application.Services, damit Wisej.NET-Pages dieselben Services auflösen können, die der Startup konfiguriert hat.
sessionübergreifendes Datenleck	Ein statisches oder gemeinsam genutztes veränderliches Objekt (ein Context, ein Change Tracker, eine gebundene Liste, eine ausgewählte Entität), das die Session eines Benutzers lesen kann, nachdem die Session eines anderen Benutzers hineingeschrieben hat.

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 1 geht. Sie können [module-01.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 SupportDesk 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 1 – „Architektur: EF Core in einer Wisej.NET-Anwendung“ – des LearnWisej-Kurses „Data Binding mit EF Core“ 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/ef-core-binding/specs/module-01.md>

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

1. Projekt „SupportDesk“, 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 die Projektmappe Support Desk Data Console mit den Projekten Web, Data, Services und Tests an, fügen Sie dem Datenprojekt die EF-Core-Provider-Pakete hinzu, erstellen Sie `SupportDeskContext` mit einer Design-Time-Factory, registrieren Sie `AddDbContextFactory` im ASP.NET-Core-Startup und stellen Sie Wisej.NET den `IServiceProvider` von Microsoft bereit. Lösen Sie dann aus einer Wisej.NET-Page einen Abfrage-Service auf und führen Sie eine erste asynchrone Zählabfrage hinter einem Lade-Flag und einer Fehlermeldung aus – noch ohne Binding.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - Projektmappe mit den Projekten Web, Data, Services und Tests und den EF-Core-Provider-Paketen
 - `SupportDeskContext` über `AddDbContextFactory` registriert, mit einer Design-Time-Factory
 - `IServiceProvider` von Microsoft für Wisej.NET bereitgestellt und ein Abfrage-Service aus einer Page aufgelöst
 - Erste asynchrone Zählabfrage mit Lade-Schutz und Fehlermeldung
 - Nirgends in der Projektmappe ein statischer `DbContext`
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:

- Welche Lebensdauer haben Sie für den `DbContext` hinter `CountTicketsAsync` gewählt, und was würde schiefgehen, wenn die Page diesen Context zwischen den Klicks am Leben hielte?
- Zwei Benutzer öffnen gleichzeitig die Support-Desk-Konsole und klicken beide auf Count. Welche Objekte sind pro Session getrennt, welche werden geteilt, und warum ist diese Aufteilung sicher?
- Was sieht der Benutzer, wenn die Datenbank nicht erreichbar ist, was macht der Button, und wie wird das Lade-Flag wieder false?

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:

- ✓ Welche Lebensdauer haben Sie für den DbContext hinter CountTicketsAsync gewählt, und was würde schiefgehen, wenn die Page diesen Context zwischen den Klicks am Leben hielte?
- ✓ Zwei Benutzer öffnen gleichzeitig die Support-Desk-Konsole und klicken beide auf Count. Welche Objekte sind pro Session getrennt, welche werden geteilt, und warum ist diese Aufteilung sicher?
- ✓ Was sieht der Benutzer, wenn die Datenbank nicht erreichbar ist, was macht der Button, und wie wird das Lade-Flag wieder false?

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 1 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 1 auf GitHub öffnen](#)

Oder klonen Sie alle Beispiele mit `git clone https://github.com/iceteagroup/LearnWisej-Samples` und öffnen Sie [Data Binding with EF Core Course/Module 1](#).



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 SupportDesk-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/ef-core-binding/specs/module-01.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 SupportDesk-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.