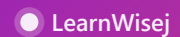


KI-Coding-Übung

Erstellen Sie das Beispielprojekt VisualOperationsStudio 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

Ergänzen Sie VisualOperationsStudio um einen TopologyImageRenderer, der das Topologiemodell in ein PNG verwandelt, ohne ein einziges Wisej-Control anzufassen. Geben Sie ihm eine Render-Methode, die Szene und Ausgabegröße entgegennimmt, die Bitmap erzeugt, deren Graphics über Graphics.FromImage bezieht, Knoten, Kanten, Beschriftungen und eine Legende mit Pens, Brushes und einem GraphicsPath zeichnet und alles freigibt, was ihm gehört, während die Objekte des Aufrufers unangetastet bleiben. Lösen Sie seinen Font über eine Methode ResolveFont auf, die eine von der Anwendung mitgelieferte Schriftfamilie versucht und auf eine garantiert vorhandene zurückfällt, statt eine Systemschrift vorauszusetzen; speichern Sie als PNG in einen MemoryStream und übergeben Sie die Bytes zum Download an die Seite. Prüfen Sie anschließend die Projektverweise, um sicherzustellen, dass Sie System.Drawing.Common nicht neben der Managed-Implementierung von Wisej hinzufügen, führen Sie den Renderer unter Windows und in einem Linux-Container aus oder bereiten Sie das vor, und beschreiben Sie, wie sich derselbe Code auf iOS-, Android- und macOS-Hybrid-Zielen verhält.

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

- ✓ TopologyImageRenderer ohne Abhängigkeit von einem Wisej-Control, der über Graphics.FromImage in eine Bitmap rendert
- ✓ Methode ResolveFont mit einer expliziten Fallback-Kette statt einer vorausgesetzten Systemschrift
- ✓ PNG in einen MemoryStream geschrieben und von der Seite heruntergeladen, wobei jedes eigene Grafikobjekt freigegeben wird
- ✓ Verweisprüfung, die bestätigt, dass Managed.System.Drawing verwendet und System.Drawing.Common nicht danebengestellt wird
- ✓ CrossPlatform.md mit Belegen von Windows und aus dem Linux-Container sowie den Folgen für iOS, Android und macOS

Warum es so aufgebaut ist

Alles, was VisualOperationsStudio bisher gezeichnet hat, gehört zu einer lebenden Sitzung: Das Gauge zeichnet sich neu, weil ein Control darum gebeten hat, und die Canvas-Topologie aus Modul 5 existiert nur als Befehle, die der Browser bereits ausgeführt hat. Der Betrieb möchte dieselbe Topologie nun in einer nächtlichen Störungs-E-Mail und als Anhang an einem Ticket – und das bedeutet ein Bild ohne Control dahinter. Gleichzeitig zieht die Anwendung in einen Linux-Container um, der Zeichencode muss also auf einem Host weiterlaufen, auf dem es weder GDI+ noch libgdiplus gibt.

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
System.Drawing.Common	Microsofts GDI+-Hülle, ab.NET 6 Windows-spezifisch; der Unix-Schalter fiel mit.NET 7 weg, also ist sie für Linux, macOS, iOS oder Android keine Option.
Managed.System.Drawing	Die NuGet-Paket-ID, umgedreht, weil der Paketstamm System.* reserviert ist; Wisej.NET 4 referenziert sie bereits für Sie.
System.Drawing.Managed.dll	Die Assembly, die dieses Paket ausliefert — die verwaltete Implementierung selbst, die Ihnen eher im Buildprotokoll als in Ihrem Quellcode begegnet.
Der Namespace System.Drawing	Unverändert: Bitmap, Graphics, Pen und Font behalten ihre Namen, sodass vorhandener Zeichencode ohne Umschreiben kompiliert.
Verwaltete Grafik-Engine	Eine rein verwaltete Implementierung auf einer angepassten, integrierten ImageSharp-Engine; im modernen.NET vermeidet sie GDI+, libgdiplus und native Aufrufe.
Graphics.FromImage	Der Einstiegspunkt fürs Off-Screen-Rendering: eine Zeichenoberfläche über einer Bitmap, die Ihnen gehört – ohne Control, ohne Paint-Event, ohne Sitzung.
PrivateFontCollection	Lädt Schriftdateien, die die Anwendung mitbringt, sodass die Ausgabe davon abhängt, was Sie ausgeliefert haben, und nicht davon, was zufällig auf dem Host installiert ist.

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



Claude verwenden

Öffnet chatgpt.com mit dem Prompt dieses Moduls. Funktioniert mit GPT-5 oder Codex; jeder Tarif mit Websuche ruft die Spezifikation ab.

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 VisualOperationsStudio 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 – „System.Drawing.Managed unter Windows, Linux, macOS, iOS und Android“ – des LearnWisej-Kurses „Zeichnen und Grafik 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/drawing-and-painting/specs/module-06.md>

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

1. Projekt „VisualOperationsStudio“, 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: Ergänzen Sie `VisualOperationsStudio` um einen `TopologyImageRenderer`, der das Topologiemodell in ein PNG verwandelt, ohne ein einziges Wisej-Control anzufassen. Geben Sie ihm eine `Render`-Methode, die Szene und Ausgabegröße entgegennimmt, die Bitmap erzeugt, deren `Graphics` über `Graphics.FromImage` bezieht, Knoten, Kanten, Beschriftungen und eine Legende mit Pens, Brushes und einem `GraphicsPath` zeichnet und alles freigibt, was ihm gehört, während die Objekte des Aufrufers unangetastet bleiben. Lösen Sie seinen Font über eine Methode `ResolveFont` auf, die eine von der Anwendung mitgelieferte Schriftfamilie versucht und auf eine garantiert vorhandene zurückfällt, statt eine Systemschrift vorauszusetzen; speichern Sie als PNG in einen `MemoryStream` und übergeben Sie die Bytes zum Download an die Seite. Prüfen Sie anschließend die Projektverweise, um sicherzustellen, dass Sie `System.Drawing.Common` nicht neben der Managed-Implementierung von Wisej hinzufügen, führen Sie den Renderer unter Windows und in einem Linux-Container aus oder bereiten Sie das vor, und beschreiben Sie, wie sich derselbe Code auf iOS-, Android- und macOS-Hybrid-Zielen verhält.
3. Liefere diese Ergebnisse und sag mir, wo sich jedes davon in der Projektmappe befindet:
 - `TopologyImageRenderer` ohne Abhängigkeit von einem Wisej-Control, der über `Graphics.FromImage` in eine Bitmap rendert
 - Methode `ResolveFont` mit einer expliziten Fallback-Kette statt einer vorausgesetzten Systemschrift
 - PNG in einen `MemoryStream` geschrieben und von der Seite heruntergeladen, wobei jedes eigene Grafikobjekt freigegeben wird
 - Verweisprüfung, die bestätigt, dass `Managed.System.Drawing` verwendet und `System.Drawing.Common` nicht danebengestellt wird
 - `CrossPlatform.md` mit Belegen von Windows und aus dem Linux-Container sowie den Folgen für iOS, Android und macOS
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:

- Ein Kollege sagt, der Renderer beweise die Plattformunabhängigkeit der Anwendung, weil der Code unter Linux kompiliert. Was müssten Sie im Container tatsächlich beobachten, bevor Sie diese Behauptung akzeptieren, und welcher Teil der Ausgabe weicht am wahrscheinlichsten ab?
- Ihr Renderer gibt die selbst erzeugte Bitmap frei. Warum wäre es ein Fehler, das Graphics im Paint-Handler eines Controls freizugeben, und welche Regel deckt beide Fälle in einem Satz ab?
- Jemand schlägt vor, diesen Renderer durch SkiaSharp zu ersetzen, weil es im Benchmark schneller ist. Was müssten Sie über die Aufgabe wissen, bevor Sie zustimmen, und was verliert das Team, wenn es die System.Drawing-API hinter sich lässt?

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:

- ✓ Ein Kollege sagt, der Renderer beweise die Plattformunabhängigkeit der Anwendung, weil der Code unter Linux kompiliert. Was müssten Sie im Container tatsächlich beobachten, bevor Sie diese Behauptung akzeptieren, und welcher Teil der Ausgabe weicht am wahrscheinlichsten ab?
- ✓ Ihr Renderer gibt die selbst erzeugte Bitmap frei. Warum wäre es ein Fehler, das Graphics im Paint-Handler eines Controls freizugeben, und welche Regel deckt beide Fälle in einem Satz ab?
- ✓ Jemand schlägt vor, diesen Renderer durch SkiaSharp zu ersetzen, weil es im Benchmark schneller ist. Was müssten Sie über die Aufgabe wissen, bevor Sie zustimmen, und was verliert das Team, wenn es die System.Drawing-API hinter sich lässt?

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 VisualOperationsStudio-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/drawing-and-painting/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 VisualOperationsStudio-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.