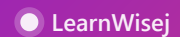


Exercice de code avec l'IA

Construisez l'exemple VisualOperationsStudio de ce module avec ChatGPT ou Claude à partir d'un prompt prêt à l'emploi qui oriente le modèle vers la spécification du module, puis relisez, exécutez et étendez ce qu'il vous renvoie.



Briefez-le, lisez-le, puis cassez-le.

Cet exercice comporte trois parties, et seule la première concerne le prompt. Vous briefez un assistant avec la spécification du module et récupérez une solution VisualOperationsStudio ; vous lisez ce qu'il a écrit et décidez s'il fait vraiment ce que le module demandait ; puis vous le poussez plus loin que ce que le module demandait. C'est dans la deuxième et la troisième partie que l'on apprend.

Ce que vous demandez

Donnez à VisualOperationsStudio une grille d'exploitation avec deux colonnes dessinées par votre code. Ajoutez des colonnes explicites à gridOperations, mettez UserPaint = true sur les seules colonnes Health et Trend, et gérez CellPaint une seule fois : sortez immédiatement lorsque e.RowIndex est négatif ou que e.ColumnIndex ne désigne aucune des deux colonnes visées, puis dessinez une barre de progression colorée selon les seuils pour Health et une sparkline en polyligne pour Trend à l'intérieur de e.ClipRectangle. Lisez les valeurs de chaque ligne depuis l'objet MachineStatus lié plutôt qu'en parcourant une liste dans le gestionnaire, respectez les couleurs de la ligne sélectionnée, et gardez le gestionnaire exempt d'effets de bord. Liez 1 000 lignes générées, faites défiler, puis construisez le même indicateur d'état une seconde fois sous forme de cellule AllowHtml afin de pouvoir écrire noir sur blanc lequel vous livreriez.

Le module demande ces livrables, et le prompt ci-dessous demande exactement ceux-là à l'assistant :

- ✓ gridOperations avec des colonnes explicites et UserPaint = true sur les seules colonnes Health et Trend
- ✓ Un unique gestionnaire CellPaint qui contrôle e.RowIndex et e.ColumnIndex avant de dessiner quoi que ce soit
- ✓ Cellule de progression colorée selon les seuils et cellule sparkline en polyligne dessinées à l'intérieur de e.ClipRectangle
- ✓ Valeurs de ligne lues depuis l'objet MachineStatus lié, sans recherche, allocation ni création de contrôle par cellule
- ✓ Une colonne d'état AllowHtml construite en parallèle, plus une note comparant les deux sur un défilement de 1 000 lignes

Pourquoi il est conçu ainsi

Le Module 2 a doté VisualOperationsStudio d'un TelemetryGauge: un contrôle, une machine, quelques centaines de redessins par heure. La grille d'exploitation est d'une autre nature. Un chef de poste ouvre une liste de mille machines et veut voir d'un coup d'œil lesquelles approchent de leur limite et lesquelles dérivent depuis le matin. Des nombres dans une colonne n'y suffiront pas, et une jauge par ligne ferait un millier de contrôles côté serveur dans une seule session. C'est la grille elle-même qui doit dessiner, une fois par cellule visible, et s'arrêter dès que la cellule sort de l'écran.

Voici les termes que l'assistant doit maîtriser. Si ce qu'il vous renvoie les emploie de façon approximative, il n'a pas compris le module :

Terme	Ce qu'il signifie ici
UserPaint	L'interrupteur explicite: DataGridViewCell.UserPaint = true pour une cellule, DataGridViewColumn.UserPaint = true pour une colonne entière. Sans lui, la grille ne vous demande jamais de dessiner.
CellPaint	L'événement de niveau grille déclenché pour les cellules dessinées par votre code lorsqu'elles doivent être tracées; vous vous abonnez une fois sur la grille, pas une fois par ligne, et vous aiguillez dans le gestionnaire.
DataGridViewCellPaintEventArgs	Dérive de PaintEventArgs, il porte donc Graphics et ClipRectangle, et ajoute RowIndex et ColumnIndex pour qu'un seul gestionnaire serve plusieurs colonnes.
pas de helpers WinForms	Les arguments d'événement documentés de Wisej.NET n'exposent pas les méthodes de confort WinForms PaintBackground() et PaintContent(). Votre gestionnaire est propriétaire de tout le résultat visuel de la cellule.
e.ClipRectangle comme surface	Le rectangle de destination local de la cellule. Les colonnes peuvent être redimensionnées et la grille mise en page de façon responsive: chaque coordonnée que vous dessinez doit donc en être dérivée, et non d'une constante.
helpers de rendu purs	Des méthodes de la forme DrawProgressCell(Graphics g, Rectangle rect, double value), pour que le gestionnaire devienne du code d'aiguillage et que le même helper puisse plus tard dessiner dans un bitmap hors écran.
contenu de cellule AllowHtml	L'alternative native au navigateur: icônes, spans, libellés et CSS rendus par le client, et la raison pour laquelle la documentation réserve CellPaint aux visuels trop complexes pour du HTML.

Construisez-le — choisissez votre assistant

Les deux boutons ouvrent un nouvel onglet avec le **même prompt prêt à l'emploi** déjà rempli. Le prompt oriente le modèle vers la spécification de ce module — un unique fichier Markdown qui contient la leçon, l'énoncé du lab et les livrables — pour qu'il travaille sur le vrai contenu au lieu de deviner de quoi parle le Module 3. Vous pouvez ouvrir vous-même [module-03.md](#) pour voir exactement ce qu'il lira.



Utiliser ChatGPT

Ouvre chatgpt.com avec le prompt de ce module déjà chargé. Fonctionne avec GPT-5 ou Codex ; toute offre capable de naviguer sur le web récupérera la spécification.



Utiliser Claude

Ouvre claude.ai avec le même prompt déjà chargé. Si vous utilisez Claude Code, vous pouvez plutôt le coller dans un terminal ouvert sur un dossier vide.



Envoyez-le tel quel et vous obtenez un projet, pas un extrait

Le prompt se termine en demandant toute la solution sous forme d'un unique [.zip](#) , avec les étapes pour la compiler et l'exécuter. Téléchargez-le, ouvrez la solution dans Visual Studio, appuyez sur F5 et VisualOperationsStudio s'ouvre sur [localhost](#) . Il vous faut le SDK .NET et le package NuGet [Wisej-4](#) ; tout le reste, le prompt demande à l'assistant de l'écrire lui-même.

Je suis le Module 3 - « Le dessin des cellules du DataGridView » - du cours LearnWisej « Dessin et peinture dans Wisej.NET », et je veux construire avec toi l'exemple de ce module. Réponds-moi en français.

Commence par lire la spécification du module. C'est un unique fichier Markdown (en anglais) qui contient la leçon complète, l'énoncé du lab et les livrables :

<https://learnwisej.com/assets/courses/drawing-and-painting/specs/module-03.md>

Construis ensuite un projet Wisej.NET 4 complet que je puisse ouvrir dans Visual Studio :

1. Projet « VisualOperationsStudio », C#, net8.0 ou ultérieur, package NuGet Wisej-4 4.1.x, hôte Kestrel qui appelle app.UseWisej().
2. Construis ce que demande le lab de la spécification : Donnez à VisualOperationsStudio une grille d'exploitation avec deux colonnes dessinées par votre code. Ajoutez des colonnes explicites à gridOperations, mettez UserPaint = true sur les seules colonnes Health et Trend, et gérez CellPaint une seule fois : sortez immédiatement lorsque e.RowIndex est négatif ou que e.ColumnIndex ne désigne aucune des deux colonnes visées, puis dessinez une barre de progression colorée selon les seuils pour Health et une sparkline en polyligne pour Trend à l'intérieur de e.ClipRectangle. Lisez les valeurs de chaque ligne depuis l'objet MachineStatus lié plutôt qu'en parcourant une liste dans le gestionnaire, respectez les couleurs de la ligne sélectionnée, et gardez le gestionnaire exempt d'effets de bord. Liez 1 000 lignes générées, faites défiler, puis construisez le même indicateur d'état une seconde fois sous forme de cellule AllowHtml afin de pouvoir écrire noir sur blanc lequel vous livreriez.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - gridOperations avec des colonnes explicites et UserPaint = true sur les seules colonnes Health et Trend
 - Un unique gestionnaire CellPaint qui contrôle e.RowIndex et e.ColumnIndex avant de dessiner quoi que ce soit
 - Cellule de progression colorée selon les seuils et cellule sparkline en polyligne dessinées à l'intérieur de e.ClipRectangle
 - Valeurs de ligne lues depuis l'objet MachineStatus lié, sans recherche, allocation ni création de contrôle par cellule
 - Une colonne d'état AllowHtml construite en parallèle, plus une note comparant les deux sur un défilement de 1 000 lignes
4. Suis la spécification partout où elle nomme une classe, une propriété ou un événement Wisej.NET - utilise exactement celui-là. Là où elle laisse le choix, dis-moi quelle API tu as choisie et pourquoi.
5. Garde l'autorité côté serveur : l'état et la validation vivent dans .NET, le navigateur affiche et remonte les informations. Ne déplace jamais une décision dans JavaScript parce que c'est plus simple là-bas.
6. Rends le résultat démontrable : je dois pouvoir l'exécuter et voir le comportement dont traite ce module, y compris ce qui se passe quand quelque chose tourne mal.

Avant d'écrire la moindre ligne de code, reformule avec tes propres mots ce que tu vas construire et où vivra chaque élément, pour que je puisse corriger la conception tant que cela ne coûte pas cher.

Quand le code est terminé, rassemble toute la solution dans un unique fichier .zip téléchargeable et donne-moi les étapes exactes pour la compiler et l'exécuter.

Enfin, réponds à ces questions de la spécification en t'appuyant sur ta propre

implémentation :

- Un collègue colle un exemple WinForms de dessin de cellule qui commence par appeler `e.PaintBackground(...)`. Expliquez ce que les arguments d'événement de Wisej.NET vous donnent à la place, et ce que le gestionnaire doit désormais décider lui-même.
- Votre helper de sparkline code en dur une largeur de tracé de 100 pixels et rend bien sur votre écran. Décrivez deux actions de l'utilisateur qui le cassent, et ce que le helper devrait lire à la place.
- La colonne d'état HTML et la colonne Health dessinée signalent toutes deux une machine critique. Défendez celle que vous livreriez pour une grille que le support recherche et lit avec un lecteur d'écran, et dites ce que vous perdriez dans un cas comme dans l'autre.

Pose-moi des questions sur tout ce qui est ambigu avant de commencer.

Puis relisez ce qui vous revient

Un code qui compile n'est pas encore un code juste. Les questions intéressantes sont des questions de conception : est-ce bien le contrôle Wisej.NET que la tâche exige, suit-il le cycle de vie que le framework vous offre, l'état de référence est-il sur le serveur ou a-t-il été laissé dans le navigateur parce que c'était pratique ? Lisez le résultat en relecteur, pas en client.

Ce module vous donne la checklist. Répondez à chaque point en vous appuyant sur le code que vous venez de recevoir :

- ✓ Un collègue colle un exemple WinForms de dessin de cellule qui commence par appeler `e.PaintBackground(...)`. Expliquez ce que les arguments d'événement de Wisej.NET vous donnent à la place, et ce que le gestionnaire doit désormais décider lui-même.
- ✓ Votre helper de sparkline code en dur une largeur de tracé de 100 pixels et rend bien sur votre écran. Décrivez deux actions de l'utilisateur qui le cassent, et ce que le helper devrait lire à la place.
- ✓ La colonne d'état HTML et la colonne Health dessinée signalent toutes deux une machine critique. Défendez celle que vous livreriez pour une grille que le support recherche et lit avec un lecteur d'écran, et dites ce que vous perdriez dans un cas comme dans l'autre.

Tout ce que vous avez dû corriger mérite d'être noté. Une courte note des conventions et des décisions retenues, collée dans le premier message de votre prochaine session, vaut plus que n'importe quel prompt astucieux.



Pas encore d'exemple de référence pour ce cours — le vôtre est le premier

Le [dépôt des exemples LearnWisej](#) contient des solutions complètes pour la plupart des cours, mais pas encore pour celui-ci. Il n'y a donc rien à quoi vous comparer et aucun raccourci : la spécification et votre propre relecture sont votre seul filet de sécurité. Gardez ce que vous construisez — et si le résultat est réussi, nous aimerions le voir.

Trois prompts pour aller plus loin

Orientez votre assistant vers le projet que vous venez de construire et essayez ceux-ci. Chacun est un exercice de relecture et d'extension, pas une réécriture.

- ✓ **Relisez-le comme une pull request.** *"Fais-moi parcourir ce code VisualOperationsStudio comme si tu relisais la pull request d'un collègue et que tu ne l'avais pas écrit. Où vit l'état, que se passe-t-il de bout en bout lors d'une action de l'utilisateur, que renverrais-tu pour correction, et quelles parties ne serais-tu pas à l'aise de montrer à un client ? Sois précis et cite les lignes dont tu parles — ne me résume pas la conception."*
- ✓ **Remplacez une pièce sans déplacer les jointures.** *"Choisis la partie de ce projet VisualOperationsStudio qui remplace le plus manifestement quelque chose de réel — une source de données factice, un stockage en mémoire, un service simplifié — et remplace-la par une implémentation digne de la production. Le contrat de part et d'autre ne doit pas changer : mêmes types, mêmes appels, même comportement en cas d'échec. Dis-moi ce que l'implémentation réelle t'a obligé à gérer et que la version provisoire ne gérait pas."*
- ✓ **Faites-le résister aux mauvais jours.** *"Examine ce projet VisualOperationsStudio pour repérer les modes de défaillance qu'il ne gère pas encore — une saisie erronée ou manquante, une dépendance lente ou indisponible, un traitement qui aboutit après que l'utilisateur a changé de page, et deux actions qui se chevauchent. Pour chacun, dis-moi d'abord s'il s'agit d'un risque réel dans Wisej.NET 4 et pourquoi, et seulement ensuite écris la correction. Là où tu estimes qu'aucune gestion n'est nécessaire, dis-le et explique ton raisonnement au lieu d'ajouter du code défensif."*



Emportez-le dans le lab

Le lab pratique qui clôt ce module demande les mêmes livrables que ceux que vous venez de faire produire à l'assistant. Servez-vous de ce que vous avez ici comme forme de votre réponse et comme preuve à l'appui — pas comme la réponse elle-même. Le lab est noté sur votre raisonnement, et cette partie doit venir de vous.