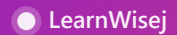


Exercice de code avec l'IA

Construisez l'exemple OperationsConsole 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 — y compris en le comparant à notre propre solution de référence sur GitHub.



Le même module, construit comme vous construirez vraiment.

Toutes les autres sections de ce module expliquent comment fonctionne « Maîtrise du DataGridView ». Celle-ci vous demande de le produire — en pilotant un assistant avec la spécification du module elle-même, puis en vérifiant que le résultat s'exécute et fait ce que décrit l'énoncé du lab. C'est la méthode de travail que vous utiliserez sur de vrais projets, mise en pratique sur un sujet dont vous connaissez déjà la bonne réponse.

Ce que vous demandez

Créez la grille Orders d'Operations Console : un DataGridView lié à une BindingSource avec des colonnes explicites et AutoGenerateColumns maîtrisé, une cellule de statut rendue avec AllowHtml et CellFormatting, une colonne d'éditeur personnalisé ou de commande, une version en mode virtuel ou adossée à un cache pour le chemin des gros volumes, et une barre de filtre et une barre d'état composées, ancrées dans la grille ou autour.

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

- ✓ DataGridView Orders avec des colonnes explicites liées à une BindingSource
- ✓ Cellule de statut rendue avec AllowHtml et CellFormatting
- ✓ Une colonne d'éditeur personnalisé ou de commande
- ✓ Version de la grille en mode virtuel ou adossée à un cache pour les gros volumes
- ✓ Barre de filtre et barre d'état composées, ancrées dans la grille ou autour

Pourquoi il est conçu ainsi

DataGridView est le contrôle qui décide le plus souvent si une application métier paraît mature.

Commandes, factures, échéances, journaux, utilisateurs et documents finissent tous dans une grille, et les utilisateurs jugent l'application à la façon dont cette grille se charge, défile, s'édite et signale son état.

Wisej.NET offre tout, du remplissage manuel au mode virtuel et au dessin de cellule; la maîtrise consiste à choisir la bonne technique pour le volume de données et la tâche de l'utilisateur.

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
BindingSource	Un intermédiaire côté serveur entre les lignes du view model et la grille, qui lui fournit une source de données stable, une position courante et le tri sans coupler l'écran à la base de données.
AutoGenerateColumns	Crée une colonne pour chaque propriété publique du type lié; pratique pour un prototype, dangereux en production, car un changement de modèle réorganise l'écran à votre insu.
colonne explicite	Une colonne que vous définissez avec liaison de champ, texte d'en-tête, largeur, alignement, format, lecture seule, mode de tri et type d'éditeur, pour que l'interface reste stable quand le modèle change.
mode virtuel	Avec VirtualMode activé, la grille ne connaît que son nombre de lignes et demande chaque valeur visible via CellValueNeeded, si bien que rien n'est créé pour les lignes que personne ne voit.
cache côté serveur	Un tampon de la taille d'une page qui alimente CellValueNeeded pour que le défilement ne sollicite pas la base à chaque cellule; l'événement DataRead est le point d'accroche documenté pour le construire et l'entretenir.
éditeur de cellule personnalisé	Un contrôle affecté à la propriété Editor d'une colonne; il doit renvoyer sa valeur dans la cellule, via CellBeginEdit et CellEndEdit s'il n'a pas d'événement de changement adapté.
HTML de cellule ou contrôle de cellule	Le HTML produit dans CellFormatting avec AllowHtml est un markup d'affichage quasi gratuit; un contrôle intégré est un vrai contrôle serveur et un widget navigateur, avec un coût par cellule.

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 5. Vous pouvez ouvrir vous-même [module-05.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 OperationsConsole 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 5 - « Maîtrise du DataGridView » - du cours LearnWisej « Maîtriser la bibliothèque de contrôles », 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/mastering-the-control-library/specs/module-05.md>

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

1. Projet « OperationsConsole », 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 : Créez la grille Orders d'Operations Console : un DataGridView lié à une BindingSource avec des colonnes explicites et AutoGenerateColumns maîtrisé, une cellule de statut rendue avec AllowHtml et CellFormatting, une colonne d'éditeur personnalisé ou de commande, une version en mode virtuel ou adossée à un cache pour le chemin des gros volumes, et une barre de filtre et une barre d'état composées, ancrées dans la grille ou autour.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - DataGridView Orders avec des colonnes explicites liées à une BindingSource
 - Cellule de statut rendue avec AllowHtml et CellFormatting
 - Une colonne d'éditeur personnalisé ou de commande
 - Version de la grille en mode virtuel ou adossée à un cache pour les gros volumes
 - Barre de filtre et barre d'état composées, ancrées dans la grille ou autour
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 :

- Si OrderRow gagne deux nouvelles propriétés au prochain sprint, qu'est-ce qui change à l'écran avec des colonnes générées, et qu'est-ce qui change avec vos colonnes explicites?
- Votre gestionnaire CellValueNeeded fonctionne avec 200 commandes. Qu'arrive-t-il à la base de données quand un utilisateur fait glisser la barre de défilement sur 200 000, et quelle partie de votre conception absorbe cette charge?
- Pourquoi le badge de statut a-t-il sa place dans CellFormatting plutôt que dans le modèle OrderRow, et que se passerait-il si un nom de client était placé dans ce HTML sans encodage?

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 :

- ✓ Si OrderRow gagne deux nouvelles propriétés au prochain sprint, qu'est-ce qui change à l'écran avec des colonnes générées, et qu'est-ce qui change avec vos colonnes explicites?
- ✓ Votre gestionnaire CellValueNeeded fonctionne avec 200 commandes. Qu'arrive-t-il à la base de données quand un utilisateur fait glisser la barre de défilement sur 200 000, et quelle partie de votre conception absorbe cette charge?
- ✓ Pourquoi le badge de statut a-t-il sa place dans CellFormatting plutôt que dans le modèle OrderRow, et que se passerait-il si un nom de client était placé dans ce HTML sans encodage?

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.



Nous l'avons déjà construit — comparez votre version à la nôtre

L'implémentation de référence du Module 5 se trouve dans le dépôt public des exemples. Construisez d'abord la vôtre, puis ouvrez la nôtre : ce sont les différences qui sont intéressantes.



[Ouvrir l'exemple du Module 5 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Mastering the Control Library Course/Module 5`.



Notre exemple est un point de départ, pas un produit fini

Il a été construit de la même manière que vous construirez le vôtre — par prompt, puis corrigé — et il s'arrête volontairement là où s'arrête le module. Attendez-vous à des éléments provisoires là où irait du code de production, à une gestion des erreurs minimale et à l'absence de tests. Voyez-y une invitation : utilisez ce que vous avez appris dans ce module, avec l'assistant, pour aller plus loin. Améliorer du code que vous n'avez pas écrit, c'est précisément la compétence que ce cours enseigne.

Trois prompts pour aller plus loin

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

- ✓ **Remplacez une pièce sans déplacer les jointures.** *"Choisis la partie de ce projet OperationsConsole 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 OperationsConsole 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."*
- ✓ **Ajoutez le deuxième cas.** *"Étends ce projet OperationsConsole avec un deuxième cas qui suit le même modèle que le premier — une autre entité, un autre écran ou un autre contrôle de même forme, câblé de la même manière. Extrais dans une classe de base ou un service ce qui est réellement partagé, mais seulement là où cela se justifie ; deux choses similaires n'appellent pas toujours une abstraction. Montre-moi le diff et justifie chaque élément que tu as extrait."*



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.