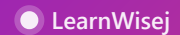


Exercice de code avec l'IA

Construisez l'exemple OpsMonitor 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 « Contrôles de données, charges utiles du navigateur et grandes surfaces d'interface ». 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

Réduisez les deux plus grandes surfaces de WisejPerfLab. La grille des tickets est liée à 50 000 entités complètes avec leurs propriétés de navigation : remplacez cette liaison par une projection TicketGridRow limitée aux colonnes visibles, définissez VirtualMode et RowCount et servez les cellules depuis un gestionnaire CellValueNeeded adossé à un service de requêtes paginées. L'arborescence des clients construit tous les nœuds au démarrage : chargez plutôt les enfants au déploiement d'un nœud, avec un espace réservé indiquant le nombre d'enfants pour les branches repliées. Mesurez chaque écran avant et après avec CPU Usage et .NET Object Allocation côté serveur et avec le panneau réseau du navigateur pour la taille de la charge utile des mises à jour.

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

- ✓ Projection TicketGridRow liée à la place des entités complètes, avec uniquement les colonnes affichées
- ✓ Grille en mode virtuel avec RowCount et un gestionnaire CellValueNeeded servi par un service de requêtes paginées
- ✓ Arborescence des clients chargeant les nœuds enfants au déploiement, avec un espace réservé de comptage pour les branches repliées
- ✓ Chiffres CPU et d'allocation avant/après pour les scénarios de chargement de la grille et de déploiement de l'arborescence
- ✓ Preuves réseau du navigateur comparant la taille de la charge utile des mises à jour avant et après, avec la réserve concernant la compression

Pourquoi il est conçu ainsi

Les grandes surfaces d'interface sont la source la plus courante de problèmes de performance dans Wisej.NET, et c'est la catégorie que le profileur ne voit pas entièrement. La grille des tickets de WisejPerfLab est liée à cinquante mille entités avec leurs propriétés de navigation; l'arborescence des clients construit tous ses nœuds au démarrage. Sur un poste de travail, ce schéma était acceptable, car le contrôle et les données vivaient dans le même processus que l'écran. Ici, le même code paie quatre fois: la mémoire serveur pour retenir les objets, le CPU serveur pour construire et sérialiser l'état de l'interface, le réseau pour transporter la mise à jour et le temps du navigateur pour l'afficher. C'est dans ce module que la meilleure optimisation n'est généralement pas un algorithme plus rapide, mais moins d'interface existant en même temps.

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
ligne de projection	Un petit type qui ne contient que les colonnes affichées par un écran, lié à la place du graphe d'entités, pour que ni le serveur ni le navigateur ne transportent des données que personne ne regarde.
mode virtuel	La grille conserve un nombre de lignes plutôt que des objets ligne et demande les valeurs des cellules au fur et à mesure des besoins, ce qui supprime les objets par ligne et le rendu des lignes jusqu'auxquelles personne n'a fait défiler.
CellValueNeeded	L'événement qui fournit une valeur à la demande en mode virtuel; il s'exécute très souvent, c'est donc par construction un chemin critique qui doit répondre rapidement.
pagination et filtrage côté serveur	Demander à la base de données la page que l'utilisateur regarde, au lieu de tout charger et de filtrer dans le code d'interface.
chargement différé des nœuds	Construire les enfants d'une arborescence au déploiement d'une branche, avec un espace réservé de comptage tant qu'elle est repliée, au lieu de matérialiser toute la hiérarchie au démarrage.
charge utile des mises à jour	La différence sérialisée que le serveur envoie après un gestionnaire; des charges utiles volumineuses ou fréquentes coûtent du temps réseau et navigateur qu'aucun outil Visual Studio ne signale.
la compression ne supprime pas le travail	Wisej.NET compresse le trafic HTTP et WebSocket, ce qui réduit le nombre d'octets mais ne rend pas une mise à jour inutile peu coûteuse à produire, à transmettre ou à afficher.

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 OpsMonitor 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 – « Contrôles de données, charges utiles du navigateur et grandes surfaces d'interface » – du cours LearnWisej « Performances et profilage », 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/performance-and-profiling/specs/module-05.md>

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

1. Projet « OpsMonitor », 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 : Réduisez les deux plus grandes surfaces de WisejPerfLab. La grille des tickets est liée à 50 000 entités complètes avec leurs propriétés de navigation : remplacez cette liaison par une projection `TicketGridRow` limitée aux colonnes visibles, définissez `VirtualMode` et `RowCount` et servez les cellules depuis un gestionnaire `CellValueNeeded` adossé à un service de requêtes paginées. L'arborescence des clients construit tous les nœuds au démarrage : chargez plutôt les enfants au déploiement d'un nœud, avec un espace réservé indiquant le nombre d'enfants pour les branches repliées. Mesurez chaque écran avant et après avec CPU Usage et .NET Object Allocation côté serveur et avec le panneau réseau du navigateur pour la taille de la charge utile des mises à jour.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Projection `TicketGridRow` liée à la place des entités complètes, avec uniquement les colonnes affichées
 - Grille en mode virtuel avec `RowCount` et un gestionnaire `CellValueNeeded` servi par un service de requêtes paginées
 - Arborescence des clients chargeant les nœuds enfants au déploiement, avec un espace réservé de comptage pour les branches repliées
 - Chiffres CPU et d'allocation avant/après pour les scénarios de chargement de la grille et de déploiement de l'arborescence
 - Preuves réseau du navigateur comparant la taille de la charge utile des mises à jour avant et après, avec la réserve concernant la compression
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 :

- Le mode virtuel a supprimé les objets ligne, mais votre trace CPU s'est à peine améliorée.

Qu'est-ce que cela vous apprend sur l'endroit où se trouvait réellement le coût, et quelle mesure examineriez-vous ensuite?

- Votre gestionnaire CellValueNeeded est correct mais lent lors d'un défilement rapide. Citez trois choses qu'il pourrait faire et qu'un gestionnaire exécuté par cellule ne doit jamais faire.
- La charge utile compressée du déploiement de l'arborescence a à peine changé après le passage au chargement différé, mais les utilisateurs trouvent l'écran plus rapide. Donnez une explication plausible que le seul nombre d'octets masquerait.

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 :

- ✓ Le mode virtuel a supprimé les objets ligne, mais votre trace CPU s'est à peine améliorée. Qu'est-ce que cela vous apprend sur l'endroit où se trouvait réellement le coût, et quelle mesure examineriez-vous ensuite?
- ✓ Votre gestionnaire CellValueNeeded est correct mais lent lors d'un défilement rapide. Citez trois choses qu'il pourrait faire et qu'un gestionnaire exécuté par cellule ne doit jamais faire.
- ✓ La charge utile compressée du déploiement de l'arborescence a à peine changé après le passage au chargement différé, mais les utilisateurs trouvent l'écran plus rapide. Donnez une explication plausible que le seul nombre d'octets masquerait.

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 `Performance and Profiling Course/Module 5`.



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

Il a été construit de la même manière que vous construisez 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.

- ✓ **Faites-le résister aux mauvais jours.** *"Examine ce projet OpsMonitor 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 OpsMonitor 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."*
- ✓ **Prouvez que ça fonctionne vraiment.** *"Passe en revue ce projet livrable par livrable par rapport à la spécification du module sur <https://learnwisej.com/assets/courses/performance-and-profiling/specs/module-05.md> et montre-moi la preuve que chacun est réellement terminé — le chemin dans le code, où cliquer et ce que je dois voir. Là où un livrable n'est que partiellement satisfait, dis-le clairement au lieu de le défendre. Ensuite, ajoute ce qui manque."*



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.