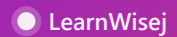


Exercice de code avec l'IA

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.



Écrire le code, c'est désormais la moitié facile.

Obtenir d'un assistant moderne du code qui fonctionne pour « Listes, arbres, répéteurs et données hiérarchiques » ne demande qu'un bon prompt. Le juger, le corriger et l'étendre demande tout ce que ce module a couvert. C'est sur ce partage que repose l'exercice : vous apportez la spécification et la relecture, l'assistant apporte la frappe.

Ce que vous demandez

Créez l'explorateur de documents d'Operations Console : un TreeView de catégories ou de dossiers avec un ID stable attaché à chaque nœud, un ListView de documents lié au nœud sélectionné, un chargement différé ou un mode virtuel sur au moins un chemin de liste ou d'arbre, des icônes ou des clés d'image pour les différents types d'éléments, et un UserControl de détail chargé via une couche service ou view model à chaque changement de sélection.

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

- ✓ TreeView de catégories avec un ID stable stocké sur chaque nœud
- ✓ ListView de documents lié au nœud d'arbre sélectionné
- ✓ Chargement différé ou mode virtuel sur au moins un chemin de liste ou d'arbre
- ✓ UserControl de détail chargé via une couche service ou view model lors de la sélection
- ✓ Icônes ou clés d'image pour les différents types d'éléments

Pourquoi il est conçu ainsi

Les contrôles de liste et d'arbre sont souvent considérés comme de simples widgets de sélection, mais dans une application de gestion ils sont la carte de l'utilisateur: dossiers, clients, commandes, procédures, rôles, tâches, documents, catalogues de produits et modules de l'application elle-même. L'explorateur de documents d'Operations Console que vous construisez dans ce module doit rester rapide quand l'arbre des catégories est profond et la liste des documents longue, et chaque clic doit correspondre à un véritable enregistrement métier plutôt qu'à un morceau de texte affiché.

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
forme des données	La première question de conception: les données sont-elles plates, groupées, hiérarchiques, répétées en cartes, ou un ensemble de propriétés? La réponse détermine le contrôle.
vues du ListView	Un seul contrôle qui affiche la même collection en LargeIcon, SmallIcon, Details ou Tile: le choix naturel pour les écrans de type fichiers ou catalogue.
mode virtuel	Un mode du ListView dans lequel l'application définit VirtualListSize et répond à RetrieveVirtualItem, si bien que les éléments n'existent que pour les lignes que le contrôle demande.
chargement différé	Ne charger que les niveaux supérieurs d'un arbre au démarrage et récupérer les enfants au déploiement d'un nœud, pour que le démarrage reste peu coûteux et l'interface réactive.
défilement virtuel et préchargement	N'afficher dans le navigateur que les lignes visibles d'une grande liste ou d'un grand arbre, avec PrefetchItems qui pré-affiche quelques lignes supplémentaires pour un défilement fluide.
identité stable	Un ID métier unique stocké dans le Tag ou le Name du nœud ou de l'élément, ou dans une table de correspondance du view model, jamais déduit du texte affiché.
la sélection comme événement	Un changement de sélection résout un ID et demande les données à un service; aucune requête ne s'exécute jamais dans le code qui construit les nœuds ou les éléments.

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 4. Vous pouvez ouvrir vous-même [module-04.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 4 - « Listes, arbres, répéteurs et données hiérarchiques » - 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-04.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 l'explorateur de documents d'Operations Console : un TreeView de catégories ou de dossiers avec un ID stable attaché à chaque nœud, un ListView de documents lié au nœud sélectionné, un chargement différé ou un mode virtuel sur au moins un chemin de liste ou d'arbre, des icônes ou des clés d'image pour les différents types d'éléments, et un UserControl de détail chargé via une couche service ou view model à chaque changement de sélection.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - TreeView de catégories avec un ID stable stocké sur chaque nœud
 - ListView de documents lié au nœud d'arbre sélectionné
 - Chargement différé ou mode virtuel sur au moins un chemin de liste ou d'arbre
 - UserControl de détail chargé via une couche service ou view model lors de la sélection
 - Icônes ou clés d'image pour les différents types d'éléments
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 :

- Deux catégories rattachées à des clients différents s'appellent toutes deux « Contracts ». Qu'utilise exactement votre gestionnaire de sélection pour les distinguer, et où se trouve cette valeur?
- Si le nombre de documents d'une catégorie passait de 300 à 300 000, quelle partie de votre explorateur casserait en premier, et quelle propriété ou quel événement modifieriez-vous pour y remédier?
- Quelles lignes de votre gestionnaire de sélection pourraient passer dans un test unitaire sans navigateur, et lesquelles relèvent plutôt du UserControl de détail?

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 :

- ✓ Deux catégories rattachées à des clients différents s'appellent toutes deux « Contracts ». Qu'utilise exactement votre gestionnaire de sélection pour les distinguer, et où se trouve cette valeur?
- ✓ Si le nombre de documents d'une catégorie passait de 300 à 300 000, quelle partie de votre explorateur casserait en premier, et quelle propriété ou quel événement modifieriez-vous pour y remédier?
- ✓ Quelles lignes de votre gestionnaire de sélection pourraient passer dans un test unitaire sans navigateur, et lesquelles relèvent plutôt du UserControl de détail?

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 4 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 4 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Mastering the Control Library Course/Module 4`.



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.

- ✓ **Relisez-le comme une pull request.** *"Fais-moi parcourir ce code OperationsConsole 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 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."*



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.