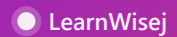


Exercice de code avec l'IA

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.



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 OperationsConsole ; 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

Remplacez les pages provisoires du Module 1 par le vrai shell d'Operations Console : un SplitContainer avec la navigation à gauche, une surface de détail TabControl à droite, une ToolBar pour les commandes fréquentes et une StatusBar qui affiche un état réel. Déplacez l'interface répétée dans des UserControls qui n'exposent que des propriétés et événements publics pertinents, mettez en œuvre une mise en page pour profil étroit avec des propriétés responsive ou une logique tenant compte du profil, et documentez vos choix de docking et d'ancrage dans LayoutNotes.md.

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

- ✓ Shell de navigation construit avec SplitContainer, TabControl, ToolBar et StatusBar
- ✓ Interface répétée déplacée dans des UserControls à surface publique réduite
- ✓ Mise en page pour profil étroit avec des propriétés responsive ou une logique tenant compte du profil
- ✓ LayoutNotes.md documentant les choix de docking et d'ancrage
- ✓ Captures d'écran en largeur bureau et en largeur étroite

Pourquoi il est conçu ainsi

Les conteneurs sont l'architecture de l'interface utilisateur: ils décident de la façon dont les contrôles sont regroupés, redimensionnés, parcourus et réutilisés. L'Operations Console du Module 1 a encore des panneaux provisoires, et un écran où chaque contrôle est positionné à la main devient fragile dès que le navigateur est redimensionné. Les problèmes de mise en page sont généralement des problèmes de conception, pas de pixels, et ce module remplace le placement au pixel près par un plan: des zones ancrées, un SplitContainer, un TabControl, un vrai modèle de commandes et des UserControls qui cachent leurs détails internes.

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
ordre de docking	Le docking attache un enfant à un côté de son conteneur ou lui fait remplir l'espace restant, et il s'applique dans l'ordre des enfants: les quatre mêmes contrôles produisent donc des shells différents selon l'ordre dans lequel ils ont été ajoutés.
ancrage	L'ancrage lie les bords d'un contrôle à ceux de son conteneur; ancré à gauche et à droite, il s'étire avec la largeur; ancré sur aucun axe, il flotte en conservant sa position relative.
SplitContainer	Deux zones adjacentes que l'utilisateur peut redimensionner: le bon choix chaque fois que c'est l'utilisateur, et non le développeur, qui doit décider de la place accordée à l'arbre de navigation ou au panneau de détail.
panneaux de mise en page	FlowLayoutPanel aligne les enfants côte à côte et passe à la ligne, TableLayoutPanel aligne lignes et colonnes, et FlexLayoutPanel répartit l'espace et passe à la ligne comme une mise en page responsive moderne.
propriétés responsive	Une propriété qui prend une valeur différente par profil client, définie au design, pour qu'une mise en page bureau puisse modifier quelques propriétés sur écran étroit sans second écran.
modèle de commandes	Une règle unique pour l'emplacement d'une commande: MenuBar, ToolBar, RibbonBar, ContextMenu ou StatusBar, chacun avec son rôle, tous invoquant des commandes de l'application au lieu de contenir de la logique métier.
surface d'un UserControl	L'unité de réutilisation: un contrôle composé qui expose quelques propriétés, événements et méthodes pertinents plutôt que chaque contrôle enfant, pour que les consommateurs ne dépendent jamais de son implémentation.

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 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 3 - « Conteneurs, mises en page, navigation et réutilisation » - 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-03.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 : Remplacez les pages provisoires du Module 1 par le vrai shell d'Operations Console : un SplitContainer avec la navigation à gauche, une surface de détail TabControl à droite, une ToolBar pour les commandes fréquentes et une StatusBar qui affiche un état réel. Déplacez l'interface répétée dans des UserControls qui n'exposent que des propriétés et événements publics pertinents, mettez en œuvre une mise en page pour profil étroit avec des propriétés responsive ou une logique tenant compte du profil, et documentez vos choix de docking et d'ancrage dans LayoutNotes.md.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Shell de navigation construit avec SplitContainer, TabControl, ToolBar et StatusBar
 - Interface répétée déplacée dans des UserControls à surface publique réduite
 - Mise en page pour profil étroit avec des propriétés responsive ou une logique tenant compte du profil
 - LayoutNotes.md documentant les choix de docking et d'ancrage
 - Captures d'écran en largeur bureau et en largeur étroite
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 le SplitContainer était ajouté à la page avant la ToolBar et la StatusBar, que verrait l'utilisateur, et pourquoi l'ordre est-il important?
- Quels membres de votre UserControl pourriez-vous renommer sans toucher aux pages consommatrices, et lesquels non? Qu'est-ce que cela vous apprend sur sa surface publique?
- Quelles propriétés ont changé pour le profil étroit, et pourquoi n'avez-vous pas construit un écran étroit distinct?

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 le SplitContainer était ajouté à la page avant la ToolBar et la StatusBar, que verrait l'utilisateur, et pourquoi l'ordre est-il important?
- ✓ Quels membres de votre UserControl pourriez-vous renommer sans toucher aux pages consommatrices, et lesquels non? Qu'est-ce que cela vous apprend sur sa surface publique?
- ✓ Quelles propriétés ont changé pour le profil étroit, et pourquoi n'avez-vous pas construit un écran étroit distinct?

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 3 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 3 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Mastering the Control Library Course/Module 3`.



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.

- ✓ **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/mastering-the-control-library/specs/module-03.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."*
- ✓ **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."*



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.