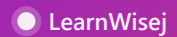


Exercice de code avec l'IA

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.



Vous ne taperez pas cet exemple. Vous le piloterez.

Presque plus personne n'écrit du code Wisej.NET à partir d'un fichier vide — un assistant rédige le premier jet et le développeur décide s'il est correct. Cela ne fonctionne que si vous maîtrisez le sujet, et c'est ce que le reste de ce module vous a apporté. Ici, vous confiez ces connaissances à ChatGPT ou à Claude, vous récupérez l'exemple du module sous forme de solution Visual Studio fonctionnelle, et vous le relisez comme vous reliriez la pull request d'un collègue.

Ce que vous demandez

Créez le shell du catalogue de contrôles d'Operations Console : une Page principale avec une zone de navigation, une zone de contenu, une zone de commandes et une zone d'état ; des pages provisoires pour Editors, Layouts, Lists and Trees, DataGridView, Dashboard et Widgets ; une note ControlSelection.md dans le projet qui indique la famille de contrôles que vous utiliserez pour chaque zone ; des noms de contrôles clairs partout ; et une zone d'état qui se met à jour à chaque changement de navigation.

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

- ✓ Shell principal avec zones de navigation, de contenu, de commandes et d'état
- ✓ Pages provisoires pour Editors, Layouts, Lists and Trees, DataGridView, Dashboard et Widgets
- ✓ ControlSelection.md indiquant la famille de contrôles choisie pour chaque zone
- ✓ Noms de contrôles clairs et zone d'état mise à jour à la navigation
- ✓ Capture d'écran et courte note expliquant les choix de contrôles

Pourquoi il est conçu ainsi

Wisej.NET fournit plus d'une centaine de contrôles, et une équipe sans modèle mental commun de ces contrôles finit avec vingt écrans de recherche différents, cinq manières de signaler une erreur de validation et du JavaScript personnalisé là où un contrôle natif suffirait. Ce module fixe le vocabulaire, les règles de décision et les conventions sur lesquels s'appuient tous les modules suivants, et démarre le shell d'Operations Console.

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
arbre des contrôles côté serveur	La hiérarchie d'objets contrôles.NET que le serveur conserve pour chaque session; votre code y définit les propriétés et y gère les événements, jamais dans du markup.
widget navigateur	L'objet JavaScript que Wisej.NET crée pour afficher un contrôle serveur, maintenu synchronisé par des changements d'état sérialisés, et non par des rechargements de page.
famille de contrôles	Un groupe de contrôles ayant une même finalité (éditeurs, conteneurs, listes, grilles, contenu, menus, notifications), au niveau duquel se prennent les décisions.
étendre	Un composant qui ajoute au design des propriétés ou un comportement à des contrôles existants, sans les sous-classer.
Widget et extension	Le contrôle Widget héberge un widget JavaScript personnalisé ou tiers au sein du modèle d'événements de Wisej.NET; une extension est une intégration packagée, généralement distribuée via NuGet.
UserControl	Une composition réutilisable de contrôles côté serveur, porteuse de sens métier et dotée d'une surface publique réduite.
guide de sélection des contrôles	Un document d'équipe vivant qui fixe, pour chaque type d'écran, les contrôles standard, le docking, le style de validation et de notification et les conventions de grille.

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 1. Vous pouvez ouvrir vous-même [module-01.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 1 – « Modèle mental de la bibliothèque de contrôles » – 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-01.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 le shell du catalogue de contrôles d'Operations Console : une Page principale avec une zone de navigation, une zone de contenu, une zone de commandes et une zone d'état ; des pages provisoires pour Editors, Layouts, Lists and Trees, DataGridView, Dashboard et Widgets ; une note `ControlSelection.md` dans le projet qui indique la famille de contrôles que vous utiliserez pour chaque zone ; des noms de contrôles clairs partout ; et une zone d'état qui se met à jour à chaque changement de navigation.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Shell principal avec zones de navigation, de contenu, de commandes et d'état
 - Pages provisoires pour Editors, Layouts, Lists and Trees, DataGridView, Dashboard et Widgets
 - `ControlSelection.md` indiquant la famille de contrôles choisie pour chaque zone
 - Noms de contrôles clairs et zone d'état mise à jour à la navigation
 - Capture d'écran et courte note expliquant les choix de contrôles
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 :

- Quand vous définissez `statusLabel.Text` dans la méthode de navigation, qu'est-ce qui transite réellement vers le navigateur, et en quoi est-ce différent d'écrire du HTML?
- Quelle zone du shell transformeriez-vous en premier en UserControl, et quelle serait sa surface publique?
- Si un module ultérieur a besoin d'une zone de navigation repliable, quel conteneur natif remplace le panneau ancré, et pourquoi est-ce préférable à un widget personnalisé?

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 :

- ✓ Quand vous définissez `statusLabel.Text` dans la méthode de navigation, qu'est-ce qui transite réellement vers le navigateur, et en quoi est-ce différent d'écrire du HTML?
- ✓ Quelle zone du shell transformeriez-vous en premier en `UserControl`, et quelle serait sa surface publique?
- ✓ Si un module ultérieur a besoin d'une zone de navigation repliable, quel conteneur natif remplace le panneau ancré, et pourquoi est-ce préférable à un widget personnalisé?

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 1 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 1 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Mastering the Control Library Course/Module 1`.



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 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."*
- ✓ **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-01.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.