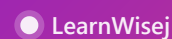


Exercice de code avec l'IA

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.



L'assistant, ce sont les mains rapides. L'architecte, c'est toujours vous.

Un assistant peut produire l'exemple de ce module en quelques minutes. Ce qu'il ne peut pas faire, c'est décider si le résultat est la bonne conception pour votre application — cette décision vous revient. C'est vous qui apportez le jugement : briefez-le correctement, puis vérifiez ce qu'il vous renvoie à la lumière de ce que le Module 7 vient de vous apprendre.

Ce que vous demandez

Encapsulez un petit widget JavaScript de notation ou d'indicateur avec le contrôle Widget, ou configurez un package d'extension Wisej.NET : chargez ses packages JavaScript et CSS, déclenchez un événement côté serveur depuis le client avec le modèle d'événement de widget, appelez une fonction côté client depuis le serveur avec CallAsync ou Eval, appliquez-lui un thème cohérent avec la console, puis peaufinez le projet de synthèse Operations Console complet et préparez une démo de 5 minutes.

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

- ✓ Écran basé sur un Widget ou une extension, avec ses packages JavaScript et CSS chargés
- ✓ Événement client vers serveur déclenché avec le modèle d'événement de widget ou un événement natif de l'extension
- ✓ Appel serveur vers client via CallAsync, Eval ou l'API de l'extension
- ✓ Contrat client/serveur documenté: packages, propriétés, événements et format du payload
- ✓ Projet de synthèse Operations Console peaufiné avec un script de démo de 5 minutes

Pourquoi il est conçu ainsi

Tôt ou tard, un produit a besoin de quelque chose que la bibliothèque native ne fournit pas: une capacité spécialisée du navigateur, une bibliothèque JavaScript tierce, une visualisation sur mesure ou une intégration réutilisable. La tentation est de basculer dans le JavaScript et de perdre le modèle Wisej.NET: l'état serveur, les thèmes, le designer et un flux d'événements qu'un autre développeur peut suivre. Ce module intègre un widget client sans renoncer à rien de tout cela, puis termine le projet de synthèse 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
contrôle natif	Un contrôle qui implémente à la fois le côté serveur et son widget navigateur, et dispose donc déjà du support au design, des thèmes, des événements et de l'état serveur; toujours le premier recours.
types d'extension	Control, Widget, Component, Extender Provider et Icon Pack: nommer le type évite de confondre « réutilisable » et « nouveau contrôle natif ».
contrôle Widget	Un conteneur client généraliste qui charge des packages JavaScript et CSS, exécute un script d'initialisation et échange événements et données avec le code serveur.
Packages et InitScript	La liste des packages déclare les fichiers client dont dépend le widget; l'InitScript crée l'objet JavaScript, branche ses événements client et n'expose que les fonctions dont le serveur a besoin.
WidgetEvent	L'événement serveur déclenché quand le code client appelle fireWidgetEvent; son type et son payload de données arrivent du navigateur et doivent être validés comme toute autre saisie utilisateur.
CallAsync / Eval	La moitié serveur vers client de la passerelle : appeler une fonction exposée par le widget, ou évaluer un script, sans quitter le C#.
contrat du widget	Les propriétés, événements, format du payload, dépendances de packages et règles de thème documentés qui permettent à un autre développeur, ou à un assistant IA correctement informé, de maintenir le widget.

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 7. Vous pouvez ouvrir vous-même [module-07.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 7 - « Widgets personnalisés, extensions, thèmes et projet de synthèse » - 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-07.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 : Encapsulez un petit widget JavaScript de notation ou d'indicateur avec le contrôle Widget, ou configurez un package d'extension Wisej.NET : chargez ses packages JavaScript et CSS, déclenchez un événement côté serveur depuis le client avec le modèle d'événement de widget, appelez une fonction côté client depuis le serveur avec `CallAsync` ou `Eval`, appliquez-lui un thème cohérent avec la console, puis peaufinez le projet de synthèse Operations Console complet et préparez une démo de 5 minutes.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Écran basé sur un Widget ou une extension, avec ses packages JavaScript et CSS chargés
 - Événement client vers serveur déclenché avec le modèle d'événement de widget ou un événement natif de l'extension
 - Appel serveur vers client via `CallAsync`, `Eval` ou l'API de l'extension
 - Contrat client/serveur documenté: packages, propriétés, événements et format du payload
 - Projet de synthèse Operations Console peaufiné avec un script de démo de 5 minutes
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 :

- Que se passerait-il si un utilisateur modifiait le payload `ratingChanged` dans les outils de développement du navigateur avant qu'il n'atteigne votre gestionnaire, et quelle ligne de votre code l'arrête?
- Votre widget s'affiche vide après que l'onglet Widgets a été masqué puis réaffiché: quels trois points vérifiez-vous, dans quel ordre, et pourquoi le HTML de démarrage n'est-il pas la solution?
- Quelle règle de votre widget pourriez-vous déplacer en JavaScript à titre d'optimisation, et laquelle ne doit jamais quitter le serveur?

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 :

- ✓ Que se passerait-il si un utilisateur modifiait le payload ratingChanged dans les outils de développement du navigateur avant qu'il n'atteigne votre gestionnaire, et quelle ligne de votre code l'arrête?
- ✓ Votre widget s'affiche vide après que l'onglet Widgets a été masqué puis réaffiché: quels trois points vérifiez-vous, dans quel ordre, et pourquoi le HTML de démarrage n'est-il pas la solution?
- ✓ Quelle règle de votre widget pourriez-vous déplacer en JavaScript à titre d'optimisation, et laquelle ne doit jamais quitter le serveur?

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 7 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 7 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Mastering the Control Library Course/Module 7`.



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.

- ✓ **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-07.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."*



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.