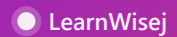


Exercice de code avec l'IA

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 l'onglet tableau de bord d'Operations Console : un graphique ChartJS en courbes ou en barres dont les libellés et les DataSets proviennent d'un view model de tableau de bord, une ProgressBar ou un indicateur de type jauge pour une valeur d'état, un aperçu de contenu PdfViewer ou HtmlPanel, un flux d'Upload, une carte ou un emplacement de contenu, et une méthode d'actualisation unique qui met à jour chaque contrôle du tableau de bord à partir d'un seul modèle.

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

- ✓ Onglet tableau de bord avec un graphique ChartJS en courbes ou en barres
- ✓ Libellés et DataSets du graphique construits à partir d'un view model de tableau de bord
- ✓ ProgressBar ou indicateur de type jauge pour une valeur d'état
- ✓ Aperçu de contenu PdfViewer ou HtmlPanel et flux d'Upload
- ✓ Une méthode d'actualisation unique qui met à jour chaque contrôle du tableau de bord

Pourquoi il est conçu ainsi

Un tableau de bord n'est pas qu'une affaire de graphiques. C'est une composition de contrôles qui transforme les données en décisions, et l'onglet tableau de bord d'Operations Console sera le premier écran qu'un responsable de service ouvrira chaque matin. Wisej.NET vous fournit des contrôles natifs de contenu et de médias ainsi que des extensions pour les graphiques, les cartes, l'affichage de documents et les jauges; la compétence intermédiaire consiste à savoir ce qui relève du modèle de contrôles Wisej.NET, ce qu'il vaut mieux confier à une bibliothèque JavaScript spécialisée encapsulée en extension ou en Widget, et comment faire en sorte que tout l'écran s'actualise comme un tout plutôt que comme un empilement de visuels qui se rechargent chacun de leur côté.

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
tableau de bord	Une composition de contrôles dont le seul rôle est de répondre à une question de décision; si un visuel n'aide personne à décider quoi que ce soit, il encombre.
extension ChartJS	Un package NuGet qui expose Chart.js comme composant côté serveur: le serveur détient l'état du graphique, le navigateur affiche le visuel, et le graphique rejoint le même modèle d'événements que tous les autres contrôles.
libellés et DataSets	Les deux moitiés des données d'un graphique: les libellés nomment les points le long de l'axe et chaque DataSet porte une série, c'est pourquoi on les définit toujours ensemble.
view model de tableau de bord	Un objet taillé pour l'affichage, comme DashboardModel, qui contient les mois, les comptages et une valeur d'achèvement, séparé de toute entité de base de données et du code qui le dessine.
frontière navigateur	Le client est le navigateur et le serveur est l'endroit où s'exécute votre code Wisej.NET; fichiers, PDF, HTML et tuiles de carte franchissent tous cette frontière dans un sens ou dans l'autre.
Upload ou chemin local	Upload transfère un fichier du navigateur vers le serveur; ce n'est pas la même chose que lire un chemin du poste, car le serveur ne peut jamais parcourir le disque de l'utilisateur.
actualisation par modèle	Un seul appel de service renvoie un modèle complet du tableau de bord et une seule méthode met à jour chaque contrôle à partir de ce modèle, par lot, avec un horodatage visible.

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 6. Vous pouvez ouvrir vous-même [module-06.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 6 - « Graphiques, tableaux de bord, contenu, médias et documents » - 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-06.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'onglet tableau de bord d'Operations Console : un graphique ChartJS en courbes ou en barres dont les libellés et les DataSets proviennent d'un view model de tableau de bord, une ProgressBar ou un indicateur de type jauge pour une valeur d'état, un aperçu de contenu PdfViewer ou HtmlPanel, un flux d'Upload, une carte ou un emplacement de contenu, et une méthode d'actualisation unique qui met à jour chaque contrôle du tableau de bord à partir d'un seul modèle.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Onglet tableau de bord avec un graphique ChartJS en courbes ou en barres
 - Libellés et DataSets du graphique construits à partir d'un view model de tableau de bord
 - ProgressBar ou indicateur de type jauge pour une valeur d'état
 - Aperçu de contenu PdfViewer ou HtmlPanel et flux d'Upload
 - Une méthode d'actualisation unique qui met à jour chaque contrôle du tableau de bord
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 :

- Quel contrôle de votre onglet tableau de bord fait le travail le plus important, et à quelle question personne ne répondrait plus si vous le retiriez?
- Qu'est-ce qui casserait en premier si le jeu de données des tickets devenait dix fois plus grand, et pourquoi le fait que le service renvoie des agrégats protège-t-il le graphique?
- Où se trouve le fichier envoyé à chaque étape du flux d'Upload, et pourquoi le serveur ne peut-il jamais ouvrir un chemin saisi par l'utilisateur?

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 :

- ✓ Quel contrôle de votre onglet tableau de bord fait le travail le plus important, et à quelle question personne ne répondrait plus si vous le retiriez?
- ✓ Qu'est-ce qui casserait en premier si le jeu de données des tickets devenait dix fois plus grand, et pourquoi le fait que le service renvoie des agrégats protège-t-il le graphique?
- ✓ Où se trouve le fichier envoyé à chaque étape du flux d'Upload, et pourquoi le serveur ne peut-il jamais ouvrir un chemin saisi par l'utilisateur?

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 6 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 6 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Mastering the Control Library Course/Module 6`.



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-06.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.