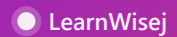


Exercice de code avec l'IA

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 2 vient de vous apprendre.

Ce que vous demandez

Créez un UserControl CustomerEditor pour Operations Console avec des champs libellés pour le nom, l'e-mail, le statut, la date de début, la limite de crédit et le type de client, en utilisant l'éditeur adapté à chaque valeur plutôt qu'une TextBox pour tout. Ajoutez des messages ErrorProvider pour les champs obligatoires, le format de l'e-mail, la plage de la limite de crédit et les dates invalides ; des commandes Save, Reset et Validate désactivées et signalant l'attente pendant l'enregistrement ; et un Toast ou une MessageBox à la fin de la validation ou de l'enregistrement.

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

- ✓ UserControl CustomerEditor avec des champs libellés pour le nom, l'e-mail, le statut, la date de début, la limite de crédit et le type de client
- ✓ Éditeurs adaptés: TextBox, DateTimePicker, NumericUpDown et ComboBox au lieu de texte libre partout
- ✓ Messages ErrorProvider pour les champs obligatoires, le format de l'e-mail, la plage de la limite de crédit et les dates invalides
- ✓ Commandes Save, Reset et Validate désactivées et signalant l'attente pendant l'enregistrement
- ✓ Retour Toast ou MessageBox à la fin de la validation ou de l'enregistrement

Pourquoi il est conçu ainsi

Les éditeurs sont le premier endroit où une bibliothèque de contrôles devient un système métier. Un formulaire qui semble correct mais accepte des saisies ambiguës ou invalides génère des tickets de support et des scripts de nettoyage longtemps après sa mise en production. Les éditeurs Wisej.NET suivent le modèle de composants.NET familier mais s'exécutent dans le navigateur, si bien qu'une même règle métier peut être appliquée à plusieurs niveaux: propriétés des éditeurs, extenders de validation,

gestionnaires d'événements, liaison de données, validation du service et contraintes de la base de données. Le rôle de la couche d'interface est d'éviter les erreurs évidentes et d'expliquer clairement les erreurs; elle ne doit jamais être la seule couche qui protège l'application.

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
éditeur adapté à la valeur	L'éditeur dont le type contraint déjà la saisie: DateTimePicker pour les dates, NumericUpDown pour les nombres bornés, ComboBox pour les valeurs de liste, afin que le serveur n'analyse jamais de texte ambigu.
modèle de commandes	Une répartition délibérée entre commandes principales (Save, Approve, Search, Send, Export), commandes secondaires (ToolBar, MenuBar, SplitButton, menu contextuel) et commandes destructrices qui exigent une confirmation ou un emplacement réfléchi.
ErrorProvider	L'extender qui attache un message précis et exploitable au contrôle erroné, au lieu d'un « Valeur invalide » générique ailleurs sur la page.
retour bloquant ou non bloquant	MessageBox interrompt pour une décision ou une confirmation critique; AlertBox et Toast signalent un état sans arrêter le déroulement.
validateurs nommés	Des méthodes comme ValidateEmail ou ValidateCreditLimit qui portent chacune une règle, pour que la règle soit testable et explicable au lieu d'être enfouie dans un gestionnaire de clic.
état d'attente	Désactiver Save et afficher une progression pendant une opération, ce qui empêche les doubles envois et relève de la fiabilité, pas de la décoration.
UserControl d'édition	Une composition de libellés, d'éditeurs, d'un ErrorProvider et d'une méthode de validation qui renvoie un objet résultat, pour que le reste de l'application ne sache jamais comment le champ e-mail est vérifié.

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 2. Vous pouvez ouvrir vous-même [module-02.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



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.

naviguer sur le web récupérera la spécification.



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 2 - « Éditeurs, boutons, validation et retour utilisateur » - 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-02.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 un UserControl CustomerEditor pour Operations Console avec des champs libellés pour le nom, l'e-mail, le statut, la date de début, la limite de crédit et le type de client, en utilisant l'éditeur adapté à chaque valeur plutôt qu'une TextBox pour tout. Ajoutez des messages ErrorProvider pour les champs obligatoires, le format de l'e-mail, la plage de la limite de crédit et les dates invalides ; des commandes Save, Reset et Validate désactivées et signalant l'attente pendant l'enregistrement ; et un Toast ou une MessageBox à la fin de la validation ou de l'enregistrement.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - UserControl CustomerEditor avec des champs libellés pour le nom, l'e-mail, le statut, la date de début, la limite de crédit et le type de client
 - Éditeurs adaptés: TextBox, DateTimePicker, NumericUpDown et ComboBox au lieu de texte libre partout
 - Messages ErrorProvider pour les champs obligatoires, le format de l'e-mail, la plage de la limite de crédit et les dates invalides
 - Commandes Save, Reset et Validate désactivées et signalant l'attente pendant l'enregistrement
 - Retour Toast ou MessageBox à la fin de la validation ou de l'enregistrement
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 :

- Quelles erreurs de validation ne peuvent plus du tout se produire grâce à l'éditeur choisi, et lesquelles nécessitent encore un message ErrorProvider?
- Si l'utilisateur clique deux fois sur Save en une seconde, que fait votre écran, et à quel endroit du code est-ce garanti?

– Que sait la page du shell de la manière dont le champ e-mail est validé, et que faudrait-il changer si la règle passait dans un service?

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 :

- ✓ Quelles erreurs de validation ne peuvent plus du tout se produire grâce à l'éditeur choisi, et lesquelles nécessitent encore un message ErrorProvider?
- ✓ Si l'utilisateur clique deux fois sur Save en une seconde, que fait votre écran, et à quel endroit du code est-ce garanti?
- ✓ Que sait la page du shell de la manière dont le champ e-mail est validé, et que faudrait-il changer si la règle passait dans un service?

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 2 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 2 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Mastering the Control Library Course/Module 2`.



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-02.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.