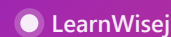


Exercice de code avec l'IA

Construisez l'exemple ValidationClinic 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 formulaire d'inscription client de ValidationClinic avec les TextBox txtName et txtEmail, un bouton Save et un bouton Cancel. Attachez des gestionnaires Validating qui définissent e.Cancel = true et un message ErrorProvider quand le nom est vide ou que l'adresse e-mail ne contient pas de @, effacent le message quand la valeur est valide, et supprimez les espaces du nom dans son gestionnaire Validated. Définissez le mode AutoValidate du formulaire, assurez-vous que Cancel ne déclenche pas de validation pour que l'utilisateur puisse toujours sortir, et appelez ValidateChildren() au début de btnSave_Click pour que Save s'arrête dès qu'un champ est invalide. Terminez par une courte note de lab qui explique le comportement du focus que vous avez choisi.

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

- ✓ Formulaire d'inscription client avec txtName et txtEmail, un bouton Save et un bouton Cancel
- ✓ Gestionnaires Validating qui définissent e.Cancel = true et un message d'erreur pour un nom vide et une adresse e-mail invalide
- ✓ Gestionnaire Validated qui supprime les espaces du nom accepté
- ✓ Gestionnaire de Save qui appelle ValidateChildren() et s'arrête quand la méthode renvoie false
- ✓ Note de lab indiquant le mode AutoValidate choisi et pourquoi Cancel fonctionne toujours sur un formulaire invalide

Pourquoi il est conçu ainsi

Le formulaire d'inscription client de ValidationClinic accepte actuellement n'importe quoi: un nom vide, une adresse e-mail sans @, ou un clic sur Save sur un formulaire que personne n'a rempli. Le premier réflexe de la plupart des débutants est une instruction if dans le bouton Save, mais l'utilisateur ne reçoit

alors aucun retour pendant la saisie et Save se transforme en mur d'erreurs tout à la fin. Ce module place la validation là où elle doit se trouver dans un formulaire Wisej.NET: sur chaque éditeur quand l'utilisateur le quitte, puis une nouvelle fois quand il clique sur Save.

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
validation	La décision de savoir si une valeur est acceptable avant que l'application ne l'utilise, suivie d'un retour visible et, quand la valeur est incorrecte, d'un refus de continuer.
Validating	L'événement qu'un contrôle déclenche quand l'utilisateur tente de le quitter (ou quand le code demande une validation); son gestionnaire lit la valeur et c'est le seul endroit où l'on décide d'accepter ou de rejeter.
CancelEventArgs.Cancel	L'indicateur qu'un gestionnaire Validating met à true pour rejeter la valeur: le contrôle est considéré comme invalide, le focus peut y être retenu et ValidateChildren() signale l'échec.
Validated	L'événement déclenché uniquement après un Validating terminé sans annulation; l'endroit pour les finitions, comme supprimer les espaces d'un texte, jamais pour enregistrer.
AutoValidate	Une propriété d'un formulaire, d'une page ou d'un contrôle utilisateur qui détermine ce qu'il advient du focus quand un contrôle enfant échoue à la validation.
CausesValidation	Un réglage propre à chaque contrôle; mis à false sur un bouton comme Cancel, un clic sur ce bouton ne valide pas le champ que l'utilisateur quitte.
ValidateChildren()	Une méthode de conteneur qui lance la validation de ses contrôles enfants, y compris les champs que l'utilisateur n'a jamais visités, et renvoie false si un gestionnaire a annulé.

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



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.

navigation 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 ValidationClinic 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 - « Fondamentaux de la validation dans Wisej.NET » - du cours LearnWisej « Validation dans Wisej.NET », 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/validation-in-wisej-net/specs/module-01.md>

Construis ensuite un projet Wisej.NET 4 complet que je puisse ouvrir dans Visual Studio :

1. Projet « ValidationClinic », 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 formulaire d'inscription client de ValidationClinic avec les TextBox `txtName` et `txtEmail`, un bouton Save et un bouton Cancel. Attachez des gestionnaires Validating qui définissent `e.Cancel = true` et un message ErrorProvider quand le nom est vide ou que l'adresse e-mail ne contient pas de @, effacent le message quand la valeur est valide, et supprimez les espaces du nom dans son gestionnaire Validated. Définissez le mode AutoValidate du formulaire, assurez-vous que Cancel ne déclenche pas de validation pour que l'utilisateur puisse toujours sortir, et appelez `ValidateChildren()` au début de `btnSave_Click` pour que Save s'arrête dès qu'un champ est invalide. Terminez par une courte note de lab qui explique le comportement du focus que vous avez choisi.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Formulaire d'inscription client avec `txtName` et `txtEmail`, un bouton Save et un bouton Cancel
 - Gestionnaires Validating qui définissent `e.Cancel = true` et un message d'erreur pour un nom vide et une adresse e-mail invalide
 - Gestionnaire Validated qui supprime les espaces du nom accepté
 - Gestionnaire de Save qui appelle `ValidateChildren()` et s'arrête quand la méthode renvoie false
 - Note de lab indiquant le mode AutoValidate choisi et pourquoi Cancel fonctionne toujours sur un formulaire invalide
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 :

- Vous retirez `e.Cancel = true` de `txtEmail_Validating` mais conservez l'appel à `SetError`. Décrivez précisément ce que voit l'utilisateur quand il quitte avec Tab le champ contenant «

maria.example.com », puis clique sur Save.

- Parmi vos vérifications, lesquelles s'exécutent pendant la saisie, et lesquelles uniquement parce que Save a appelé ValidateChildren()? Donnez une action utilisateur qui atteint Save avec un champ dont le gestionnaire ne s'est jamais exécuté.
- Vous passez le formulaire de EnablePreventFocusChange à EnableAllowFocusChange. Qu'est-ce qui change pour l'utilisateur, qu'est-ce qui reste identique, et btnCancel.CausesValidation = false est-il toujours utile?

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 :

- ✓ Vous retirez e.Cancel = true de txtEmail_Validating mais conservez l'appel à SetError. Décrivez précisément ce que voit l'utilisateur quand il quitte avec Tab le champ contenant « maria.example.com », puis clique sur Save.
- ✓ Parmi vos vérifications, lesquelles s'exécutent pendant la saisie, et lesquelles uniquement parce que Save a appelé ValidateChildren()? Donnez une action utilisateur qui atteint Save avec un champ dont le gestionnaire ne s'est jamais exécuté.
- ✓ Vous passez le formulaire de EnablePreventFocusChange à EnableAllowFocusChange. Qu'est-ce qui change pour l'utilisateur, qu'est-ce qui reste identique, et btnCancel.CausesValidation = false est-il toujours utile?

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 `Validation in Wisej.NET 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.

- ✓ **Remplacez une pièce sans déplacer les jointures.** *"Choisis la partie de ce projet ValidationClinic 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 gèrait pas."*
- ✓ **Faites-le résister aux mauvais jours.** *"Examine ce projet ValidationClinic 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 ValidationClinic 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."*



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.