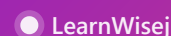


## Exercice de code avec l'IA

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

Faites en sorte que la grille des contacts de ValidationClinic refuse les données incorrectes. Gérez `gridContacts.CellValidating` : lisez `e.FormattedValue` plutôt que la valeur de la cellule, effacez l'`ErrorText` de la cellule, puis rejetez une valeur `EmailColumn` vide ou sans @ et une valeur `AgeColumn` en dehors de 0 à 120 avec `e.Cancel = true` et un `ErrorText` de cellule. Ajoutez une méthode `ValidateRow` qui définit l'`ErrorText` de ligne et l'erreur de la cellule `ClosedDateColumn` quand `Status` vaut `Closed` sans date de clôture, effacez les erreurs obsolètes dans `CellValidated`, et gérez `DataError` avec `ThrowException = false`, un message de cellule compréhensible et une trace de débogage de l'exception. Ignorez la nouvelle ligne de substitution tant qu'elle ne contient pas de données utilisateur, et bloquez `Save` tant qu'une ligne porte encore une erreur.

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

- ✓ Gestionnaire `CellValidating` qui lit `e.FormattedValue` et valide les colonnes e-mail et âge
- ✓ Méthode `ValidateRow` qui définit l'`ErrorText` de ligne et l'erreur de la cellule de date de clôture pour les contacts `Closed`
- ✓ Gestionnaire `CellValidated` qui efface les erreurs de cellule obsolètes
- ✓ Gestionnaire `DataError` avec `ThrowException = false`, un message compréhensible et une exception journalisée
- ✓ Garde-fou de `Save` qui ignore la nouvelle ligne de substitution et refuse d'enregistrer tant qu'une ligne porte une erreur

## Pourquoi il est conçu ainsi

Les Modules 1 à 5 ont protégé le formulaire d'inscription de ValidationClinic, un contact à la fois. La grille des contacts est différente: un utilisateur peut parcourir vingt lignes avec Tab, taper une adresse e-mail sans @, saisir « abc » comme âge et marquer un contact Closed sans date de clôture, le tout sans ouvrir de formulaire. Une cellule de grille n'est pas une TextBox; les gestionnaires Validating et l' ErrorProvider que vous avez déjà écrits ne l'atteignent donc pas. Ce module donne à la grille ses propres règles de champ, règles de ligne et règles de source de données, et veille à ce qu'aucune d'elles ne transforme une simple faute de frappe en plantage ou en écran bloqué.

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                                                                                                                                                                                              |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CellValidating       | L'événement de grille déclenché quand la cellule courante est sur le point de perdre le focus; c'est la version grille de l'événement Validating d'un contrôle et l'endroit où rejeter une valeur isolée.          |
| e.FormattedValue     | La valeur exactement telle que l'utilisateur l'a tapée dans l'éditeur, avant qu'elle ne soit validée dans la cellule ou la source de données; lire la Value de la cellule à la place vous donne l'ancienne valeur. |
| e.Cancel             | Le mettre à true refuse de terminer la modification: le curseur reste dans la cellule incorrecte jusqu'à ce que l'utilisateur la corrige ou annule la modification.                                                |
| CellValidated        | Déclenché après qu'une cellule a passé la validation; un endroit sûr pour effacer une erreur qui ne s'applique plus ou rafraîchir un récapitulatif.                                                                |
| ErrorText de cellule | Un message sur une seule cellule, affiché avec une icône d'erreur dans cette cellule; à utiliser pour un problème portant sur une seule valeur.                                                                    |
| ErrorText de ligne   | Un message sur la ligne entière, affiché dans l'en-tête de ligne; à utiliser pour une règle qui dépend de plusieurs cellules de la même ligne.                                                                     |
| DataError            | Déclenché quand la grille ne peut pas convertir, mettre en forme ou valider une valeur; avec e.ThrowException = false, vous remplacez l'exception par un message compréhensible.                                   |

## 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



Utiliser Claude

Ouvre [chatgpt.com](https://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.

Ouvre [claude.ai](https://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 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 6 - « Validation des cellules et erreurs de ligne dans DataGridView » - 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-06.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 : Faites en sorte que la grille des contacts de ValidationClinic refuse les données incorrectes. Gérez `gridContacts.CellValidating` : lisez `e.FormattedValue` plutôt que la valeur de la cellule, effacez l'`ErrorText` de la cellule, puis rejetez une valeur `EmailColumn` vide ou sans @ et une valeur `AgeColumn` en dehors de 0 à 120 avec `e.Cancel = true` et un `ErrorText` de cellule. Ajoutez une méthode `ValidateRow` qui définit l'`ErrorText` de ligne et l'erreur de la cellule `ClosedDateColumn` quand `Status` vaut `Closed` sans date de clôture, effacez les erreurs obsolètes dans `CellValidated`, et gérez `DataError` avec `ThrowException = false`, un message de cellule compréhensible et une trace de débogage de l'exception. Ignorez la nouvelle ligne de substitution tant qu'elle ne contient pas de données utilisateur, et bloquez `Save` tant qu'une ligne porte encore une erreur.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
  - Gestionnaire `CellValidating` qui lit `e.FormattedValue` et valide les colonnes e-mail et âge
  - Méthode `ValidateRow` qui définit l'`ErrorText` de ligne et l'erreur de la cellule de date de clôture pour les contacts `Closed`
  - Gestionnaire `CellValidated` qui efface les erreurs de cellule obsolètes
  - Gestionnaire `DataError` avec `ThrowException = false`, un message compréhensible et une exception journalisée
  - Garde-fou de `Save` qui ignore la nouvelle ligne de substitution et refuse d'enregistrer tant qu'une ligne porte une erreur
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 :

- La règle e-mail d'un camarade lit `grid.Rows[e.RowIndex].Cells[e.ColumnIndex].Value` et «

fonctionne en test ». Décrivez la modification exacte qui passe entre les mailles, et pourquoi.

– Votre règle d'âge vérifie la plage de 0 à 120. Que doit-elle faire quand le texte n'est pas du tout un nombre, et pourquoi un gestionnaire `DataError` reste-t-il nécessaire même si la règle rejette elle-même « abc »?

– La règle `Closed` peut s'afficher comme erreur de cellule, erreur de ligne, ou les deux. Qu'avez-vous choisi, et que verrait encore un utilisateur qui ne survole jamais une icône?

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 :

- ✓ La règle e-mail d'un camarade lit `grid.Rows[e.RowIndex].Cells[e.ColumnIndex].Value` et « fonctionne en test ». Décrivez la modification exacte qui passe entre les mailles, et pourquoi.
- ✓ Votre règle d'âge vérifie la plage de 0 à 120. Que doit-elle faire quand le texte n'est pas du tout un nombre, et pourquoi un gestionnaire `DataError` reste-t-il nécessaire même si la règle rejette elle-même « abc »?
- ✓ La règle `Closed` peut s'afficher comme erreur de cellule, erreur de ligne, ou les deux. Qu'avez-vous choisi, et que verrait encore un utilisateur qui ne survole jamais une icône?

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 `Validation in Wisej.NET 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.

- ✓ **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 gérait 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.