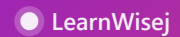


Exercice de code avec l'IA

Construisez l'exemple GlobalDesk 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.



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

Localisez l'éditeur client de GlobalDesk depuis le designer Visual Studio. Terminez d'abord la mise en page neutre en anglais du contrôle utilisateur CustomerEditor, puis passez Localizable à true et utilisez le sélecteur Language du designer pour ajouter une variante allemande, en ne changeant que ce qui doit différer : les libellés, la largeur des deux boutons les plus longs et la propriété Location d'un libellé. Exécutez l'application en de-DE et comparez le rendu réel du navigateur avec la vue du designer, puis ajoutez un bouton Recreate editor qui libère et reconstruit le contrôle pour constater par vous-même pourquoi les ressources du designer ne suivent pas un changement de culture à l'exécution. Terminez par une note de lab qui liste les propriétés que vous avez redéfinies et celles que vous avez délibérément laissées à la mise en page neutre.

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

- ✓ Contrôle utilisateur CustomerEditor avec Localizable à true sur une mise en page neutre en anglais complète
- ✓ Variante de langue allemande dans le designer qui redéfinit les libellés ainsi qu'au moins une Size et une Location
- ✓ Ancrage, docking ou AutoSize appliqués pour que les libellés allemands ne puissent pas être tronqués
- ✓ Bouton Recreate editor qui libère et reconstruit le conteneur après un changement de culture
- ✓ Note de lab listant les propriétés redéfinies et celles laissées à la mise en page neutre

Pourquoi il est conçu ainsi

Le Module 1 a donné à GlobalDesk un tableau de bord dont tous les libellés viennent de Strings.resx, et c'est bien le bon foyer pour le texte que le code génère. L'éditeur client, lui, est différent: c'est un écran conçu, agencé à l'œil dans le designer Visual Studio. Faire passer chacun de ses libellés par un appel à une classe utilitaire oblige à retoucher le fichier du designer à la main, et cela ne résout toujours pas le vrai

problème: un Kundennummer speichern allemand ne tient pas dans un bouton dimensionné pour Save. Ce module utilise le flux de localisation que Visual Studio propose déjà, de sorte que les libellés d'un écran conçu et sa géométrie peuvent varier d'une langue à l'autre, sans chaîne codée en dur et sans second fichier de mise en page.

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
Localizable	Une propriété de design sur un conteneur de premier niveau; true fait basculer le designer dans le flux de localisation, pour que les valeurs puissent être stockées par culture.
Sélecteur Language du designer	Apparaît dès que Localizable vaut true; la culture sélectionnée est la version que vous modifiez, si bien que chaque changement n'est enregistré que pour cette culture.
La langue neutre	La version par défaut sur laquelle toutes les cultures retombent, et la seule où l'on ajoute, supprime ou restructure des contrôles; les vues de culture contiennent des redéfinitions, pas un second écran.
Propriétés localisables	Pas seulement Text: Size, Location, les couleurs et les images peuvent chacune avoir une valeur différente par culture, parce que la traduction change la géométrie autant que les mots.
Fichiers de ressources générés	Le designer écrit les valeurs localisées dans des fichiers de ressources plutôt que dans le code C# du designer, ce qui garde les valeurs propres à une langue hors du code que vous lisez et fusionnez au quotidien.
Appliquées à la création	Les valeurs localisées dans le designer sont appliquées pendant la création des contrôles: changer la culture de la session plus tard ne réécrit donc pas un conteneur qui existe déjà.
Mise en page tolérante à la traduction	Le docking, l'ancrage, FlowLayoutPanel, TableLayoutPanel, FlexLayoutPanel, AutoSize et le padding laissent le texte grandir de lui-même: il vous faut donc moins de redéfinitions de géométrie.

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,

naviguer sur le web récupérera la spécification.

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 GlobalDesk 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 - « Localiser l'interface Wisej.NET dans le designer Visual Studio » - du cours LearnWisej « Localisation 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/localization-in-wisej-net/specs/module-02.md>

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

1. Projet « GlobalDesk », 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 : Localisez l'éditeur client de GlobalDesk depuis le designer Visual Studio. Terminez d'abord la mise en page neutre en anglais du contrôle utilisateur `CustomerEditor`, puis passez `Localizable` à `true` et utilisez le sélecteur `Language` du designer pour ajouter une variante allemande, en ne changeant que ce qui doit différer : les libellés, la largeur des deux boutons les plus longs et la propriété `Location` d'un libellé. Exécutez l'application en de-DE et comparez le rendu réel du navigateur avec la vue du designer, puis ajoutez un bouton `Recreate editor` qui libère et reconstruit le contrôle pour constater par vous-même pourquoi les ressources du designer ne suivent pas un changement de culture à l'exécution. Terminez par une note de lab qui liste les propriétés que vous avez redéfinies et celles que vous avez délibérément laissées à la mise en page neutre.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Contrôle utilisateur `CustomerEditor` avec `Localizable` à `true` sur une mise en page neutre en anglais complète
 - Variante de langue allemande dans le designer qui redéfinit les libellés ainsi qu'au moins une `Size` et une `Location`
 - Ancrage, docking ou `AutoSize` appliqués pour que les libellés allemands ne puissent pas être tronqués
 - Bouton `Recreate editor` qui libère et reconstruit le conteneur après un changement de culture
 - Note de lab listant les propriétés redéfinies et celles laissées à la mise en page neutre
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 :

- Un coéquipier ajoute un nouveau champ téléphone à CustomerEditor pendant que l'allemand est sélectionné dans la liste Language. Que verra probablement l'utilisateur anglais, et quelle règle l'aurait évité?
- La confirmation d'enregistrement que l'éditeur affiche après une mise à jour réussie vient toujours de Strings.resx plutôt que des ressources du designer. Donnez deux raisons pour lesquelles c'est le bon endroit, et nommez un libellé de cet écran qui relève au contraire vraiment du designer.
- Votre bouton Recreate editor fonctionne, mais un collègue soutient que l'écran devrait simplement réappliquer les propriétés localisées au lieu de se reconstruire. Que devrait connaître ce code que la reconstruction ignore, et dans quel cas son approche serait-elle le meilleur compromis?

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 :

- ✓ Un coéquipier ajoute un nouveau champ téléphone à CustomerEditor pendant que l'allemand est sélectionné dans la liste Language. Que verra probablement l'utilisateur anglais, et quelle règle l'aurait évité?
- ✓ La confirmation d'enregistrement que l'éditeur affiche après une mise à jour réussie vient toujours de Strings.resx plutôt que des ressources du designer. Donnez deux raisons pour lesquelles c'est le bon endroit, et nommez un libellé de cet écran qui relève au contraire vraiment du designer.
- ✓ Votre bouton Recreate editor fonctionne, mais un collègue soutient que l'écran devrait simplement réappliquer les propriétés localisées au lieu de se reconstruire. Que devrait connaître ce code que la reconstruction ignore, et dans quel cas son approche serait-elle le meilleur compromis?

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.



Pas encore d'exemple de référence pour ce cours — le vôtre est le premier

Le [dépôt des exemples LearnWisej](#) contient des solutions complètes pour la plupart des cours, mais pas encore pour celui-ci. Il n'y a donc rien à quoi vous comparer et aucun raccourci : la spécification et votre propre relecture sont votre seul filet de sécurité. Gardez ce que vous construisez — et si le résultat est réussi, nous aimerions le voir.

Trois prompts pour aller plus loin

Orientez votre assistant vers le projet que vous venez de construire et essayez ceux-ci. Chacun est un exercice de relecture et d'extension, pas une réécriture.

- ✓ **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/localization-in-wisej-net/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 GlobalDesk 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."*
- ✓ **Remplacez une pièce sans déplacer les jointures.** *"Choisis la partie de ce projet GlobalDesk 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."*



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.