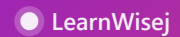


Exercice de code avec l'IA

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.



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 la page tableau de bord de GlobalDesk avec un titre de bienvenue, un bouton de navigation Customers et un petit panneau d'aperçu de la culture. Ajoutez Resources/Strings.resx comme ressource neutre avec au moins douze clés sémantiques stables telles que Navigation.Customers et CustomerEditor.Save, écrivez une classe utilitaire statique Texts au-dessus de Wisej.Resources.ResourceManager qui renvoie la clé entre crochets quand elle est absente, et remplacez chaque libellé codé en dur de la page par un appel à Texts.Get. Remplissez ensuite le panneau d'aperçu à partir de vraies valeurs : mettez en forme la date du jour, une quantité et un montant avec Application.CurrentCulture au lieu de composer les chaînes à la main, et terminez par une courte note de lab qui distingue la valeur stockée de sa présentation dépendante de la culture.

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

- ✓ Page tableau de bord de GlobalDesk dont tous les libellés viennent de clés de ressources, sans aucun texte visible codé en dur
- ✓ Fichier neutre Resources/Strings.resx contenant au moins douze clés sémantiques stables
- ✓ Classe utilitaire Texts au-dessus de Wisej.Resources.ResourceManager qui signale une clé manquante au lieu de renvoyer une chaîne vide
- ✓ Panneau d'aperçu de la culture qui met en forme une date, un nombre et un montant avec Application.CurrentCulture
- ✓ Note de lab expliquant quelles valeurs sont des données et lesquelles relèvent d'une présentation dépendante de la culture

Pourquoi il est conçu ainsi

GlobalDesk est un petit service desk écrit pour un seul bureau. Chaque titre, chaque libellé de bouton et chaque message de confirmation est un littéral anglais tapé directement dans le C#, et le tableau de bord

affiche les montants et les dates comme le ferait une machine américaine. À l'ouverture d'une filiale allemande, aucun de ces textes n'a nulle part où aller — et le développeur qui commence par retoucher les libellés à la main découvre la seconde moitié du problème: un libellé allemand au-dessus d'un montant qui se lit toujours 1,850.75 ne fait pas un écran allemand.

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
localisation	Adapter une application à la langue et à la culture d'un utilisateur: textes traduits, dates, nombres et montants correctement mis en forme, place pour des mots plus longs, et une mise en page capable de se mettre en miroir pour les langues de droite à gauche.
nom de culture	Un identifiant comme en, de-DE ou fr-CA qui nomme une langue et, éventuellement, une région; la partie langue sélectionne les ressources préférées, le nom complet pilote les conventions de mise en forme.
ressource neutre	Le jeu de ressources de repli complet, conservé dans un fichier comme Resources/Strings.resx sans culture dans son nom; chaque clé demandée par l'application doit y exister.
Wisej.Resources.ResourceManager	L'objet qui prend un nom de base de ressource et un assembly et renvoie la valeur d'une clé dans la culture la mieux adaptée, de sorte que votre code n'ouvre jamais un fichier lui-même.
repli de culture	La règle.NET pour trouver une valeur: essayer la culture complète, puis la culture langue seule, puis la ressource neutre, et utiliser la première qui possède la clé.
clé de ressource sémantique	Un nom stable qui décrit le sens plutôt que la formulation, comme CustomerEditor.Save ou Validation.Required, pour que le texte anglais puisse être reformulé sans toucher à aucune autre langue.
Application.CurrentCulture	La culture de la session Wisej.NET en cours; deux utilisateurs connectés en même temps peuvent avoir deux valeurs différentes, et c'est pourquoi la culture ne doit jamais vivre dans un champ statique.

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



Utiliser Claude

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.

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 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 1 - « Fondamentaux de la localisation : culture, langue et ressources » - 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-01.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 : Créez la page tableau de bord de GlobalDesk avec un titre de bienvenue, un bouton de navigation Customers et un petit panneau d'aperçu de la culture. Ajoutez `Resources/Strings.resx` comme ressource neutre avec au moins douze clés sémantiques stables telles que `Navigation.Customers` et `CustomerEditor.Save`, écrivez une classe utilitaire statique `Texts` au-dessus de `Wisej.Resources.ResourceManager` qui renvoie la clé entre crochets quand elle est absente, et remplacez chaque libellé codé en dur de la page par un appel à `Texts.Get`. Remplissez ensuite le panneau d'aperçu à partir de vraies valeurs : mettez en forme la date du jour, une quantité et un montant avec `Application.CurrentCulture` au lieu de composer les chaînes à la main, et terminez par une courte note de lab qui distingue la valeur stockée de sa présentation dépendante de la culture.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Page tableau de bord de GlobalDesk dont tous les libellés viennent de clés de ressources, sans aucun texte visible codé en dur
 - Fichier neutre `Resources/Strings.resx` contenant au moins douze clés sémantiques stables
 - Classe utilitaire `Texts` au-dessus de `Wisej.Resources.ResourceManager` qui signale une clé manquante au lieu de renvoyer une chaîne vide
 - Panneau d'aperçu de la culture qui met en forme une date, un nombre et un montant avec `Application.CurrentCulture`
 - Note de lab expliquant quelles valeurs sont des données et lesquelles relèvent d'une présentation dépendante de la culture
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 changez la formulation anglaise du bouton Save de "Save" à "Save customer". Quels

fichiers doivent changer si votre clé est `CustomerEditor.Save`, et quels fichiers auraient dû changer si la clé avait été la phrase anglaise elle-même?

- Votre classe utilitaire `Texts` renvoie `[Preview.Amount]` pour un label. Nommez deux erreurs différentes qui produisent exactement ce marqueur, et dites comment vous les distingueriez sans débogueur.
- Un collègue propose de stocker le total de commande en base de données sous la forme de la chaîne "1.850,75 €" pour que les utilisateurs allemands le voient toujours correctement. Expliquez ce qui casse, et quelle partie de cette chaîne est une donnée et quelle partie est de la présentation.

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 changez la formulation anglaise du bouton `Save` de "Save" à "Save customer". Quels fichiers doivent changer si votre clé est `CustomerEditor.Save`, et quels fichiers auraient dû changer si la clé avait été la phrase anglaise elle-même?
- ✓ Votre classe utilitaire `Texts` renvoie `[Preview.Amount]` pour un label. Nommez deux erreurs différentes qui produisent exactement ce marqueur, et dites comment vous les distingueriez sans débogueur.
- ✓ Un collègue propose de stocker le total de commande en base de données sous la forme de la chaîne "1.850,75 €" pour que les utilisateurs allemands le voient toujours correctement. Expliquez ce qui casse, et quelle partie de cette chaîne est une donnée et quelle partie est de la présentation.

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.

- ✓ **Ajoutez le deuxième cas.** *"Étends ce projet GlobalDesk 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/localization-in-wisej-net/specs/module-01.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."*



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.