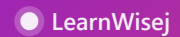


Exercice de code avec l'IA

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.



Écrire le code, c'est désormais la moitié facile.

Obtenir d'un assistant moderne du code qui fonctionne pour « Culture du navigateur, culture de session et changement de langue à l'exécution » ne demande qu'un bon prompt. Le juger, le corriger et l'étendre demande tout ce que ce module a couvert. C'est sur ce partage que repose l'exercice : vous apportez la spécification et la relecture, l'assistant apporte la frappe.

Ce que vous demandez

Donnez à GlobalDesk un changement de langue qui va jusqu'au bout. Mettez culture sur auto dans Default.json pour que la session démarre sur la langue du navigateur, ajoutez une ComboBox de langue proposant l'anglais, l'allemand et une culture langue-région, et définissez Application.CurrentCulture à partir de CultureInfo.GetCultureInfo dans son gestionnaire SelectedIndexChanged, sans jamais stocker le choix dans un champ statique. Gérez Application.CultureChanged pour appeler une méthode ApplyTextResources qui rafraîchit tous les libellés issus des ressources et une méthode UpdateCulturePreview qui remet en forme les exemples de date, de nombre et de montant, et recréez dans le même gestionnaire l'éditeur client localisé dans le designer. Vérifiez la même page via le paramètre d'URL lang, puis rédigez une note de lab décrivant ce qu'un champ de culture statique ferait subir à un deuxième utilisateur connecté au même moment.

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

- ✓ Default.json avec culture sur auto, accompagné d'une note sur les cas où une culture fixe est le bon choix
- ✓ Gestionnaire de la ComboBox de langue qui définit Application.CurrentCulture à partir de CultureInfo.GetCultureInfo
- ✓ Gestionnaire Application.CultureChanged qui appelle ApplyTextResources et UpdateCulturePreview
- ✓ Éditeur localisé dans le designer recréé après le changement pour qu'aucun libellé ne reste dans l'ancienne langue
- ✓ Vérification via le paramètre d'URL lang et note expliquant pourquoi la culture ne doit pas vivre dans un champ statique

Pourquoi il est conçu ainsi

Après le Module 3, GlobalDesk dispose d'un fichier de ressources neutre, d'un fichier allemand à côté, d'une classe utilitaire Texts, d'un éditeur client localisé dans le designer et même d'un libellé système Wisej qui lui est propre. Ce à quoi il n'a toujours pas de réponse, c'est la question que pose d'abord un vrai utilisateur: comment obtenir tout cela dans ma langue, et qu'arrive-t-il à l'écran quand je le demande? Tout ce qui précède a été décidé pendant la construction de l'application; ce module traite des deux moments où la décision se prend vraiment — au démarrage d'une session, et lorsqu'un utilisateur change d'avis en cours de route.

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
détection automatique du navigateur	Wisej.NET peut lire la langue annoncée par le navigateur et démarrer la session avec elle, de sorte que chaque utilisateur obtient une culture sensée sans qu'un réglage unique côté serveur décide pour tout le monde.
culture dans Default.json	Le paramètre d'application qui décide d'où vient la culture d'une nouvelle session; sa valeur par défaut est auto, et un nom fixe comme de-DE fige au contraire toutes les sessions sur une seule locale.
rightToLeft dans Default.json	Son paramètre voisin, dont la valeur auto suit la culture que la session finit par adopter; le Module 5 s'appuie dessus, mais vous le rencontrez ici, dans le même fichier.
Application.CurrentCulture	Obtient ou définit la culture de la session Wisej.NET en cours, ce qui est exactement la portée dont a besoin une application web multi-utilisateur: le choix d'un utilisateur ne doit jamais redessiner l'écran d'un autre.
paramètre d'URL lang	Une façon prise en charge de demander une langue à l'entrée, par exemple? lang=de, qui rend triviale la production de liens de test et d'URL d'entrée propres à une langue.
Application.CultureChanged	Déclenché quand la culture courante s'écarte de celle avec laquelle la session a démarré; l'endroit unique où réappliquer les textes issus des ressources et remettre en forme ce qui est déjà à l'écran.
contrôles localisés dans le designer	Leurs valeurs localisées sont appliquées à la création des contrôles: un changement de culture ultérieur ne les réécrit donc pas — vous recréez le conteneur ou réappliquez les valeurs dans le code.

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 4. Vous pouvez ouvrir vous-même [module-04.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 naviguer sur le web récupérera la spécification.



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.



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 4 - « Culture du navigateur, culture de session et changement de langue à l'exécution » - 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-04.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 : Donnez à GlobalDesk un changement de langue qui va jusqu'au bout. Mettez culture sur auto dans `Default.json` pour que la session démarre sur la langue du navigateur, ajoutez une `ComboBox` de langue proposant l'anglais, l'allemand et une culture langue-région, et définissez `Application.CurrentCulture` à partir de `CultureInfo.GetCultureInfo` dans son gestionnaire `SelectedIndexChanged`, sans jamais stocker le choix dans un champ statique. Gérez `Application.CultureChanged` pour appeler une méthode `ApplyTextResources` qui rafraîchit tous les libellés issus des ressources et une méthode `UpdateCulturePreview` qui remet en forme les exemples de date, de nombre et de montant, et recréez dans le même gestionnaire l'éditeur client localisé dans le designer. Vérifiez la même page via le paramètre d'URL `lang`, puis rédigez une note de lab décrivant ce qu'un champ de culture statique ferait subir à un deuxième utilisateur connecté au même moment.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - `Default.json` avec culture sur auto, accompagné d'une note sur les cas où une culture fixe est le bon choix
 - Gestionnaire de la `ComboBox` de langue qui définit `Application.CurrentCulture` à partir de `CultureInfo.GetCultureInfo`
 - Gestionnaire `Application.CultureChanged` qui appelle `ApplyTextResources` et `UpdateCulturePreview`
 - Éditeur localisé dans le designer recréé après le changement pour qu'aucun libellé ne reste dans l'ancienne langue
 - Vérification via le paramètre d'URL `lang` et note expliquant pourquoi la culture ne doit pas vivre dans un champ statique
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 :

- Votre méthode `ApplyTextResources` rafraîchit parfaitement le tableau de bord, et pourtant l'éditeur client garde ses libellés anglais après un passage à l'allemand. Expliquez ce qui distingue ces libellés, et nommez deux stratégies qui corrigent le problème.
- Un collègue propose de mettre le `CultureInfo` choisi en cache dans un champ statique « pour éviter de le rechercher deux fois ». Décrivez ce que verrait réellement un deuxième utilisateur, connecté depuis un autre pays au même moment.
- L'aperçu de la culture affiche un montant. Après un passage de en-US à de-DE, qu'est-ce qui a changé en mémoire et qu'est-ce qui n'a changé qu'à l'écran — et laquelle de ces deux choses écririez-vous un jour en base de données?

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 :

- ✓ Votre méthode `ApplyTextResources` rafraîchit parfaitement le tableau de bord, et pourtant l'éditeur client garde ses libellés anglais après un passage à l'allemand. Expliquez ce qui distingue ces libellés, et nommez deux stratégies qui corrigent le problème.
- ✓ Un collègue propose de mettre le `CultureInfo` choisi en cache dans un champ statique « pour éviter de le rechercher deux fois ». Décrivez ce que verrait réellement un deuxième utilisateur, connecté depuis un autre pays au même moment.
- ✓ L'aperçu de la culture affiche un montant. Après un passage de en-US à de-DE, qu'est-ce qui a changé en mémoire et qu'est-ce qui n'a changé qu'à l'écran — et laquelle de ces deux choses écririez-vous un jour en base de données?

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.

- ✓ **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."*
- ✓ **Faites-le résister aux mauvais jours.** *"Examine ce projet GlobalDesk 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 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."*



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.