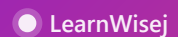


Exercice de code avec l'IA

Construisez l'exemple AdaptiveOps 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.



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

Obtenir d'un assistant moderne du code qui fonctionne pour « Fondamentaux de la mise en page : docking, ancrage, AutoSize, AutoScroll et composition du shell » 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

Refactorisez le shell d'Adaptive Operations Console en conteneurs imbriqués : une barre d'outils en haut, un rail de navigation à gauche, un espace de travail en remplissage, un panneau de détails à droite et une région d'état, chacun étant un Panel ou un UserControl avec sa propre mise en page locale. Supprimez tous les gestionnaires Resize qui définissent Bounds, corrigez l'ordre de docking dans le designer, utilisez Padding pour les espacements entre les régions ancrées, donnez un MinimumSize à l'espace de travail et à la grille, et faites défiler l'éditeur de détails avec AutoScroll et des barres de défilement masquées aux largeurs étroites.

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

- ✓ Shell reconstruit à partir de conteneurs imbriqués avec les régions barre d'outils, navigation, espace de travail, détails et état
- ✓ Tout le code Resize manuel qui définissait Bounds supprimé
- ✓ Ordre de docking, Padding et MinimumSize définis intentionnellement sur chaque région
- ✓ Éditeur de détails défilant via AutoScroll avec barres de défilement masquées aux largeurs étroites
- ✓ Régions de navigation et de détails extraites dans des UserControls

Pourquoi il est conçu ainsi

Adaptive Operations Console dispose désormais d'un thème et de CSS ciblé, mais son shell déplace encore les contrôles à la main: un gestionnaire Resize recalcule Bounds pour la barre d'outils, le rail de navigation, la grille des tickets et l'éditeur de détails chaque fois que le navigateur change de taille. Cela fonctionne à la largeur testée par le développeur, casse à toutes les autres, et ne peut pas survivre aux profils tablette et téléphone qu'introduit le Module 6. Dans Wisej.NET, la conception adaptative repose sur des conteneurs et

des règles de mise en page, pas sur des coordonnées: des régions qui occupent des bords, un espace de travail qui remplit ce qui reste, et des éditeurs qui défilent au lieu de se chevaucher.

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
moteur de mise en page par défaut	Chaque contrôle Wisej.NET a un moteur de mise en page; celui par défaut prend en charge la position absolue, le docking et l'ancrage, si bien qu'un écran peut combiner un placement précis dans le designer et des régions qui se redimensionnent d'elles-mêmes.
priorité de docking	Les enfants ancrés consomment l'espace dans l'ordre où ils apparaissent dans la collection de contrôles du parent; c'est pourquoi les cinq mêmes panneaux produisent un shell correct ou un shell qui se chevauche selon l'ordre des enfants.
DisplayRectangle et Padding	Les contrôles ancrés se disposent dans le DisplayRectangle du parent, que Padding réduit; le moteur par défaut ignore Margin pour les enfants ancrés, donc les espacements viennent du padding ou de contrôles d'espacement.
ancrage	Un ancrage conserve la distance d'un contrôle aux côtés choisis du parent, ce qui convient aux régions de type formulaire: labels en haut à gauche, zones de texte gauche-droite, boutons de commande en bas à droite.
AutoSize et mise en page circulaire	Un conteneur qui se dimensionne d'après des enfants dockés ou ancrés sur lui n'a pas de réponse stable; cette boucle est la principale mise en garde du module concernant AutoSize.
AutoScroll avec barres masquées	AutoScroll fait défiler un conteneur quand le contenu ne tient pas, et ScrollBars.Hidden conserve le défilement tactile sur téléphone sans dessiner de barres de défilement.
région UserControl	Une région du shell avec sa propre mise en page locale et une petite surface publique, pour que le shell affiche, masque, ancre ou déplace la région entière selon le profil, sans connaître ses coordonnées internes.

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



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 AdaptiveOps 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 – « Fondamentaux de la mise en page : docking, ancrage, AutoSize, AutoScroll et composition du shell » – du cours LearnWisej « Thèmes et mises en page responsives », 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/theming-and-responsive-layouts/specs/module-04.md>

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

1. Projet « AdaptiveOps », 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 : Refactorisez le shell d'Adaptive Operations Console en conteneurs imbriqués : une barre d'outils en haut, un rail de navigation à gauche, un espace de travail en remplissage, un panneau de détails à droite et une région d'état, chacun étant un Panel ou un UserControl avec sa propre mise en page locale. Supprimez tous les gestionnaires Resize qui définissent Bounds, corrigez l'ordre de docking dans le designer, utilisez Padding pour les espacements entre les régions ancrées, donnez un MinimumSize à l'espace de travail et à la grille, et faites défiler l'éditeur de détails avec AutoScroll et des barres de défilement masquées aux largeurs étroites.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Shell reconstruit à partir de conteneurs imbriqués avec les régions barre d'outils, navigation, espace de travail, détails et état
 - Tout le code Resize manuel qui définissait Bounds supprimé
 - Ordre de docking, Padding et MinimumSize définis intentionnellement sur chaque région
 - Éditeur de détails défilant via AutoScroll avec barres de défilement masquées aux largeurs étroites
 - Régions de navigation et de détails extraites dans des UserControls
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 :

- Si le panneau d'espace de travail était ancré avant le panneau d'état dans l'ordre des enfants, que verrait l'utilisateur en bas de la grille, et quelle commande du designer corrige le problème sans changer une seule taille?
- Vous avez défini une Margin de 12 pixels sur le rail de navigation et l'espace n'est pas apparu. Pourquoi, et quels sont les deux mécanismes qui produisent bien un espace entre des

régions ancrées?

– L'éditeur de détails est un UserControl avec AutoScroll. Quels membres le shell doit-il connaître pour le déplacer sous la grille sur tablette, et lesquels ne doit-il jamais avoir besoin de connaître?

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 :

- ✓ Si le panneau d'espace de travail était ancré avant le panneau d'état dans l'ordre des enfants, que verrait l'utilisateur en bas de la grille, et quelle commande du designer corrige le problème sans changer une seule taille?
- ✓ Vous avez défini une Margin de 12 pixels sur le rail de navigation et l'espace n'est pas apparu. Pourquoi, et quels sont les deux mécanismes qui produisent bien un espace entre des régions ancrées?
- ✓ L'éditeur de détails est un UserControl avec AutoScroll. Quels membres le shell doit-il connaître pour le déplacer sous la grille sur tablette, et lesquels ne doit-il jamais avoir besoin de connaître?

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 4 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 4 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Theming and Responsive Layouts Course/Module 4`.



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 AdaptiveOps 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 AdaptiveOps 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 AdaptiveOps 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.