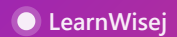


Exercice de code avec l'IA

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.



Briefez-le, lisez-le, puis cassez-le.

Cet exercice comporte trois parties, et seule la première concerne le prompt. Vous briefez un assistant avec la spécification du module et récupérez une solution AdaptiveOps ; vous lisez ce qu'il a écrit et décidez s'il fait vraiment ce que le module demandait ; puis vous le poussez plus loin que ce que le module demandait. C'est dans la deuxième et la troisième partie que l'on apprend.

Ce que vous demandez

Ajoutez à Adaptive Operations Console une feuille de style AdaptiveOps.css à portée limitée, avec des classes pour les cartes de métriques et les badges compacts, appliquez-les via CssClass, et n'utilisez CssStyle que pour une seule valeur calculée, par exemple une largeur de progression. Attribuez l'AppearanceKey action-button à la commande principale, ajoutez un état personnalisé stale aux lignes de la grille des tickets ou à un panneau d'état, et implémentez dans la barre d'outils un basculement de thème clair/sombre qui change le thème pour la session courante uniquement, sans modifier le thème partagé.

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

- ✓ AdaptiveOps.css à portée limitée avec les classes metric-card et compact-badge appliquées via CssClass
- ✓ CssStyle utilisé pour exactement une valeur ponctuelle calculée, avec la raison documentée
- ✓ Commande principale utilisant l'AppearanceKey action-button
- ✓ État personnalisé stale ajouté via States et stylé dans le thème
- ✓ Basculement de thème clair/sombre limité à la session qui laisse le thème partagé intact

Pourquoi il est conçu ainsi

Le thème AdaptiveOps du Module 2 porte désormais l'identité de chaque contrôle standard d'Adaptive Operations Console, et les demandes qui arrivent ensuite sont toutes petites: une carte de métrique a besoin d'une surface arrondie et surélevée, un badge SLA doit être compact, une barre de progression a besoin d'une largeur calculée à partir de données en direct, la commande Assign doit ressembler à l'action principale, et l'équipe de nuit veut une variante sombre. Chaque demande a son raccourci tentant — une chaîne CssStyle ici, un BackColor là, le thème partagé modifié sur place — et chaque raccourci déplace

discrètement une décision visuelle vers une couche qui ne la porte pas. Six mois plus tard, personne ne sait dire ce que rend le thème, ce que le CSS écrase et ce que le code rapièce. Ce module place chacune de ces cinq demandes dans le mécanisme qui la porte.

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
CssClass	Un ou plusieurs noms de classe ajoutés à l’élément principal du widget dans le navigateur: le point d’accroche réutilisable qu’une feuille de style applicative ciblée peut viser sans toucher au thème.
CssStyle	Une chaîne CSS écrite directement comme attribut style de l’élément principal du widget; adaptée à une seule valeur calculée, inadaptée comme outil de style général, car les styles inline sont difficiles à inspecter, à intégrer au thème ou à confier à un designer.
feuille de style applicative ciblée	Un fichier CSS chargé à côté du thème dont les sélecteurs partent de noms de classe qui vous appartiennent, comme <code>metric-card</code> , et n’atteignent jamais le DOM interne d’un widget Wisej standard.
extender historique StyleSheet	Un ancien composant qui ajoute des classes CSS aux contrôles éligibles et héberge des règles dans sa propriété <code>StyleSheetSource</code> ; une aide à la migration, inutile dans le nouveau code puisque chaque contrôle dispose désormais de <code>CssClass</code> et de <code>CssStyle</code> .
AppearanceKey	Le nom d’une appearance du thème actif avec laquelle un contrôle s’affiche, pour que le <code>C#</code> exprime l’intention (<code>action-button</code>) et que le thème porte les couleurs, polices et métriques de chaque état.
états personnalisés	Des noms ajoutés à la collection <code>States</code> d’un contrôle, que le thème peut styler comme <code>hovered</code> ou <code>focused</code> , pour des conditions d’interface transitoires comme <code>stale</code> , <code>warning</code> ou <code>dirty</code> qu’un designer doit pouvoir voir et restyler.
thème de démarrage ou <code>Application.Theme</code>	Le thème global de l’application est choisi avant son exécution, dans <code>Web.config</code> ou <code>Default.json</code> ; <code>Application.Theme</code> permet à une session de travailler avec un objet de thème à l’exécution sans changer ce que voient les autres sessions.

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 3. Vous pouvez ouvrir vous-même [module-03.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 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 3 – « CSS, CssClass, CssStyle, états et changements de thème à l'exécution » – 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-03.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 : Ajoutez à Adaptive Operations Console une feuille de style `AdaptiveOps.css` à portée limitée, avec des classes pour les cartes de métriques et les badges compacts, appliquez-les via `CssClass`, et n'utilisez `CssStyle` que pour une seule valeur calculée, par exemple une largeur de progression. Attribuez l'`AppearanceKey` `action-button` à la commande principale, ajoutez un état personnalisé `stale` aux lignes de la grille des tickets ou à un panneau d'état, et implémentez dans la barre d'outils un basculement de thème clair/sombre qui change le thème pour la session courante uniquement, sans modifier le thème partagé.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - `AdaptiveOps.css` à portée limitée avec les classes `metric-card` et `compact-badge` appliquées via `CssClass`
 - `CssStyle` utilisé pour exactement une valeur ponctuelle calculée, avec la raison documentée
 - Commande principale utilisant l'`AppearanceKey` `action-button`
 - État personnalisé `stale` ajouté via `States` et stylé dans le thème
 - Basculement de thème clair/sombre limité à la session qui laisse le thème partagé intact
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 designer demande un rayon d'angle plus petit sur toutes les cartes de métriques. Quel fichier change, et combien d'endroits auriez-vous dû modifier si le rayon avait été défini avec `CssStyle`?
- Pourquoi le bouton `Assign` n'a-t-il besoin d'aucun code de survol ou d'appui, et que perdriez-vous exactement en remplaçant son `AppearanceKey` par des valeurs `BackColor` et

ForeColor correspondant à la palette?

– Deux personnes sont connectées et l'une d'elles passe en mode sombre. Décrivez ce que voit chacune, et désignez la ligne de votre code qui le garantit.

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 designer demande un rayon d'angle plus petit sur toutes les cartes de métriques. Quel fichier change, et combien d'endroits auriez-vous dû modifier si le rayon avait été défini avec `CssStyle`?
- ✓ Pourquoi le bouton Assign n'a-t-il besoin d'aucun code de survol ou d'appui, et que perdriez-vous exactement en remplaçant son `AppearanceKey` par des valeurs `BackColor` et `ForeColor` correspondant à la palette?
- ✓ Deux personnes sont connectées et l'une d'elles passe en mode sombre. Décrivez ce que voit chacune, et désignez la ligne de votre code qui le garantit.

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 3 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 3 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Theming and Responsive Layouts Course/Module 3`.



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.

- ✓ **Relisez-le comme une pull request.** *"Fais-moi parcourir ce code AdaptiveOps 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 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 gèrerait 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."*



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.