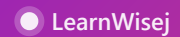


Exercice de code avec l'IA

Construisez l'exemple IconDesk 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.



Le même module, construit comme vous construirez vraiment.

Toutes les autres sections de ce module expliquent comment fonctionne « Ressources embarquées et resource.wx ». Celle-ci vous demande de le produire — en pilotant un assistant avec la spécification du module elle-même, puis en vérifiant que le résultat s'exécute et fait ce que décrit l'énoncé du lab. C'est la méthode de travail que vous utiliserez sur de vrais projets, mise en pratique sur un sujet dont vous connaissez déjà la bonne réponse.

Ce que vous demandez

Ajoutez à l'application IconDesk un dossier Resources contenant logo.svg, status-ok.svg, status-warning.svg, photo.png et badge.gif, et passez chacun d'eux en Build Action Embedded Resource. Construisez une page ResourceLab qui les affiche tous les cinq via des sources d'image resource.wx, écrivez l'une de ces sources en plaçant le nom de l'assembly de l'application devant le fichier pour que la résolution ne puisse pas être ambiguë, et laissez les autres non qualifiées. Déposez ensuite un fichier de remplacement à côté de l'application déployée pour remplacer une ressource embarquée, et capturez l'avant et l'après. Terminez en écrivant délibérément une URL de ressource mal orthographiée et en notant les étapes qui ont permis d'identifier la panne.

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

- ✓ Dossier Resources avec logo.svg, status-ok.svg, status-warning.svg, photo.png et badge.gif tous marqués Embedded Resource
- ✓ Page ResourceLab affichant chaque ressource embarquée via une source d'image resource.wx
- ✓ Une source d'image qualifiée par le nom de l'assembly de l'application et les autres laissées non qualifiées
- ✓ Fichier déployé qui remplace une ressource embarquée, capturé avant et après
- ✓ Note de dépannage listant les étapes qui ont permis d'identifier une URL de ressource mal orthographiée

Pourquoi il est conçu ainsi

Jusqu'ici, chaque ressource d'IconDesk est quelque chose que l'application a trouvé sur le disque ou demandé au thème. Cela fonctionne jusqu'à ce que quelqu'un déploie le site et oublie un dossier: le navigateur demande une icône, le serveur répond 404, et l'utilisateur voit un rectangle vide là où devrait se

trouver le badge d'état. Personne ne le détecte en revue, parce que la machine du développeur a toujours le fichier.

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
Build Action = Embedded Resource	La propriété de fichier de Visual Studio qui compile un fichier dans le manifeste de l'assembly au lieu de le copier à côté de l'exécutable, si bien que la ressource est livrée dans la DLL.
gestionnaire resource.wx	Le gestionnaire HTTP de Wisej.NET qui sert les ressources embarquées au navigateur; ce que vous affectez est une chaîne de source d'image, jamais un objet image côté serveur.
URL de ressource non qualifiée	resource.wx/company-logo.svg demande à Wisej.NET de chercher dans les assemblies chargés une ressource portant ce nom, ce qui est pratique et légèrement ambigu.
URL de ressource qualifiée par l'assembly	resource.wx/MyCompany.Branding/company-logo.svg nomme l'assembly (sans.dll) pour que la résolution ne puisse pas tomber sur le fichier homonyme de quelqu'un d'autre.
types de ressources autorisés	Le gestionnaire documente exactement neuf extensions:.css,.js,.html,.jpg,.jpeg,.png,.gif,.svg et.bmp; tout le reste revient en 404.
remplacement par fichier déployé	Un fichier déployé avec l'application peut être servi à la place de la ressource embarquée au même chemin, ce qui permet à une installation de rehabiller ou de corriger un visuel sans recompiler l'assembly.
nom de ressource de manifeste	Le nom que.NET stocke réellement, dérivé du nom de l'assembly, du namespace par défaut et du dossier où se trouve le fichier; le deviner est la cause habituelle d'une URL de ressource qui ne renvoie rien.

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 5. Vous pouvez ouvrir vous-même [module-05.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.

naviguer 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 IconDesk 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 5 - « Ressources embarquées et resource.wx » - du cours LearnWisej « Icônes et images 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/icons-and-images/specs/module-05.md>

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

1. Projet « IconDesk », 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 l'application IconDesk un dossier Ressources contenant `logo.svg`, `status-ok.svg`, `status-warning.svg`, `photo.png` et `badge.gif`, et passez chacun d'eux en Build Action Embedded Resource. Construisez une page `ResourceLab` qui les affiche tous les cinq via des sources d'image `resource.wx`, écrivez l'une de ces sources en plaçant le nom de l'assembly de l'application devant le fichier pour que la résolution ne puisse pas être ambiguë, et laissez les autres non qualifiées. Déposez ensuite un fichier de remplacement à côté de l'application déployée pour remplacer une ressource embarquée, et capturez l'avant et l'après. Terminez en écrivant délibérément une URL de ressource mal orthographiée et en notant les étapes qui ont permis d'identifier la panne.
3. Produis ces livrables et indique-moi où se trouve chacun d'eux dans la solution :
 - Dossier Ressources avec `logo.svg`, `status-ok.svg`, `status-warning.svg`, `photo.png` et `badge.gif` tous marqués Embedded Resource
 - Page `ResourceLab` affichant chaque ressource embarquée via une source d'image `resource.wx`
 - Une source d'image qualifiée par le nom de l'assembly de l'application et les autres laissées non qualifiées
 - Fichier déployé qui remplace une ressource embarquée, capturé avant et après
 - Note de dépannage listant les étapes qui ont permis d'identifier une URL de ressource mal orthographiée
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 :

- Deux assemblies référencés embarquent tous deux un fichier nommé `logo.svg`. Vers quoi une source non qualifiée se résout-elle, et que changeriez-vous pour que la réponse cesse de

dépendre de l'ordre de chargement?

- Votre remplacement fonctionne sur le serveur de test mais plus après la version suivante. Quelles parties du processus de déploiement ont pu le supprimer, et où le remplacement aurait-il dû être consigné?
- Un coéquipier embarque `icon.ico` et signale que `resource.wx` est cassé. Que se passe-t-il réellement, et quelle est la plus petite correction possible?

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 :

- ✓ Deux assemblés référencés embarquent tous deux un fichier nommé `logo.svg`. Vers quoi une source non qualifiée se résout-elle, et que changeriez-vous pour que la réponse cesse de dépendre de l'ordre de chargement?
- ✓ Votre remplacement fonctionne sur le serveur de test mais plus après la version suivante. Quelles parties du processus de déploiement ont pu le supprimer, et où le remplacement aurait-il dû être consigné?
- ✓ Un coéquipier embarque `icon.ico` et signale que `resource.wx` est cassé. Que se passe-t-il réellement, et quelle est la plus petite correction possible?

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 IconDesk 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/icons-and-images/specs/module-05.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 IconDesk 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.