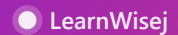


Exercice de code avec l'IA

Construisez le Sensor Monitor de ce module avec ChatGPT ou Claude à partir d'un prompt prêt à l'emploi qui oriente le modèle vers le guide de ce module, puis relisez, exécutez et étendez ce qu'il vous fournit — y compris en le comparant à notre propre build de référence sur GitHub.



Vous n'allez pas taper cet exemple. Vous allez le diriger.

Presque plus personne n'écrit de code d'intégration à partir d'un fichier vide — un assistant rédige la première version et le développeur décide si elle est juste. Cela ne fonctionne que si vous maîtrisez le vocabulaire, le contrat et les règles du framework, c'est-à-dire exactement ce que le reste de ce module vous a appris. Dans cet exercice, vous transmettez ces connaissances à ChatGPT ou à Claude, vous récupérez l'exemple du Module 1 sous forme de solution Visual Studio fonctionnelle, puis vous le relisez comme vous reliriez la pull request d'un collègue.

Ce que vous allez en retirer

L'exemple du Module 1 est un **Sensor Monitor** : un `TemperatureGauge : Widget` qui héberge une bibliothèque de jauge tierce dans un conteneur de widget Wisej.NET, avec `Value` comme propriété serveur validée et `ThresholdExceeded` comme événement .NET ordinaire. À côté de la jauge se trouve une **trace Serveur ⇌ Client** en direct, afin que chaque message qui transite sur le réseau puisse être confronté au contrat que vous avez écrit.

Le construire par prompt prend un quart d'heure. L'intérêt n'est pas la saisie que vous économisez — ce sont les trois choses que vous devez savoir faire ensuite :

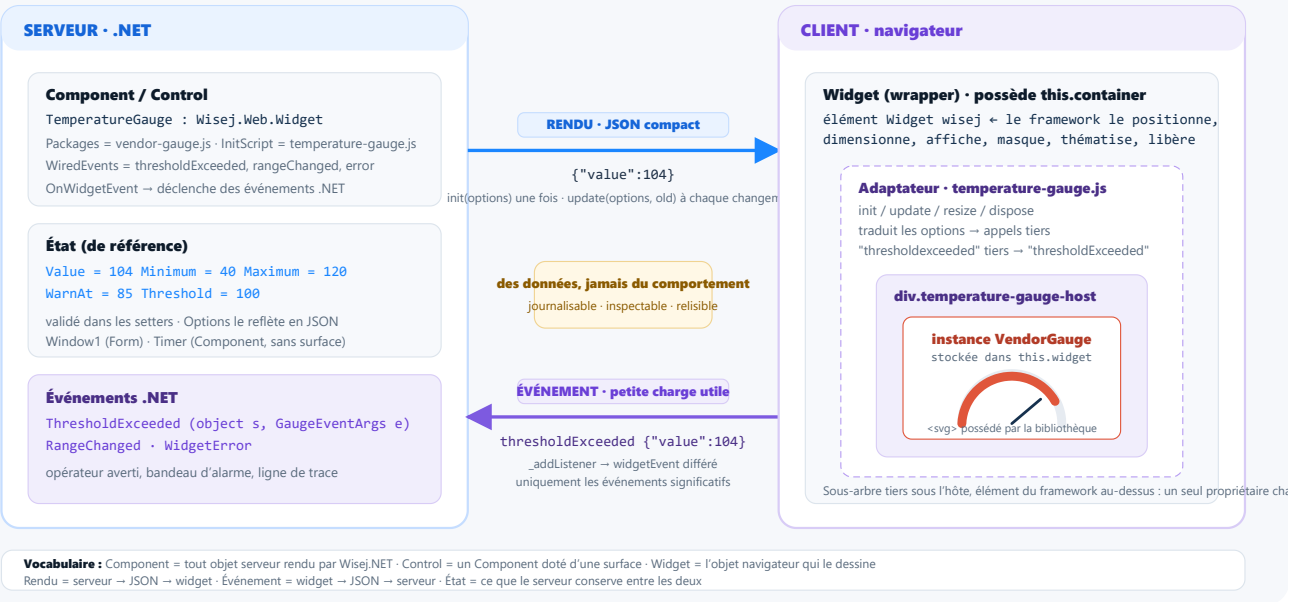
- ✓ **Briefer** l'assistant avec le contrat et les règles du framework, pour que la première version utilise déjà `Wisej.Web.Timer` et héberge la bibliothèque tierce dans un élément enfant de `this.container`.
- ✓ **Vérifier** ce qui vous revient. Un code qui compile n'est pas encore un code qui survit à un vrai chargement de page.
- ✓ **Étendre** du code que vous n'avez pas écrit, ce à quoi ressemble désormais le reste de votre carrière.

L'architecture que vous demandez

Avant de rédiger le prompt, soyez capable de le dire d'une traite : *le serveur possède l'état de l'application, le navigateur possède le DOM et l'instance de la bibliothèque tierce, et seules des données circulent entre les deux* — du JSON compact à l'aller lors du rendu, de petites charges utiles au retour avec les événements. La valeur de la jauge, ses limites et son seuil sont de l'état serveur, car ils doivent être auditables. L'animation de l'aiguille, non. Quand l'assistant vous livre quelque chose qui garde la valeur de référence en JavaScript, vous savez déjà que c'est faux.

IntegrationLab · Architecture de l'intégration (Module 1)

Le serveur possède l'état. Le navigateur possède le DOM et l'instance tierce. Seules des données transitent sur le réseau.



Terme de la leçon	Ce que l'assistant doit produire
Component / Control	TemperatureGauge côté serveur, qui possède Value , Minimum , Maximum , WarnAt , Threshold . Le Wisej.Web.Timer qui diffuse les relevés est un Component sans surface.
Widget	wwwroot/temperature-gauge.js — l'adaptateur côté navigateur qui crée l'instance tierce dans un élément enfant de this.container et la conserve dans this.widget .
Package	wwwroot/vendor-gauge.js , la bibliothèque tierce de substitution, chargée une seule fois par page.
Rendu	Options → JSON compact → init(options) au premier rendu, update(options, old) pour les champs de premier niveau modifiés.
Événement	_addListener / _getEventData → OnWidgetEvent → ThresholdExceeded , RangeChanged , WidgetError .

Construisez-le — choisissez votre assistant

Les deux boutons ouvrent un nouvel onglet avec le même prompt prêt à l'emploi déjà saisi. 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 à partir du 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 va lire. Lisez le prompt avant de l'envoyer : c'est la partie de cet exercice qui mérite d'être apprise.



Utiliser ChatGPT



Utiliser Claude

Ouvre chatgpt.com avec le prompt du Module 1 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 chargé. Les utilisateurs de Claude Code peuvent 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 la solution complète sous forme d'un unique [.zip](#) , accompagné des étapes de compilation et d'exécution. Téléchargez-le, ouvrez [IntegrationLab.slnx](#) dans Visual Studio, appuyez sur F5 et le Sensor Monitor s'affiche sur [localhost](#) . Il vous faut le SDK .NET et le package NuGet [Wisej-4](#) ; rien d'autre n'est téléchargé, car le prompt demande à l'assistant d'écrire lui-même la bibliothèque de jauge « tierce ».

Je suis le Module 1 – « Architecture et vocabulaire de l'intégration » – du « Cours d'intégration d'applications » de LearnWisej, et je veux construire avec toi l'exemple du 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/app-integration/specs/module-01.md>

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

1. Projet "IntegrationLab", C#, net8.0 ou ultérieur (la build de référence utilise net10.0), package NuGet Wisej-4 4.1.x, hôte Kestrel qui appelle app.UseWisej().
2. Un TemperatureGauge : Wisej.Web.Widget qui héberge une bibliothèque de jauge tierce. Fournis toi-même la bibliothèque « tierce » sous la forme wwwroot/vendor-gauge.js et enregistre-la comme Package, pour que rien ne doive être téléchargé.
3. Le serveur possède l'état : Value, Minimum, Maximum, WarnAt, Threshold. Valide-les dans les setters des propriétés et reflète chacune dans Options comme champ de premier niveau, pour que Wisej.NET ne rende que ce qui a changé.
4. Un adaptateur client wwwroot/temperature-gauge.js, incorporé dans l'assembly et affecté à InitScript. Il doit créer l'instance tierce dans un élément ENFANT de this.container, la conserver dans this.widget et câbler les événements via _addListener / _getEventData – n'appelle jamais fireWidgetEvent de façon synchrone depuis update(), et ne définis jamais un membre qui masque une méthode qooxdoo comme getWidth.
5. WiredEvents = thresholdExceeded, rangeChanged, error, exposés côté serveur comme des événements .NET ordinaires avec des EventArgs typés qui ne transportent que des données.
6. Une Window1 "Sensor Monitor" avec la jauge à gauche et un journal de trace Server-to-Client en direct à droite, plus des boutons qui exercent cinq chemins : un relevé valide, une série diffusée pilotée par un Wisej.Web.Timer, une valeur hors plage rejetée, une charge utile volontairement corrompue que l'adaptateur doit intercepter et signaler par un seul événement error, et une resynchronisation qui refait le rendu de l'état de référence du serveur.
7. Un dossier docs/ avec ClientServerContract.md, WidgetTriage.md, IntegrationDecisionRecord.md et un schéma d'architecture.

Écris docs/ClientServerContract.md avant d'écrire le moindre code, et montre-le-moi d'abord.

Quand le code est terminé, empaquette toute la solution dans un unique .zip que je puisse télécharger, et donne-moi les étapes exactes pour la compiler et l'exécuter.

Pose-moi des questions sur tout ce qui est ambigu avant de commencer.

Relisez ensuite ce qui vous revient

Les erreurs qui coûtent du temps ici sont celles que le compilateur ne peut pas détecter : un code qui compile proprement puis se comporte mal lors d'un vrai chargement de page. Ces deux-là nous ont coûté une séance de débogage quand nous avons construit l'exemple de référence, et ce sont les premières choses à vérifier dans le vôtre :

- ✓ L'adaptateur appelait `fireWidgetEvent` de façon synchrone depuis un callback de la bibliothèque tierce. Les événements déclenchés pendant qu'une mise à jour du serveur est

appliquée sont perdus — le câblage doit passer par `_addListener` / `_getEventData` .

- ✓ L'adaptateur définissait `this.getWidth` , masquant une méthode qooxdoo. Le widget s'affichait avec une largeur de zéro pixel jusqu'à ce que la fonction utilitaire soit renommée.

Exécutez ensuite les cinq chemins demandés par le prompt — relevé valide, série diffusée, valeur rejetée, charge utile corrompue, resynchronisation — et comparez chaque ligne de trace avec `docs/ClientServerContract.md` . Si un message transite sur le réseau sans être décrit par le contrat, c'est que le contrat ou le code est faux, et c'est à vous de décider lequel.



Nous avons déjà construit celui-ci — comparez le vôtre avec le nôtre

L'implémentation de référence du Module 1 se trouve dans le dépôt public des exemples, avec les livrables `docs/` , la classe serveur, l'adaptateur et la fenêtre de trace. Construisez d'abord le vôtre, puis ouvrez le nôtre : ce sont les différences qui sont intéressantes.



[Ouvrir l'exemple du Module 1 sur GitHub](#)

Ou clonez l'ensemble avec `git clone https://github.com/iceteagroup/LearnWisej-Samples` et ouvrez `Application Integration Course/Module 1/IntegrationLab.slnx`.



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. La jauge tierce est un substitut, il n'y a pas de tests unitaires, la prise en charge du Designer est minimale et rien n'est durci pour la production. Voyez-y une invitation : utilisez ce que vous avez appris dans ce module, avec l'aide de l'assistant, pour aller plus loin. Améliorer du code que vous n'avez pas écrit, c'est la compétence que ce cours enseigne réellement.

Trois prompts pour aller plus loin avec l'exemple

Orientez votre assistant vers le dossier cloné (ou collez-lui les fichiers concernés) et essayez ceux-ci. Chacun est un exercice de relecture et d'extension, pas une réécriture.

- ✓ **Remplacez le substitut par une vraie bibliothèque.** « Dans ce projet `IntegrationLab`, remplace `wwwroot/vendor-gauge.js` par une vraie jauge tierce — par exemple une radial bar ApexCharts, ou une jauge canvas chargée comme Package. Ne modifie pas `docs/ClientServerContract.md` : le contrat serveur, les champs `Options` et les trois événements câblés doivent rester exactement tels quels, et seul l'adaptateur peut changer. Montre-moi le diff, et dis-moi ce que la vraie bibliothèque t'a obligé à gérer que le substitut n'exigeait pas. »
- ✓ **Ajoutez un second widget en réutilisant le pattern.** « Ajoute un `HumidityGauge : Widget` à ce projet en suivant la même architecture que `TemperatureGauge` : état validé côté serveur et reflété dans `Options`, adaptateur qui héberge la bibliothèque tierce dans un élément enfant, événements câblés via `_addListener` / `_getEventData`. Place les deux jauges dans la fenêtre `Sensor Monitor` avec un journal de trace commun, et extrais ce qui est réellement commun dans une classe de base — mais seulement si elle se justifie. Mets à jour le document de contrat pour couvrir les deux. »
- ✓ **Rendez-le résistant aux mauvais jours.** « Passe en revue ce projet pour repérer les modes de défaillance qu'il ne gère pas encore : le script tiers qui ne se charge pas, un événement qui arrive après la libération du widget, deux changements rapides de `Value` en concurrence avec le premier rendu, et un navigateur sans `ResizeObserver`. Pour chacun, dis-moi d'abord s'il s'agit d'un risque réel dans Wisej.NET 4 et pourquoi — et seulement ensuite écris le correctif. Ajoute une entrée dans `docs/IntegrationDecisionRecord.md` pour tout ce que tu as décidé de laisser en l'état. »



Emportez-le dans le lab

Le lab qui clôt ce module demande un schéma d'architecture, un tableau de tri des widgets et un registre des décisions d'intégration pour trois widgets candidats. Ce que vous construisez ici — votre version ou la nôtre — contient déjà un exemple concret des trois dans `docs/`. Servez-vous-en comme modèle de réponse, pas comme réponse.