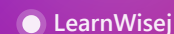


Ce qu'est Wisej.NET (et quand l'utiliser)

Une vision claire du framework avant d'écrire la moindre ligne de code.



Objectif de la leçon

À la fin de cette leçon, vous saurez expliquer ce qu'est Wisej.NET, décrire son modèle mental et déterminer quand c'est l'outil adapté à un projet.

1. Ce qu'est Wisej.NET

Wisej.NET est un framework et un environnement de développement pour créer des **applications web temps réel** avec les langages .NET et le flux de travail Visual Studio que de nombreux développeurs desktop connaissent déjà. Une application Wisej.NET s'exécute dans un navigateur, mais la logique applicative est écrite pour l'essentiel **en C# ou VB.NET côté serveur**.

Un premier modèle mental utile : **avec Wisej.NET, on a l'impression de construire une application de type desktop, mais l'application terminée est livrée par le navigateur web**. Au lieu d'écrire à la main chaque morceau de HTML, de CSS et de JavaScript, vous travaillez avec les contrôles, les propriétés et les événements de Wisej.NET et avec du code côté serveur.



Définition simple

Wisej.NET permet à un développeur .NET de créer des applications web avec des outils de conception visuelle, des contrôles et du code C# piloté par les événements — au lieu de partir d'un framework JavaScript front-end.

2. Comment ça fonctionne

Dans une application Wisej.NET classique, le navigateur affiche l'interface utilisateur tandis que le serveur gère les objets de l'application et la logique des événements. Quand un utilisateur clique sur un bouton, modifie un champ ou sélectionne une ligne, l'événement est traité par du code C# sur le serveur. Wisej.NET garde le navigateur et le serveur synchronisés pour que l'interface se mette à jour sans accroc.

Élément	Ce que cela signifie pour un débutant
Navigateur	L'endroit où l'utilisateur voit l'application et interagit avec elle — pages, boutons, libellés, grilles et formulaires.
Serveur	Exécute le code de l'application .NET. Les clics sur les boutons et la plupart des événements sont traités ici.
Designer	La surface visuelle de Visual Studio où vous disposez les contrôles et définissez leurs propriétés.
Contrôles	Des blocs d'interface réutilisables : <code>Button</code> , <code>Label</code> , <code>TextBox</code> , <code>DataGridView</code> , <code>Panel</code> , <code>Page</code> , <code>Window</code> .
Événements	Ce qui se produit — <code>Click</code> , <code>TextChanged</code> , <code>SelectedIndexChanged</code> — et déclenche votre code.

3. Quand utiliser Wisej.NET

Wisej.NET excelle lorsque le projet est une **application métier** et que l'équipe veut développer vite avec .NET, Visual Studio et un designer visuel — surtout quand l'application est riche en formulaires, en données, ou proche d'une application desktop.

Cas adaptés	Pourquoi Wisej.NET aide
Outils métier internes	Tableaux de bord, formulaires, écrans d'administration et outils de workflow, construits rapidement.
Applications CRUD	Les contrôles correspondent naturellement à la création, la lecture, la mise à jour et la suppression d'enregistrements.
Applications centrées sur la DataGridView	Tableaux de données, listes d'enregistrements et vues de détail sont des patterns fondamentaux de Wisej.NET.
Modernisation d'applications desktop	L'expérience de type WinForms est conservée lors du passage au web.
Prototypes rapides	Le Designer et le modèle événementiel produisent rapidement des écrans fonctionnels.

4. Quand *ne pas* l'utiliser

- Utilisez un site web traditionnel ou un CMS pour des pages de contenu surtout publiques et orientées SEO.
- Utilisez un framework front-end lorsque vous avez précisément besoin d'une architecture de composants JavaScript sur mesure.
- Utilisez un générateur de sites statiques lorsqu'il n'y a ni véritable workflow métier ni interaction avec une base de données.
- Utilisez le développement mobile natif lorsque l'intégration poussée avec l'appareil et la distribution via les stores d'applications sont l'objectif principal.

5. Le modèle mental de Wisej.NET

Raisonnez en termes d'**écrans**, **de contrôles**, **de propriétés** et **d'événements**. Un écran contient des contrôles. Les contrôles ont des noms et des propriétés. Les utilisateurs interagissent avec les contrôles. Les événements exécutent du code — et ce code peut mettre à jour l'interface, appeler un service, enregistrer des données ou naviguer.

Exemple de flux

```
L'utilisateur clique sur  
→ l'événement btnSave_Click s'exécute  
→ il lit les valeurs des zones de texte  
→ il appelle une méthode de service  
→ le service enregistre les données  
→ l'interface affiche un message d'état
```



Bonne habitude de débutant

Ne mettez pas toute votre logique métier dans le clic du bouton. Gardez le gestionnaire du clic lisible et appelez une méthode utilitaire, un service ou une classe repository dès que la logique grossit.

Vocabulaire clé

- **Projet** — l'application entière dans Visual Studio : code, fichiers du Designer, références et configuration.
- **Page / Window** — un écran ou une vue présentée à l'utilisateur (les Window prennent souvent la forme d'une boîte de dialogue).
- **Contrôle** — un objet d'interface comme un bouton, un libellé, une zone de texte, une grille ou une case à cocher.
- **Propriété** — un réglage d'un contrôle : `Text`, `Name`, `Dock`, `Size`, `Visible`.
- **Événement** — quelque chose qui se produit (un clic, un changement de sélection) et qui exécute du code.
- **Code-behind** — le fichier `.cs` habituel où se trouve le comportement.
- **Classe de service** — une classe distincte qui contient la logique métier et l'accès aux données, à l'écart de l'interface.