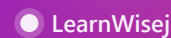


Premiers pas avec Wisej.NET

D'un Visual Studio vide à une application qui tourne — et les habitudes qui gardent le code propre.



Objectif de la leçon

Créer un premier projet Wisej.NET, l'ouvrir dans le Designer, l'exécuter en local, examiner la structure de la solution et savoir où se trouvent les fichiers de démarrage et de configuration.

1. Partir du modèle de projet

Dans Visual Studio, créez un nouveau projet à partir des modèles Wisej.NET installés. La liste exacte varie selon la version, mais la démarche reste la même : **créer un nouveau projet** → **rechercher « Wisej.NET »** → **choisir un modèle d'application web Wisej.NET** → **lui donner un nom** → **créer**.

Nom de projet d'entraînement suggéré

WisejTrainingApp



Avant de commencer

Vérifiez que l'extension Wisej.NET pour Visual Studio et ses modèles sont installés. Si les modèles n'apparaissent pas, contrôlez de nouveau l'installation ou redémarrez Visual Studio.

2. Ouvrir le Designer Wisej.NET

Le Designer est l'endroit où vous placez les contrôles visuellement. Ouvrez la page ou la fenêtre principale et passez en mode conception, puis faites glisser des contrôles depuis la boîte à outils, définissez leurs propriétés et double-cliquez sur un contrôle pour créer un gestionnaire d'événements.

Tâche dans le Designer	Ce que vous faites
Ajouter un contrôle	Faites glisser un <code>Button</code> , un <code>Label</code> , une <code>TextBox</code> , un <code>Panel</code> ou une <code>DataGridView</code> sur la surface.
Définir le texte	Sélectionnez le contrôle, ouvrez les Propriétés et modifiez la propriété <code>Text</code> .
Renommer un contrôle	Remplacez <code>Name</code> par un nom explicite comme <code>btnRun</code> , <code>lblStatus</code> , <code>txtCustomerName</code> .
Créer le code du clic	Double-cliquez sur un bouton (ou utilisez le panneau Événements) pour créer un gestionnaire <code>Click</code> .
Ajuster la mise en page	Utilisez <code>Dock</code> , <code>Anchor</code> , <code>Size</code> , <code>Location</code> , <code>Margin</code> , <code>Padding</code> .

3. L'exécuter en local

Exécuter en local signifie tester sur votre propre ordinateur avant de publier. Utilisez le bouton habituel Démarrer / Déboguer — le projet est compilé, un serveur local démarre et l'application s'ouvre dans le navigateur.

1. Cliquez sur **Démarrer / Déboguer** dans Visual Studio.
2. Attendez la fin de la compilation du projet.
3. Vérifiez que le navigateur ouvre l'application Wisej.NET.
4. Testez un clic sur un bouton pour confirmer que votre code s'exécute.
5. Utilisez des points d'arrêt pour inspecter les gestionnaires d'événements et les variables.



Réflexe de dépannage

Si l'application ne s'ouvre pas, lisez d'abord la **Liste d'erreurs** et la fenêtre **Sortie** de Visual Studio. La plupart des problèmes de débutant viennent de packages manquants, d'erreurs de compilation, d'un mauvais projet de démarrage ou d'un fichier renommé dont les références n'ont pas été mises à jour.

4. Examiner la structure de la solution

L'Explorateur de solutions n'est pas un tas de fichiers au hasard — il montre comment l'application est organisée.

Fichier ou dossier	À quoi il sert
<code>Program.cs</code>	Le code de démarrage qui lance l'application et affiche souvent la première Page ou Window.
<code>Page1.cs</code> / <code>Window1.cs</code>	Le code-behind : là où vous écrivez les gestionnaires d'événements et le comportement de l'écran.
<code>*.Designer.cs</code>	La mise en page et l'initialisation des contrôles générées par le Designer. À ne pas modifier à la main.
<code>*.resx</code>	Les fichiers de ressources du Designer et les données localisées.
Fichiers de configuration	<code>Web.config</code> et la configuration JSON propre à Wisej.NET.
<code>Models/</code>	Des classes de données simples comme <code>Ticket.cs</code> ou <code>Customer.cs</code> .
<code>Services/</code>	La logique métier, par exemple <code>TicketService.cs</code> .

5. Les bonnes habitudes du développeur

Habitude	À éviter	À privilégier
Nommer clairement les contrôles	<code>button1</code> , <code>label2</code>	<code>btnSaveTicket</code> , <code>lblStatus</code>
Garder les événements lisibles	200 lignes dans <code>btnSave_Click</code>	Appeler <code>ValidateInput()</code> et <code>ticketService.Save(ticket)</code>
Séparer la logique métier	Toute la logique d'enregistrement dans la page	Un <code>TicketService</code> porte les règles
Organiser les fichiers	Tout à la racine du projet	Des dossiers Models, Services, Pages, Controls

6. Exemple : un gestionnaire d'événements lisible

Un gestionnaire d'événements doit se lire comme une courte histoire — récupérer la saisie, appeler la logique, mettre à jour l'interface :

```
private void btnSayHello_Click(object sender, EventArgs e)
{
    string name = txtName.Text.Trim();

    if (string.IsNullOrEmpty(name))
    {
        lblStatus.Text = "Please enter a name.";
        return;
    }

    lblStatus.Text = $"Hello, {name}!";
}
```

À mesure que l'application grandit, déplacez les règles et les opérations sur les données dans des classes de service :

```
private readonly TicketService ticketService = new TicketService();

private void btnSaveTicket_Click(object sender, EventArgs e)
{
    Ticket ticket = ReadTicketFromScreen();
    ticketService.Save(ticket);
    lblStatus.Text = "Ticket saved.";
}
```



Vous êtes prêt pour le lab

Ensuite : regardez la vidéo pas à pas de construction, réussissez la vérification des connaissances, puis créez et exécutez votre premier écran Wisej.NET dans le lab pratique.