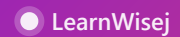


Propriétés, événements et code généré par le Designer

Ce que génère le Designer, où placer votre code et comment brancher proprement l'événement d'un bouton.



Objectif de la leçon

Comprendre comment fonctionnent les contrôles une fois placés : définir des propriétés, brancher des événements et garder votre code séparé du fichier généré par le Designer.

1. Propriétés, événements et méthodes

Une fois un contrôle posé sur la page, trois notions décrivent la façon de travailler avec lui :

Notion	Signification	Exemple
Propriété	Un réglage ou une valeur d'un contrôle	<code>btnStart.Text = "Start"</code>
Événement	Quelque chose qui se produit, souvent à cause de l'utilisateur	<code>btnStart.Click</code>
Gestionnaire d'événements	La méthode qui s'exécute quand l'événement se produit	<code>btnStart_Click(...)</code>
Méthode	Une commande que vous appelez pour déclencher une action	<code>lstEventLog.Items.Add(...)</code>

2. Où placer votre code

Une page comme `DashboardWindow` est répartie sur plusieurs fichiers. Savoir quel fichier contient quoi garde votre projet propre.

Fichier	Ce qu'il contient
<code>DashboardWindow.cs</code>	Vos gestionnaires d'événements, vos méthodes utilitaires et votre logique lisible.
<code>DashboardWindow.Designer.cs</code>	Le code de mise en page généré par le Designer. Ne le modifiez pas à la main.
<code>Program.cs</code>	Le code de démarrage qui ouvre la première page ou fenêtre.



IntelliSense & la documentation de l'API

Pendant la saisie, IntelliSense affiche les propriétés, méthodes et événements disponibles sur un contrôle — `Text`, `Enabled`, `Click`, `Show`, `Hide`. Quand vous avez besoin de plus de détails, tournez-vous vers la documentation de l'API.

3. Le cycle de vie pour débutants

Comprendre l'ordre dans lequel les choses se produisent évite des bugs déroutants — comme utiliser un contrôle avant qu'il n'existe.

L'application démarre depuis

- la première page ou fenêtre est créée
- `Form` construit les contrôles à partir du Designer
- l'utilisateur clique sur des boutons ou modifie des saisies
- les `EventHandler` s'exécutent en réponse

4. Brancher l'événement Click d'un bouton

Double-cliquez sur un bouton dans le Designer pour créer son gestionnaire `Click`. Dans le lab, les boutons du tableau de bord n'ont besoin ni d'une vraie base de données ni d'une API — l'objectif est un **comportement piloté par les événements** : un clic modifie des libellés, des indicateurs d'état et des messages de journal.

```
private void btnStart_Click(object sender, EventArgs e)
{
    lblStatus.Text = "Status: Running";
    lblServerStatus.Text = "Server: Online";
    lblDatabaseStatus.Text = "Database: Online";
    lblApiStatus.Text = "API Service: Online";
    AddLog("Dashboard started.");
}

private void AddLog(string message)
{
    lstEventLog.Items.Add(DateTime.Now.ToString("hh:mm tt") + " - " + message);
}
```

Remarquez que la journalisation répétée est extraite dans une petite méthode utilitaire `AddLog()`, pour que chaque gestionnaire reste court et lisible.

5. Bonnes habitudes de gestion des événements

- Utilisez des gestionnaires d'événements pour les actions de l'utilisateur comme les clics sur les boutons.
- Gardez chaque gestionnaire facile à lire.
- Déplacez la logique répétée dans des méthodes utilitaires comme `AddLog()`.

- Ne modifiez pas à la main le code de mise en page généré par le Designer, sauf si vous savez exactement pourquoi.
- Ne cachez pas d'appels lents à une base de données ou à une API dans de simples événements d'interface, sauf si la tâche le demande explicitement.



Vous êtes prêt pour le lab

Ensuite : regardez la vidéo pas à pas du tableau de bord, réussissez la vérification des connaissances, puis construisez et exécutez vous-même la page du tableau de bord.