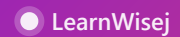


Propiedades, eventos y código generado por el Designer

Qué genera el Designer, dónde va tu código y cómo conectar limpiamente el evento de un botón.



Objetivo de la lección

Entiende cómo funcionan los controles una vez colocados: establece propiedades, conecta eventos y mantén tu código separado del archivo generado por el Designer.

1. Propiedades, eventos y métodos

Una vez que un control está en la página, tres ideas describen cómo trabajas con él:

Concepto	Significado	Ejemplo
Propiedad	Un ajuste o valor de un control	<code>btnStart.Text = "Start"</code>
Evento	Algo que ocurre, a menudo por acción del usuario	<code>btnStart.Click</code>
Manejador de eventos	El método que se ejecuta cuando ocurre el evento	<code>btnStart_Click(...)</code>
Método	Una orden a la que llamas para que ocurra algo	<code>lstEventLog.Items.Add(...)</code>

2. Dónde va tu código

Una página como `DashboardWindow` se reparte entre varios archivos. Saber qué contiene cada uno mantiene limpio tu proyecto.

Archivo	Qué va ahí
<code>DashboardWindow.cs</code>	Tus manejadores de eventos, métodos auxiliares y lógica legible.
<code>DashboardWindow.Designer.cs</code>	El código de diseño generado por el Designer. No lo edites a mano.
<code>Program.cs</code>	Código de inicio que abre la primera página o ventana.



IntelliSense y la documentación de la API

Mientras escribes, IntelliSense muestra las propiedades, métodos y eventos disponibles en un control:

`Text` , `Enabled` , `Click` , `Show` , `Hide` . Cuando necesites más detalle, recurre a la documentación de la API.

3. El ciclo de vida para principiantes

Entender el orden en que ocurren las cosas evita errores confusos, como intentar usar un control antes de que exista.

```
La aplicación arranca desde
→ se crea la primera página o ventana
→ construye los controles del Designer
→ el usuario hace clic en botones o cambia entradas
→ los se ejecutan en respuesta
```

4. Conecta el evento Click de un botón

Haz doble clic en un botón en el Designer para crear su manejador `Click`. En el laboratorio, los botones del dashboard no necesitan una base de datos ni una API reales: el objetivo es un **comportamiento basado en eventos**, en el que un clic cambia etiquetas, indicadores de estado y mensajes del registro.

```
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);
}
```

Fíjate en que el registro repetido se extrae a un pequeño helper `AddLog()` para que cada manejador siga siendo breve y legible.

5. Hábitos seguros al gestionar eventos

- Usa manejadores de eventos para las acciones del usuario, como los clics en botones.
- Mantén cada manejador fácil de leer.
- Traslada la lógica repetida a métodos auxiliares como `AddLog()`.
- No edites a mano el código de diseño generado por el Designer salvo que sepas exactamente por qué.
- No escondas llamadas lentas a bases de datos o API dentro de eventos simples de la interfaz, salvo que la tarea lo pida expresamente.



Ya estás listo para el laboratorio

A continuación: mira el recorrido guiado del dashboard, supera la evaluación de conocimientos y después construye y ejecuta tú mismo la página del dashboard.