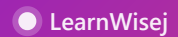


Eigenschaften, Events und vom Designer erzeugter Code

Was der Designer erzeugt, wohin Ihr Code gehört und wie Sie ein sauberes Button-Event verdrahten.



Lernziel

Verstehen Sie, wie Controls nach dem Platzieren funktionieren: Eigenschaften festlegen, Events verdrahten und Ihren Code von der vom Designer erzeugten Datei getrennt halten.

1. Eigenschaften, Events und Methoden

Sobald ein Control auf der Page liegt, beschreiben drei Begriffe, wie Sie damit arbeiten:

Begriff	Bedeutung	Beispiel
Eigenschaft	Eine Einstellung oder ein Wert an einem Control	<code>btnStart.Text = "Start"</code>
Event	Etwas, das passiert – oft durch den Benutzer ausgelöst	<code>btnStart.Click</code>
Event-Handler	Die Methode, die beim Eintreten des Events ausgeführt wird	<code>btnStart_Click(...)</code>
Methode	Ein Befehl, den Sie aufrufen, damit etwas passiert	<code>lstEventLog.Items.Add(...)</code>

2. Wohin Ihr Code gehört

Eine Page wie `DashboardWindow` ist auf mehrere Dateien verteilt. Wer weiß, welche Datei wofür zuständig ist, hält sein Projekt sauber.

Datei	Was hineingehört
<code>DashboardWindow.cs</code>	Ihre Event-Handler, Hilfsmethoden und gut lesbare Logik.
<code>DashboardWindow.Designer.cs</code>	Vom Designer erzeugter Layout-Code. Nicht von Hand bearbeiten.
<code>Program.cs</code>	Startcode, der die erste Page bzw. das erste Window öffnet.



IntelliSense & die API-Dokumentation

Beim Tippen zeigt IntelliSense die Eigenschaften, Methoden und Events, die ein Control bietet – `Text`, `Enabled`, `Click`, `Show`, `Hide`. Brauchen Sie mehr Details, greifen Sie zur API-Dokumentation.

3. Der Lebenszyklus für Einsteiger

Wer die Reihenfolge der Abläufe kennt, vermeidet verwirrende Fehler – etwa ein Control zu verwenden, bevor es existiert.

```
Die App startet in
→ die erste Page bzw. das erste Window wird erzeugt
→ baut die Controls aus dem Designer auf
→ der Benutzer klickt auf Buttons oder ändert Eingaben
→ als Reaktion laufen die
```

4. Ein Click-Event eines Buttons verdrahten

Doppelklicken Sie im Designer auf einen Button, um seinen `Click`-Handler anzulegen. Im Lab brauchen die Dashboard-Buttons keine echte Datenbank und keine API – es geht um **ereignisgesteuertes**

Verhalten: Ein Klick ändert Labels, Statusanzeigen und Log-Meldungen.

```
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);
}
```

Beachten Sie, dass das wiederholte Logging in eine kleine Hilfsmethode `AddLog()` ausgelagert ist – so bleibt jeder Handler kurz und lesbar.

5. Sichere Gewohnheiten beim Event-Handling

- Verwenden Sie Event-Handler für Benutzeraktionen wie Button-Klicks.
- Halten Sie jeden Handler leicht lesbar.
- Lagern Sie wiederholte Logik in Hilfsmethoden wie `AddLog()` aus.
- Bearbeiten Sie vom Designer erzeugten Layout-Code nicht von Hand, außer Sie wissen genau, warum.
- Verstecken Sie langsame Datenbank- oder API-Aufrufe nicht in einfachen UI-Events, außer die Aufgabe verlangt es ausdrücklich.



Sie sind bereit für das Lab

Als Nächstes: Sehen Sie sich den Video-Walkthrough zum Dashboard an, bestehen Sie den Wissenstest und erstellen und starten Sie dann die Dashboard-Page selbst.