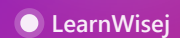


Properties, Events & Designer-Generated Code

What the Designer generates, where your code belongs, and how to wire a clean button event.



Lesson goal

Understand how controls work after they're placed: set properties, wire events, and keep your code separate from the Designer-generated file.

1. Properties, events and methods

Once a control is on the page, three ideas describe how you work with it:

Concept	Meaning	Example
Property	A setting or value on a control	<code>btnStart.Text = "Start"</code>
Event	Something that happens, often because of the user	<code>btnStart.Click</code>
Event handler	The method that runs when the event happens	<code>btnStart_Click(...)</code>
Method	A command you call to make something happen	<code>lstEventLog.Items.Add(...)</code>

2. Where your code goes

A page like `DashboardWindow` is split across files. Knowing which file owns what keeps your project clean.

File	What belongs there
<code>DashboardWindow.cs</code>	Your event handlers, helper methods, and readable logic.
<code>DashboardWindow.Designer.cs</code>	Generated layout code from the Designer. Don't edit this by hand.
<code>Program.cs</code>	Startup code that opens the first page or window.



IntelliSense & the API docs

While typing, IntelliSense shows the properties, methods and events available on a control — `Text` , `Enabled` , `Click` , `Show` , `Hide` . When you need more detail than that, reach for the API documentation.

3. The beginner lifecycle

Understanding the order things happen prevents confusing bugs — like trying to use a control before it exists.

App starts from

- the first page or window is created
- builds the controls from the Designer
- the user clicks buttons or changes inputs
- run in response

4. Wire a button Click event

Double-click a button in the Designer to create its `Click` handler. In the lab, the dashboard buttons don't need a real database or API — the goal is **event-driven behavior**: a click changes labels, status indicators and log messages.

```
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);
}
```

Notice the repeated logging is pulled out into a small `AddLog()` helper so every handler stays short and readable.

5. Safe event-handling habits

- Use event handlers for user actions like button clicks.
- Keep each handler easy to read.
- Move repeated logic into helper methods such as `AddLog()`.
- Don't manually edit Designer-generated layout code unless you know exactly why.
- Don't hide slow database or API calls inside simple UI events unless the task specifically calls for it.



You're ready for the lab

Next: watch the dashboard walkthrough, pass the knowledge check, then build and run the dashboard page yourself.