WISEJ.NET 2.0 CONTROLS AND A NORTHWIND APPLICATION · MODULE 2

A Side Button That Opens and Closes a Detail Panel

Lesson Guide · Lab / Exam Guide

LearnWisej

Lesson Guide

Learn the side-button interaction with a small supplied component: a Boolean state, one change event and a panel controlled by its host.

> **Control learning focus**
> Learn the side-button interaction with a small supplied component: a Boolean state, one change event and a panel controlled by its host.

1. What you will learn

A detail panel should open predictably without taking over the main screen. Build the side-button behavior in a small practice page before connecting it to any order data. The included CollapsibleSideButton is original course code, not a built-in framework class. Its entire implementation is supplied so the exercise has no external extension-source dependency.

2. Key concepts

Concept	Why it matters
Side-button pattern	A compact action opens or closes a companion panel.
CollapsibleSideButton	An original course component derived from Wisej.Web.Button.
Collapsed	True means the companion panel should be closed.
CollapsedChanged	The component's event reports a state transition.
Single state owner	The Boolean property is the authority; caption and panel follow it.
Idempotent assignment	Assigning the current state again must not raise another change.

3. Lesson narrative

Read the supplied component

Open the supplied CollapsibleSideButton.cs. The class derives from Wisej.Web.Button and adds a collapsed field, a Collapsed property and a CollapsedChanged event. This is a small teaching implementation of the

side-button interaction. Do not assume the class exists in the framework toolbox: add the supplied file to your practice project.

The constructor starts with the caption Show details and attaches one Click handler. That handler assigns the opposite of the current Collapsed value. The property setter ignores assignments that do not change the value, updates the caption and raises CollapsedChanged. Read these operations in order and explain which line prevents duplicate notifications.

The component does not know which panel it controls. This keeps it reusable for order details, customer information or a help panel. It also makes the event contract visible: the host listens for a state change and decides what that state means in the application.

Connect the component to a panel

Create one CollapsibleSideButton and one Panel during practice-page initialization. Set the panel width and dock it to the right. Set its initial Visible value from the inverse of side.Collapsed. Place a Label inside so you can see whether the panel really opens and closes.

Subscribe to CollapsedChanged once and assign detail.Visible = !side.Collapsed. The inverse is intentional: collapsed means the panel is closed. Avoid a second Boolean such as detailsOpen unless you have a clear synchronization rule. Two independent flags allow the caption, property and panel to disagree.

Click the button and inspect all three observations: Collapsed is false, the caption says Hide details, and the panel is visible. Click again and confirm the inverse state. The containing page owns the panel's location and content; the component only owns the toggle behavior.

Handle programmatic changes safely

Set side.Collapsed = true from a Reset button or when navigating away from the view. Use the property rather than changing only the panel's Visible value. That keeps the caption, state and event path synchronized even when the learner did not click the side button.

Set Collapsed to true twice and count the events. Only a real state change should raise CollapsedChanged. The early return in the setter prevents duplicate work. This matters when a navigation routine closes details repeatedly or several initialization steps apply the same state.

Keep the event handler small. Setting the panel's visibility should not modify the Collapsed property again. Circular updates are hard to debug and can produce recursion. When later loading order details, put that work in a separate method called after the panel is shown.

Test the behavior before adding data

Start with the default closed state, then run open, close, open and reset. Record the caption, Boolean and visibility after every action. Repeat the open-close sequence ten times. The page should still contain one button and one detail panel.

Inspect a narrow browser width and a long detail message. Reserve enough space for the panel or choose an appropriate alternative layout in the host. A toggle button does not automatically make a fixed-width panel responsive. Keyboard activation and a meaningful caption must also remain usable.

Use AI to explain the setter and suggest tests for repeated assignment. Require the assistant to preserve the distinction between this supplied component and framework APIs. Save the tested component and

host wiring; the final project will use the same event to expose the selected order.

Learn the control, then verify the implementation

Explain the control properties, data binding and event path yourself. Give AI the supplied example and one bounded task. Review the patch, compile against the published Wisej 2.0.54 packages, test independent expectations and observe browser behavior.

Learning objectives

By the end of this module you'll be able to:

- ✓ Explain the difference between a button's state and the panel it controls.
- ✓ Use the supplied CollapsibleSideButton component built on the Wisej 2.0 Button API.
- ✓ Handle CollapsedChanged and synchronize panel visibility.
- ✓ Test initial state, repeated clicks and programmatic state changes.

Included course examples

`code/NorthwindCourse/CollapsibleSideButton.cs`

These original files are included in the code download. Build the exercise before comparing with the completed example.

Lab / Exam Guide

Build the module slice, verify its behavior and submit the required evidence.

Hands-on lab

Add the supplied CollapsibleSideButton to a practice Page and use it to control one right-docked detail panel.

Starting point

Use a separate practice Page so you can understand this control in isolation. Use the published Wisej 2.0.54 packages and the supplied original examples. A licensed browser run is required for UI evidence.

Suggested approach

- ✓ Add the supplied CollapsibleSideButton.cs file and read its state setter.
- ✓ Create one button and one right-docked Panel with a visible label.
- ✓ Initialize panel visibility from Collapsed and subscribe once to CollapsedChanged.
- ✓ Test clicking, programmatic reset and assigning the same state twice.
- ✓ Record state, caption, visibility and event counts at wide and narrow widths.

Required deliverables

- ✓ An explanation of the original component's property and event
- ✓ A working button-to-panel connection
- ✓ A transition table for open close and reset
- ✓ An event count proving repeated assignment is ignored
- ✓ Keyboard and narrow-width observations

Student review questions

- ✓ Why does the host use the inverse of Collapsed?
- ✓ Which value is authoritative when the page resets?
- ✓ What prevents the same assignment from causing repeated work?

Acceptance and scoring

Submit five evidence items, each worth 20 points. A working claim with repeatable evidence earns 20, a partial implementation with a documented gap earns 10, and missing or contradictory evidence earns 0. Pass at 80/100. The in-player code checks are pattern checks, not a compiler or proof of runtime correctness; the instructor or reviewer must inspect the evidence.

Control configuration example

This original one-shot fixture demonstrates control configuration. Run it once in a scratch Page. In the final application, configure existing designer controls and refresh their data instead of creating duplicate

controls on each click. Use System, System.Linq, Wisej.Web, Wisej.Web.Ext.NavigationBar and the supplied NorthwindCourse namespace as needed. Reuse existing controls in the final application.

```
private void btnRun_Click(object sender, EventArgs e)
{
    try {
        var detail = new Panel { Dock = DockStyle.Right, Width = 230 };
        var side = new CollapsibleSideButton();
        detail.Visible = !side.Collapsed;
        side.CollapsedChanged += (s, args) ⇒ {
            detail.Visible = !side.Collapsed;
        };
        Controls.Add(detail);
        Controls.Add(side);
        side.Collapsed = false;
    } catch (Exception ex) { MessageBox.Show(ex.Message); }
}
```

Learn the control, then verify the implementation

Explain the control properties, data binding and event path yourself. Give AI the supplied example and one bounded task. Review the patch, compile against the published Wisej 2.0.54 packages, test independent expectations and observe browser behavior.