WISEJ.NET 2.0 CONTROLS AND A NORTHWIND APPLICATION · MODULE 3

FlexLayoutPanel for Adaptable Control Layouts

Lesson Guide · Lab / Exam Guide

LearnWisej

Lesson Guide

Arrange controls with FlexLayoutPanel, explain direction and fill weight, and make a horizontal-to-vertical layout work in both directions.

Control learning focus

Arrange controls with FlexLayoutPanel, explain direction and fill weight, and make a horizontal-to-vertical layout work in both directions.

1. What you will learn

Fixed coordinates break when a label grows or a browser becomes narrow. In this exercise you will build a filter row from ordinary controls, then turn it into a vertical stack. Learn the layout container first; business filtering will be added after the individual controls are understood.

2. Key concepts

Concept	Why it matters
LayoutStyle	Chooses a horizontal or vertical arrangement.
Spacing	Sets the distance between neighboring children.
Fill weight	Shares remaining space among participating child controls.
SetFillWeight	Sets the panel's extender property on a child control.
Preferred size	A child's content and size requirements influence layout.
Responsive transition	Both narrow and wide configurations must be explicitly restored.

3. Lesson narrative

Build a horizontal practice row

Create a FlexLayoutPanel on a practice Page and add a Label, TextBox and Button. Set LayoutStyle to FlexLayoutStyle.Horizontal and Spacing to a visible value such as 10. Give the row enough height to show every child. Use actual parent-child relationships: adding a control to the Page instead of the panel prevents the panel from arranging it.

Run the page and resize it gradually. Observe the label's required space, the editor's width and the action button. Layout behavior comes from the panel's direction and its children's size settings. Spacing adds gaps; it is not a substitute for giving a long editor enough width.

Change the label text to a longer sentence. Record which children change position or size. This makes layout constraints visible before business code obscures them. Avoid fixing the result with manual Location assignments because the layout engine will arrange the children again.

Assign remaining space deliberately

Call panel.SetFillWeight(editor, 1) to give the editor a share of the available space. SetFillWeight is provided by the containing FlexLayoutPanel; it is not a property you invent on TextBox. The first argument must refer to a child controlled by that panel.

Compare two weighted editors with weights one and two, keeping other settings the same. Their shares concern the space that remains after the layout accounts for other requirements. Do not promise an exact one-third and two-thirds of the entire page: minimum sizes, fixed controls and gaps also consume room.

Keep action buttons at a usable size and allow the editor to absorb extra width. Check very narrow sizes where there may be no spare room. Fill weight cannot create space that the parent does not have. At that point the application needs another direction, scrolling or another explicit layout choice.

Switch direction and restore the wide layout

Choose a practice breakpoint, for example 850 pixels of page width. Below it set LayoutStyle to Vertical and increase the panel's height. Above it restore Horizontal and the original height. Use one method to apply both states so the wide settings are not forgotten after a phone-sized test.

The same child controls must survive the transition. Creating a new TextBox in the narrow mode would discard text, focus and event subscriptions. Change layout properties on existing objects. Record the editor text before resizing and verify that it is still present afterward.

Consider where the width is measured. A page may have a left navigation strip and a right detail panel, so its total width is not identical to the width available to the filter container. The example uses a simple page breakpoint; a production design should choose its transition based on the space the content actually receives.

Review content growth and usability

Test short text, long text and empty text at wide and narrow widths. Confirm that controls remain readable, their labels are associated with the right inputs and the action button is reachable. FlexLayoutPanel arranges children, but it does not automatically decide every sensible size or overflow policy.

Use a keyboard to move through the row and through the stacked layout. Layout direction should not create a confusing interaction order. Record wide-to-narrow and narrow-to-wide screenshots, along with any clipping or horizontal scrolling you observe.

Ask AI to explain SetFillWeight, parent ownership and the two responsive branches before changing the code. Reject a patch that creates replacement controls on resize. Save the tested panel: the capstone will fill it with the tag editor, date inputs, switch and side button.

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:

- ✓ Place controls in a FlexLayoutPanel and set LayoutStyle.
- ✓ Explain Spacing and the allocation of remaining space with SetFillWeight.
- ✓ Switch between horizontal and vertical layouts without recreating controls.
- ✓ Test long content and narrow-to-wide layout transitions.

Included course examples

`code/NorthwindCourse/NorthwindPage.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

Build a FlexLayoutPanel practice row that switches direction while preserving the same child controls and their values.

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 a Label TextBox and Button to one FlexLayoutPanel.
- ✓ Set horizontal LayoutStyle and Spacing and inspect the initial layout.
- ✓ Use SetFillWeight on the editor and explain what remaining space means.
- ✓ Switch to a taller vertical panel below a chosen breakpoint and restore the wide settings above it.
- ✓ Test long text keyboard order and a complete narrow-to-wide round trip.

Required deliverables

- ✓ A diagram of the parent and its three children
- ✓ An explanation of direction spacing and fill weight
- ✓ A two-branch resize method using existing controls
- ✓ Wide and narrow screenshots with long content
- ✓ Evidence that editor text and event subscriptions survive resizing

Student review questions

- ✓ Which object owns SetFillWeight?
- ✓ Why is remaining space different from the page width?
- ✓ Which properties must be restored when the page widens?

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 row = new FlexLayoutPanel { Dock = DockStyle.Top, Height = 80 };
        var editor = new TextBox { Text = "Keep this value" };
        row.LayoutStyle = FlexLayoutStyle.Horizontal;
        row.Spacing = 10;
        row.Controls.Add(editor);
        row.Controls.Add(new Button { Text = "Apply" });
        row.SetFillWeight(editor, 1);
        Controls.Add(row);
        Resize += (s, args) ⇒ {
            row.LayoutStyle = Width < 850 ? FlexLayoutStyle.Vertical :
FlexLayoutStyle.Horizontal;
            row.Height = Width < 850 ? 160 : 80;
        };
    } 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.