WISEJ.NET 2.0 CONTROLS AND A NORTHWIND APPLICATION · MODULE 7

Build the Northwind Application With the Controls You Learned

Lesson Guide · Lab / Exam Guide

LearnWisej

Lesson Guide

Build a complete Northwind-style teaching dashboard from a blank Wisej 2.0 page, reusing the navigation, side button, flexible layout, tag editor, switch and grid patterns from the earlier modules.

Control learning focus

Build a complete Northwind-style teaching dashboard from a blank Wisej 2.0 page, reusing the navigation, side button, flexible layout, tag editor, switch and grid patterns from the earlier modules.

1. What you will learn

Now turn the isolated exercises into an application. The final project is an original Northwind-style teaching dashboard with Overview, Sales, Orders, Fulfillment, Products, Customers and Team views. It uses a deliberately small synthetic dataset so you can explain and verify every result. Build it from the steps below, then compare your work with the supplied complete example.

2. Key concepts

Concept	Why it matters
Application shell	Navigation selects a view inside a shared content area.
Control composition	Each previously learned control keeps a small clear responsibility.
Shared order filters	Countries dates and open-only state feed one filter method.
View projection	Sales and customer summaries are derived from matching orders.
Detail identity	Order details use the selected bound record's OrderId.
Acceptance demonstration	Independent expected values and observed interactions prove the build.

3. Lesson narrative

Build the shell from a blank application

Create a fresh Wisej 2 Web Application, run its empty page and add the two pinned packages used in module one. Add OrderData.cs and CollapsibleSideButton.cs from the course download. Create a NorthwindPage class derived from Page. In the template-generated application Main method, assign

Application.MainPage = new NorthwindCourse.NorthwindPage(). The downloadable project is a compiled example library, so retain the web host and configuration created by the application template.

Add the left-docked NavigationBar and a fill-docked body Panel. Create seven NavigationBarItem objects for Overview, Sales, Orders, Fulfillment, Products, Customers and Team. Map each item object to a destination name. In ItemClick, update activeView, close the detail panel through the side button property and call RefreshRows. This reuses module one's item-identity pattern.

Add a top FlexLayoutPanel, a right detail Panel, a bottom status Label and a fill-docked DataGridView. Add a second fill-docked Label for the Overview summary and show only the relevant fill control. Initialize all controls once. Inspect dock order and parent ownership before adding filters.

Add the controls and one refresh path

Put TagTextBox, two DateTimePicker controls, the CheckBox with Appearance.Switch and CollapsibleSideButton in the filter container. Give the tag editor a comma separator and three-tag limit. Label the switch Open orders only. Initialize the date range to 1 through 6 May 1998 so the teaching data is visible on the first run.

Wire TextChanged, CheckedChanged and ValueChanged to one RefreshRows method. That method calls OrderData.Filter using the current controls, then supplies the resulting records to the active view. It also updates the count, monetary total and status message. Keep the calculations in OrderData rather than scattering slightly different queries through separate handlers.

Wire CollapsedChanged to detail.Visible = !side.Collapsed. When opening details, read the current bound OrderRow and display its OrderId, customer, country and amount. Enable the side button for Orders and Fulfillment, whose grid rows represent actual orders. Close it when changing views so a customer summary cannot leave stale order details open.

Implement the seven views

Overview displays the matching-order count and net amount. Sales groups matching orders by day and displays the date, order count and total. Orders displays the matching OrderRow objects directly. Fulfillment adds the unshipped condition to the shared filter. Use explicit column mappings for each projection and keep the source list unchanged.

Customers groups matching orders by customer and country and shows an order count and total. Products displays the three supplied synthetic inventory records, and Team displays two supplied staff records. Hide the order filter row in Products and Team because those views do not use order filters. Their status messages state that scope clearly.

Build each view yourself before opening the completed NorthwindPage.cs for comparison. Verify the model-to-column mapping and selected-row type after each view change. A Sales summary row is not an OrderRow, so the detail loader must not assume that every bound grid record represents an order.

Prove the application and explain the reuse

Start with all filters clear and the full date range. Overview must show six orders and 700.00. Germany must show two and 270.00; enabling Open orders only must reduce that to one and 150.00. Clearing the country

tags while keeping the switch on must show three and 280.00. Test the same conditions in Orders and explain any additional restriction in Fulfillment.

Exercise all seven navigation entries, open and close order details, clear the last tag, enter an unknown country and invert the date range. Resize from wide to narrow and back while preserving filter values. In narrow mode use CompactView and a vertical FlexLayoutPanel; restore both direction and dimensions when the window widens.

Submit a short guided demonstration that names the control responsible for every interaction. Include the package build log, pure data-test output, numerical expectations and browser observations. Explain which parts are synthetic teaching data and which steps would be needed to connect a real database. AI may help implement a bounded change, but you must explain the controls and verify the final result.

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:

- ✓ Assemble the previously practiced controls into a coherent application shell.
- ✓ Build seven business views from the supplied original teaching data.
- ✓ Share one filter pipeline across order-based views and preserve selected-record identity.
- ✓ Demonstrate the complete application with numerical interaction and responsive evidence.

Included course examples

`code/NorthwindCourse/NorthwindPage.cs` · `code/NorthwindCourse/CollapsibleSideButton.cs` · `code/NorthwindCourse/OrderData.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 and demonstrate a seven-view Northwind-style dashboard using the controls practiced in modules one through six.

Starting point

Start a fresh application and reuse the control patterns from modules one through six. Use the published Wisej 2.0.54 packages and the supplied original examples. A licensed browser run is required for UI evidence.

Suggested approach

- ✓ Create a blank Wisej 2.0 web application and add the supplied original data and side-button component.
- ✓ Build the seven-item NavigationBar shell and the body detail status and flexible filter containers.
- ✓ Wire the tag editor switch and date inputs to one RefreshRows method and bind each view explicitly.
- ✓ Add selected-order details and implement both directions of the compact and vertical layout transition.
- ✓ Run the complete numerical interaction and responsive acceptance demonstration before comparing with the supplied solution.

Required deliverables

- ✓ A runnable template-hosted application and package build log
- ✓ A seven-view navigation demonstration explaining the reused controls
- ✓ Verified counts and totals for the four specified filter states
- ✓ Order detail empty-state invalid-date and responsive evidence
- ✓ An AI decision log and a final explanation of data scope and remaining production work

Student review questions

- ✓ Which earlier module explains each control in the final screen?
- ✓ Why do Products and Team hide the order filters?
- ✓ What evidence shows the views agree about the same filtered orders?

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 all = OrderData.Create();
        var open = OrderData.Filter(all, "", true,
            new DateTime(1998, 5, 1), new DateTime(1998, 5, 6));
        if (all.Count ≠ 6 || all.Sum(r ⇒ r.NetAmount) ≠ 700m)
            throw new InvalidOperationException("Baseline differs");
        if (open.Count ≠ 3 || open.Sum(r ⇒ r.NetAmount) ≠ 280m)
            throw new InvalidOperationException("Open-order result differs");
        Application.MainPage = new NorthwindPage();
    } 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.