WISEJ.NET 2.0 CONTROLS AND A NORTHWIND APPLICATION · MODULE 1

NavigationBar From the First Item to a Working Menu

Lesson Guide · Lab / Exam Guide

LearnWisej

Lesson Guide

Learn NavigationBar on its own: create items, handle ItemClick, select a content panel and test compact navigation.

> **Control learning focus**
>
> Learn NavigationBar on its own: create items, handle ItemClick, select a content panel and test compact navigation.

1. What you will learn

A dashboard needs a predictable way to move between screens. Begin with a two-item practice application. Learn what the navigation control supplies and what your handler must do before adding business data. Save this example: the final project will reuse the same behavior for the Northwind-style menu.

2. Key concepts

Concept	Why it matters
NavigationBar	An extension control that presents an application menu and optional user panel.
NavigationBarItem	A menu entry with visible text and an identity used by the application.
ItemClick	The event supplies the selected item in e.Item.
Dock	DockStyle.Left reserves a strip while a content panel fills the rest.
CompactView	A reduced navigation presentation suited to a narrow layout.
View state	The menu selects a view; the view owns its data and controls.

3. Lesson narrative

Create the practice application

Use Windows, a compatible Visual Studio installation, the .NET Framework development tools and a licensed Wisej 2.0 installation. Create a Wisej 2 Web Application from its template and run the empty page first. The downloadable example library targets .NET Framework 4.8 and pins Wisej-2 and Wisej-2-

NavigationBar to 2.0.54. The packages supply runtime assemblies; the designer, web application template and license are separate setup requirements.

Add the NavigationBar extension reference and the namespace Wisej.Web.Ext.NavigationBar. Place a NavigationBar on a Page, set Dock to Left and give it a usable width. Add a Panel with Dock set to Fill for content. NavigationBar is an extension; Panel and Label belong to the framework. Keep that distinction clear when resolving missing types.

Create the controls once during initialization. Run again and confirm that the menu occupies the left edge and content uses the remaining area. If content obscures the menu, inspect parent containers and dock order. Do not compensate with arbitrary screen coordinates. The downloadable library provides examples to add to a template-generated web application; it is not a standalone web host.

Create and identify the items

Create two NavigationBarItem objects called overviewItem and ordersItem. Set Text to Overview and Orders, then add them to NavigationBar.Items. Keep their references as fields when multiple methods need them. Text is a label for people; the selected item object is the reliable identity used by your handler.

Subscribe to ItemClick once. Compare e.Item with the two item objects and update a content Label. This first exercise intentionally has no database. A successful result is a menu whose selections produce two different, predictable messages. Point to the exact event argument and assignment that cause the change.

Select each entry five times. There should still be two items, one content label and one response per click. Creating a new set of controls in every handler duplicates views and consumes memory. Later you may cache separate view objects, but a navigation event should select existing application state.

Configure compact presentation

NavigationBar.Text names the application. ShowUser controls the optional user area; UserName and UserStatus describe it when enabled. Leave ShowUser false in this exercise until an actual signed-in identity exists. A visible user area is presentation and does not authenticate anyone.

Set CompactView to true and then false. Verify that both entries remain selectable and expansion restores the expected layout. The final project will set this property when its page becomes narrow. Your application chooses the breakpoint; the control renders the requested presentation.

Keep meaningful text for each item and inspect keyboard focus in the chosen theme. A compact menu must remain understandable. A selected menu entry also does not grant permission to execute a business operation; that check belongs to the application logic.

Explain and verify the event path

Trace one interaction aloud: the learner selects Orders, NavigationBar raises ItemClick, the handler receives e.Item, and the application displays the Orders content. This explanation is the learning target. A screenshot alone does not show that you know how to use the event.

Capture the initial page, both selected messages and the compact-to-expanded round trip. Count the controls before and after repeated selection. Rename an item and confirm that object-based routing still reaches the correct view. Keep the exercise small enough for another learner to reproduce.

Ask an AI assistant to review the event subscription and docking order using the supplied example and the installed API information. Require it to explain every proposed property. Build the patch and rerun your checks before saving the checkpoint.

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:

- ✓ Create a Wisej 2.0 practice project with matching framework and extension packages.
- ✓ Configure a NavigationBar and NavigationBarItem objects.
- ✓ Use ItemClick and item identity to select a view.
- ✓ Verify compact navigation and repeated selection without duplicate controls.

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 standalone NavigationBar practice page that switches between Overview and Orders and survives a compact-mode round trip.

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

- ✓ Create and run the empty Wisej 2.0 practice project with matching package versions.
- ✓ Add a left-docked NavigationBar and a fill-docked content Panel.
- ✓ Create two items once and route ItemClick using e.Item identity.
- ✓ Toggle CompactView in both directions and repeat each selection five times.
- ✓ Capture both views, docking behavior and handler-count evidence.

Required deliverables

- ✓ A build log showing the pinned package versions
- ✓ A two-item menu and an explained ItemClick handler
- ✓ Evidence that changing item labels preserves routing
- ✓ Compact and expanded screenshots
- ✓ Repeated-selection results with no extra controls or handlers

Student review questions

- ✓ Which code owns each item identity?
- ✓ Why should initialization not run on every click?
- ✓ How will this menu select a business view in the final project?

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 nav = new NavigationBar();
        var item = new NavigationBarItem { Text = "Orders" };
        var output = new Label { Dock = DockStyle.Fill };
        nav.Dock = DockStyle.Left;
        nav.Items.Add(item);
        nav.ItemClick += (s, args) => {
            if (args.Item == item) output.Text = "Orders selected";
        };
        nav.CompactView = false;
        Controls.Add(output);
        Controls.Add(nav);
    } 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.