WISEJ.NET 2.0 CONTROLS AND A NORTHWIND APPLICATION · MODULE 5

CheckBox Switch Appearance and Boolean State

Lesson Guide · Lab / Exam Guide

LearnWisej

Lesson Guide

Use Appearance.Switch on a CheckBox, explain Checked and CheckedChanged, and connect a clearly labeled Boolean filter to application state.

> **Control learning focus**
>
> Use Appearance.Switch on a CheckBox, explain Checked and CheckedChanged, and connect a clearly labeled Boolean filter to application state.

1. What you will learn

A switch looks simple but can hide an ambiguous business rule. In this module a CheckBox labeled Open orders only has one precise meaning: checked includes only unshipped orders; unchecked includes both shipped and unshipped orders. Build and test that meaning before adding the switch to a dashboard.

2. Key concepts

Concept	Why it matters
CheckBox	The underlying Boolean input control.
Appearance.Switch	A presentation option on CheckBox rather than a separate Switch class.
Checked	The current Boolean selection.
CheckedChanged	The event used to react to a change in Checked.
Predicate	The business rule that decides whether a record is included.
Label contract	Text must state what turning the switch on means.

3. Lesson narrative

Configure the CheckBox appearance

Create a Wisej.Web.CheckBox and set Appearance to Appearance.Switch. Keep using CheckBox APIs: the appearance changes how the control looks, not its underlying role. There is no separate built-in Switch class to instantiate for this lesson.

Set Text to Open orders only and initialize Checked to false. Add a Label that reports the current Boolean value. Run the page and observe the control with both values. The label should remain understandable

without relying on the switch's color.

Check the chosen theme and keyboard interaction. A visual treatment can alter spacing and emphasis, but the user still needs a clear focus indicator and a predictable way to toggle the Boolean. Use a label that describes the enabled state rather than a vague word such as Status.

Connect the change event to one rule

Subscribe once to CheckedChanged. Read the control's current Checked value inside the handler and call a separate filter method. Do not assume that the last mouse click is the only way the value can change: initialization, binding or another application action may also set it.

The filter predicate is !openOnly || !order.Shipped. When openOnly is false, the first part permits every order. When it is true, the row must be unshipped. Write this explanation before using the expression so the negations do not become guesswork.

Use the supplied six-order data to verify the result. With the switch off, six orders total 700.00. With it on, three open orders total 280.00. These are teaching-data expectations, not totals from a production business database.

Keep display state and data consistent

Recompute the visible row list, count and net amount from the same filtered collection. If the grid uses the switch but the total uses all rows, the screen contradicts itself. One refresh method should produce all values that describe the visible orders.

Set Checked programmatically as part of Reset and verify that the same event path updates the result. Keep the control object and event subscription rather than creating a new switch each time. A second application Boolean can drift from Checked unless the synchronization rule is explicit.

The switch changes which records are displayed; it does not ship an order or change the stored Shipped property. Keep filtering and mutation separate. If a future application adds an action that changes shipment status, that action needs its own validation and confirmation behavior.

Combine filters and test both directions

Combine the switch with the country filter using AND. Open orders in Germany produce one row worth 150.00 in the supplied data. Clearing the country tags while leaving the switch on restores all three open orders, not all six orders.

Test off, on, off and Reset, including a filter combination that matches no rows. The empty result should display zero matching rows and a clear message. It should not keep a stale count or present the previous selection as though it still matches.

Ask AI to explain a truth table for the predicate and then compare it with your observations. Include both shipped and unshipped records in the test data. A test containing only open orders cannot prove that the switch excludes shipped ones.

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:

- ✓ Configure a CheckBox with Appearance.Switch.
- ✓ Explain Checked independently of its visual appearance.
- ✓ Use CheckedChanged to apply the open-orders filter.
- ✓ Verify initial state programmatic changes and accessible labeling.

Included course examples

`code/NorthwindCourse/NorthwindPage.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

Use CheckBox with Appearance.Switch to apply and clear an open-orders filter with matching row counts and totals.

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 a CheckBox with Switch appearance and an explicit enabled-state label.
- ✓ Initialize Checked to false and show its value in a diagnostic label.
- ✓ Handle CheckedChanged through one shared refresh method.
- ✓ Verify all-orders and open-orders counts and totals using the supplied data.
- ✓ Combine the switch with Germany tags and test Reset and an empty result.

Required deliverables

- ✓ A configured CheckBox and clear label
- ✓ A truth table for the open-order predicate
- ✓ Observed six-order and three-order results
- ✓ The one-order Germany result and its 150.00 total
- ✓ Keyboard reset and empty-result evidence

Student review questions

- ✓ What changes when Appearance becomes Switch?
- ✓ Why does filtering not change an order's shipment status?
- ✓ Which collection supplies both the grid and its total?

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 toggle = new CheckBox { Text = "Open orders only" };
        var output = new Label { Dock = DockStyle.Bottom };
        toggle.Appearance = Appearance.Switch;
        toggle.Checked = false;
        toggle.CheckedChanged += (s, args) ⇒ {
            var rows = OrderData.Filter(OrderData.Create(), "", toggle.Checked,
                new DateTime(1998, 5, 1), new DateTime(1998, 5, 6));
            output.Text = rows.Count + " matching orders";
        };
        Controls.Add(toggle);
        Controls.Add(output);
    } 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.