WISEJ.NET 2.0 CONTROLS AND A NORTHWIND APPLICATION · MODULE 4

TagTextBox for Multiple Filter Values

Lesson Guide · Lab / Exam Guide

LearnWisej

Lesson Guide

Use the Wisej 2.0 TagTextBox to collect multiple values, explain its text and tag events, and normalize the values before filtering data.

> **Control learning focus**
>
> Use the Wisej 2.0 TagTextBox to collect multiple values, explain its text and tag events, and normalize the values before filtering data.

1. What you will learn

A multi-value filter should be understandable and repeatable. First build a tag editor that prints its normalized values. Then connect the same parsing function to a tiny order list. The goal is to understand how tags become filter values before combining the editor with the other controls.

2. Key concepts

Concept	Why it matters
TagTextBox	The Wisej 2.0 class for an editor that presents values as tags.
SeparatorChar	Separates values in the editor's Text representation.
TextChanged	Notifies the application that the editor's text changed.
Tag lifecycle	TagAdded TagRemoved and TagRejected describe tag interactions.
Normalization	Trim values remove empty entries and compare with an explicit case policy.
Empty filter	The application decides whether no tags means all rows or no rows.

3. Lesson narrative

Create the editor and inspect its text

The class name in the published Wisej 2.0 assemblies is TagTextBox. Add a Wisej.Web.TagTextBox to the practice page, set SeparatorChar to a comma and set MaxTagCount to three. Give it enough width to enter and display several country names. The editor is sometimes described informally as tag input; use its actual C# type name in code.

Add Germany and UK as tags and inspect Text. The separator determines how the values are represented as text. Subscribe to TextChanged and display the current text in a Label. Do not immediately write a database query: first observe the values the control supplies.

Inspect TagCount as you add and remove values. MaxTagCount limits the number of tags, with zero meaning unlimited. The limit is an editor constraint, not a validation rule establishing that a country name exists. Those are separate decisions.

Understand the tag events

Use TagAdded and TagRemoved to observe individual changes and TagRejected to explain a rejected addition, such as exceeding the configured limit. TagTextBoxEventArgs.Text identifies the tag associated with these events. Keep the feedback specific enough for the learner to understand what happened.

Use one main path to recompute the filter, normally TextChanged in this course. If every tag event and TextChanged independently refresh the whole application, one user action can produce repeated work. Log the events during practice and decide which one owns the refresh.

TagRender can customize tag presentation, but color does not validate a value. Start with the default rendering. If you later add rendering rules, preserve readable text and ensure invalid or unknown values have an explanation rather than relying only on color.

Turn text into filter values

The supplied OrderData.ParseCountries function splits on commas, trims each value, removes empty entries and removes duplicates using an explicit case-insensitive comparison. Give the same function inputs such as Germany, germany and a value surrounded by spaces. The result should represent one country rather than three different filters.

In this course no tags means all countries. One or more tags means that an order's country must match any selected country. Countries within the tag list are combined with OR; the country filter is later combined with the date and open-order filters using AND. Write this rule before writing the query.

An unknown country produces no matching rows in the supplied small dataset. The page explains the empty result and lets the learner clear the filter. Do not silently replace an unknown value with a known country or treat no matches as an application crash. Real applications may instead restrict input to a validated vocabulary.

Test additions removals and empty results

Test one country, two countries, duplicates with different case and whitespace, an unknown country and an empty editor. For the supplied six-order dataset, Germany selects two orders and Germany plus UK selects four. Clearing the last tag restores all six. These independently stated expectations make the filter test meaningful.

Set MaxTagCount to three and attempt a fourth tag. Capture the rejection feedback without losing the existing tags. Then remove a tag and add another. A limit should constrain input consistently while allowing the user to correct it.

Ask AI to review normalization and the empty-filter rule. Require examples with expected results before accepting code. Keep business filtering in a testable method outside the control event. The capstone's

TextChanged handler will call that method and refresh the relevant views.

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 TagTextBox and configure SeparatorChar and MaxTagCount.
- ✓ Explain TextChanged and the tag lifecycle events.
- ✓ Normalize whitespace and duplicates before applying a country filter.
- ✓ Test empty input unknown values and removal of the last tag.

Included course examples

`code/NorthwindCourse/OrderData.cs` · `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 TagTextBox practice page that prints normalized country values and filters the supplied six-order dataset predictably.

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 TagTextBox with comma separators and a three-tag limit.
- ✓ Log TextChanged and tag lifecycle events while adding and removing values.
- ✓ Use OrderData.ParseCountries for trimming duplicate removal and case policy.
- ✓ Apply the documented empty-filter and country-OR rules to the small dataset.
- ✓ Verify one country multiple countries unknown input and clearing the last tag.

Required deliverables

- ✓ A configured TagTextBox and explained event path
- ✓ An event log showing addition removal and rejection
- ✓ Normalization examples with expected outputs
- ✓ Filter counts for Germany Germany plus UK and no tags
- ✓ Clear empty-result feedback and a recovery action

Student review questions

- ✓ How does a visual tag become a filter value?
- ✓ Which event owns the refresh?
- ✓ What should clearing the final tag do?

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 tags = new TagTextBox { SeparatorChar = ',', MaxTagCount = 3 };
        var output = new Label { Dock = DockStyle.Bottom };
        tags.TextChanged += (s, args) ⇒ {
            var values = OrderData.ParseCountries(tags.Text);
            output.Text = values.Length == 0 ? "All countries" : string.Join(", ", values);
        };
        tags.TagRejected += (s, args) ⇒ MessageBox.Show("Tag rejected: " + args.Text);
        Controls.Add(tags);
        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.