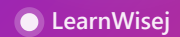


Application Resources, ResourceManager and Wisej System Text

Lesson Guide · Lab / Exam Guide



Lesson Guide

The shared resource layer: neutral and culture-specific .resx files that carry identical keys, Wisej.Resources.ResourceManager lookup by base name and assembly, placeholders formatted with string.Format and the active culture, services that return keys rather than English sentences, and Resources-[LANG].resx for Wisej system labels. What this means for a developer who knows basic C# and Wisej.NET controls and now has to ship the application in more than one language — and how to read this module.



Production focus

The shared resource layer: neutral and culture-specific .resx files that carry identical keys, Wisej.Resources.ResourceManager lookup by base name and assembly, placeholders formatted with string.Format and the active culture, services that return keys rather than English sentences, and Resources-[LANG].resx for Wisej system labels.

1. The production problem

Module 2 left the GlobalDesk customer editor looking convincing in German: every caption the designer owns changes with the culture and the layout survives. Then the user presses Save and reads *Customer saved.* in English, because that sentence is not a property of any control — it is a string literal in a click handler, beside *This field is required.* and the status line that counts loaded contacts.

Designer localization can never reach those strings: they are generated by code, shared by several screens, or produced by a service that has no idea a user interface exists. This module gives GlobalDesk a second resource layer for exactly that text, then goes one level lower: the labels Wisej.NET itself draws on a `MessageBox` or a calendar.

2. Key concepts

Concept	Why it matters
shared application resource file	<code>Resources/Strings.resx</code> holds the user-visible text that no single designed screen owns: save confirmations, validation messages, status lines, report headings and service results.
culture-specific sibling	<code>Strings.de.resx</code> and <code>Strings.it.resx</code> sit next to the neutral file and carry the same keys ; a translator changes values, never key names.
stable semantic key	A name such as <code>CustomerEditor.Saved</code> describes the meaning of the text, so rewording the English sentence never forces a code change or breaks the translations.
<code>Wisej.Resources.ResourceManager</code>	The lookup object that follows the .NET resource model: given a base name and an assembly it finds the value for the active culture and falls back toward the neutral file.
missing-key marker	A helper that returns <code>[Key]</code> when a lookup fails turns an invisible defect into something you can see and search for, instead of a blank label.
format string with placeholders	The whole sentence with its <code>{0}</code> and <code>{1}</code> lives in the resource and <code>string.Format</code> fills it, because languages order the name, the date and the verb differently.
<code>Resources-[LANG].resx</code>	A separate Wisej.NET system resource file in the project root or a <code>/Resources</code> folder that overrides or adds framework labels such as <code>\$Ok</code> , <code>\$Cancel</code> and <code>\$Offline</code> .

3. Lesson narrative

Two files, one set of keys

Start with the neutral file. In GlobalDesk it is `Resources/Strings.resx`, and it already carries the keys Module 1 introduced: `Navigation.Customers`, `CustomerEditor.Save`, `CustomerEditor.Cancel`, `CustomerEditor.Saved`, `Validation.Required`. The neutral file is the one .NET falls back to when nothing better exists, so it is never allowed to be incomplete. A key missing here is a developer defect, not a translation gap.

A translated file is the same file name with a culture name before the extension: `Strings.de.resx` for German, `Strings.it.resx` for Italian. Only the **value** column differs. Rename `CustomerEditor.Saved` in the German file and it stops answering the lookup, so the screen quietly falls back to English. The key list is a contract: adding a key means adding it everywhere.

One German file serves `de`, `de-DE` and `de-AT` alike: .NET walks from the most specific culture toward the neutral resource and takes the first match. Add `Strings.de-AT.resx` only when Austrian wording genuinely differs.

Looking text up through ResourceManager

Wisej.NET provides `Wisej.Resources.ResourceManager`, which follows the .NET resource model and knows how to locate culture-specific resources. You construct one from a base name and the assembly that contains the resources; the base name is the project namespace plus the folder and file name, which for GlobalDesk is `GlobalDesk.Resources.Strings`. Getting it wrong is the most common beginner failure, and it shows up as every key coming back missing at once.

Wrap the manager in one small static helper, so the rest of the application never touches it directly and one method decides what a missing key does:

```
using Wisej.Resources;

public static class Texts
{
    private static readonly ResourceManager RS =
        new ResourceManager("GlobalDesk.Resources.Strings", typeof(Texts).Assembly);

    public static string Get(string key)
        ⇒ RS.GetString(key) ?? $"[{key}]";
}
```

The bracketed marker is a good development default: an empty caption looks like a layout bug, while `[CustomerEditor.Saved]` tells you exactly which key to add and where. In production you may prefer to log the missing key instead, which Module 7 turns into a policy. What you must never do is let a missing resource become silent blank text.

Designer resources, shared resources, and where each string belongs

Two resource mechanisms now live in the same project, and they answer different questions. Designer resources belong to one designed surface, are written into that container's own `.resx` files, and are applied when its controls are created. Shared application resources belong to the application and are read on demand through `Texts.Get`, so they are as current as the moment you call them.

The practical test is ownership. If a caption exists because you dropped a `Button` on the customer editor, the designer owns it. If the text appears in a `MessageBox`, in a status line, in a grid column header you build in code, or on two different screens, it belongs in `Strings.resx`. Duplicating one sentence into three designer files is how half-translated applications happen: the third copy is always the one nobody updates.

The same reasoning reaches into the domain. A save service should not return the sentence *Customer saved*; it should return a stable result such as the key `CustomerEditor.Saved` or a result code the UI maps to a key. Business logic then stays testable, no English leaks out of the service layer, and the screen remains the only place that knows how to talk to a user.

Wisej's own labels: system localization and Resources-[LANG].resx

Not every word on screen is yours. Wisej.NET localizes its own framework resources — `MessageBox` buttons, calendar labels, and the culture-sensitive behavior of numeric, date and currency editors — and

ships built-in system translations for English, German, French, Italian, Turkish, Spanish and Portuguese. Run GlobalDesk under German and the Yes/No buttons change by themselves.

For a language Wisej.NET does not cover, or wording your product insists on, Wisej.NET supports a `Resources-[LANG].resx` file in the project root or a `/Resources` folder. Its keys are the framework's own and begin with a dollar sign: `$Ok`, `$Cancel`, `$Yes`, `$No`, `$Retry`, the calendar navigation labels and `$Offline`. Keep the two apart: `Strings.de.resx` is what GlobalDesk says, `Resources-de.resx` is what Wisej.NET says.

Messages, validation and images under the active culture

Sentences that mention data need placeholders, and placeholders are where beginners lose word order. Store the complete sentence in the resource — `Customer.LastOrder` holds `{0} last ordered on {1}`. — and let code supply the values: `string.Format(Application.CurrentCulture, Texts.Get("Customer.LastOrder"), customerName, lastOrderDate)`. Passing the culture also formats the date the way the session expects. Never concatenate `"Customer " + name + " saved"`: a translator cannot move a fragment that code glued together.

Placeholders are meaningless to someone who only sees the value column, so use the comment field in the `.resx` editor to say what `{0}` and `{1}` are. Validation messages deserve the same care: `Validation.Required` and `Validation.Email` are reused by several screens, which is why they belong in the shared file rather than in each form.

A `.resx` file can hold more than strings — images and other data are legitimate entries, and designer localization can swap an image per language. Do it on purpose, though. A picture that differs only because it has words baked into it is a maintenance trap; leave that text as real text in the UI, where `Texts.Get` translates it and no designer has to redraw anything.



Localization layers habit

Decide where each string lives before you type it. Designer resources belong to one designed screen and are applied when its controls are created; shared application resources in `.resx` carry the text that code generates or several screens reuse; `Resources-[LANG].resx` overrides Wisej.NET system labels. Never store a formatted date, number or currency: keep the value and format it for the active culture at display time. And remember that culture is per session, so one user's language must never change another user's screen.



Learning objectives

By the end of this module you'll be able to:

- ✓ Create neutral and culture-specific `.resx` resource files.
- ✓ Retrieve values through `ResourceManager` using stable keys.
- ✓ Organize reusable strings separately from designer resources.
- ✓ Explain Wisej.NET built-in system localization and custom `Resources-[LANG].resx` files.
- ✓ Use culture-aware resources for messages, validation and images.

Lab / Exam Guide

What you'll build in the hands-on lab, the suggested approach, and the required deliverables.



Hands-on lab

Grow the GlobalDesk shared resources. Add `Strings.de.resx` beside `Resources/Strings.resx` carrying exactly the same keys with translated values, add a `Customer.LastOrder` entry that holds two placeholders, and build the sentence with `string.Format` and `Application.CurrentCulture` instead of concatenating fragments. Route every message the customer editor shows through the `Texts` helper, including the save confirmation and the two validation messages, and have the save service return a stable result key such as `CustomerEditor.Saved` rather than an English sentence the UI prints directly. Then add a `Resources-de.resx` file that overrides one Wisej system label, prove it in a `MessageBox`, and write a short note on the difference between your product text and Wisej system text.

Starting point

You open GlobalDesk as Module 2 left it: the dashboard reads its captions from `Resources/Strings.resx` through the `Texts` helper, and the `CustomerEditor` user control is designer-localized with a German variant. The save confirmation, the two validation messages and the last-order line are still English literals in code, and there is no translated shared resource file yet.

Suggested approach

- ✓ Add `Resources/Strings.de.resx` next to the neutral file and copy the key list across unchanged, translating only the values; run the application under `de-DE` and confirm the dashboard captions switch before you touch any code.
- ✓ Add a `Customer.LastOrder` entry whose value is a complete sentence with `{0}` and `{1}`, fill the comment field so a translator knows the first is the customer name and the second the order date, and build the line with `string.Format(Application.CurrentCulture, Texts.Get("Customer.LastOrder"), customerName, lastOrderDate)`.
- ✓ Replace the literals in `CustomerEditor` with `Texts.Get` calls for `CustomerEditor.Saved`, `Validation.Required` and `Validation.Email`, and delete any string concatenation you find on the way.
- ✓ Change the save service so it returns a stable result — the key `CustomerEditor.Saved` or a result code — and let the editor map that result to text through `Texts.Get`, so no English sentence is left in the service layer.
- ✓ Add `Resources-de.resx` in the project root or the `/Resources` folder, override one Wisej.NET system label such as `$Ok` or `$Cancel`, then show a `MessageBox` under German and compare its buttons with the English run.

Required deliverables

- ✓ `Strings.de.resx` carrying the same keys as the neutral file with translated values only
- ✓ `Customer.LastOrder` resource built with `string.Format` and the active culture, never concatenated

- ✓ Save service returning a resource key or result code instead of an English sentence
- ✓ Resources-de.resx overriding at least one Wisej system label, shown in a MessageBox
- ✓ Lab note separating product text, Wisej system text and diagnostic text

Student review questions

- ✓ Delete one key from `Strings.de.resx` and run under German. Describe what the user sees, why nothing crashes, and how you would notice this in a real application with two hundred keys.
- ✓ Your save service currently returns `CustomerEditor.Saved`. What would break in the unit tests, and in the German screen, if it returned the sentence itself instead?
- ✓ A colleague adds the word "Cancel" to `Resources-de.resx` under a key of their own invention. Explain why the `MessageBox` button does not change, and where that string actually belonged.



Localization layers habit

Decide where each string lives before you type it. Designer resources belong to one designed screen and are applied when its controls are created; shared application resources in .resx carry the text that code generates or several screens reuse; Resources-[LANG].resx overrides Wisej.NET system labels. Never store a formatted date, number or currency: keep the value and format it for the active culture at display time. And remember that culture is per session, so one user's language must never change another user's screen.