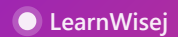


AI Coding Exercise

Build this module's GlobalDesk sample with ChatGPT or Claude from a ready-made prompt that points the model at this module's specification, then review, run and extend what it gives you.



You are not going to type this sample. You are going to direct it.

Almost nobody writes Wisej.NET code from a blank file any more — an assistant writes the first draft and the developer decides whether it is right. That only works if you know the material, which is what the rest of this module gave you. Here you hand that knowledge to ChatGPT or Claude, get this module's sample back as a working Visual Studio solution, and review it the way you would review a colleague's pull request.

What you are asking for

Run one honest translation round on GlobalDesk. Open the solution in ResX Resource Manager, add Italian as a new language column, and use the untranslated filter to list every key that still has no value. Add comments to the three keys whose meaning is ambiguous out of context, starting with the one-word Open caption, then translate or machine-draft the Italian values and mark the product name and the two technical identifiers as invariant so nobody translates them. Export the neutral and Italian columns, re-import them, and review the resulting .resx diff before you commit it as one reviewable change. Finish with a one-page workflow note comparing the Visual Studio editor, ResX Resource Manager and a cloud platform for a team whose translators are not developers.

The module asks for these deliverables, and the prompt below asks the assistant for exactly them:

- ✓ Strings.it.resx produced through ResX Resource Manager with no untranslated key left
- ✓ Resource comments on at least three ambiguous keys so a translator has the context
- ✓ Invariant markers on the product name and the identifiers that must not be translated
- ✓ Export, re-import and a reviewed.resx diff committed as one change
- ✓ Workflow note comparing the Visual Studio editor, ResX Resource Manager and a cloud TMS

Why it is shaped this way

GlobalDesk now carries a neutral Resources/Strings.resx, a German Strings.de.resx, designer resources on the customer editor and a Resources-de.resx for Wisej system labels. Adding Italian with the Visual Studio editor means opening those files one tab at a time and comparing key lists by eye, and the person who speaks Italian is usually not the person who can open the solution. This module turns translation into a workflow with a visible gap list, a handoff and a review step.

These are the terms the assistant has to get right. If what comes back uses them loosely, it has not understood the module:

Term	What it means here
Matrix view	Keys down the rows and languages across the columns, so a missing Italian value and a German term that drifted are both visible at once.
ResX Resource Manager	An open-source.NET localization tool, a Visual Studio extension and a standalone application, that shows a solution's RESX resources in one central grid.
Language column	Adding a target language creates the culture-specific file beside the neutral one and gives you an empty column to fill; the keys never change.
Untranslated filter	Lists only the entries with no value in the selected language, so "are we finished?" becomes a number instead of an opinion.
Resource comment	A note stored next to the value telling a translator whether Open is a verb, an adjective, a ticket state or a command.
Invariant string	A marker on entries that must never be translated, such as the product name and technical identifiers.
Export / import round trip	Columns sent out to a translator or a RESX-capable platform and returned as culture-specific.resx files the build treats like any other resource.

Build it — pick your assistant

Both buttons open a new tab with the **same ready-made prompt** already filled in. The prompt points the model at this module's specification — a single Markdown file holding the lesson, the lab brief and the deliverables — so it works from the real material instead of guessing what Module 6 is about. You can open [module-06.md](#) yourself to see exactly what it will read.



Use ChatGPT

Opens [chatgpt.com](#) with this module's prompt loaded. Works with GPT-5 or Codex; any plan that can browse the web will fetch the spec.



Use Claude

Opens [claude.ai](#) with the same prompt loaded. Claude Code users can paste it into a terminal in an empty folder instead.



Send it as-is and you get a project, not a snippet

The prompt ends by asking for the whole solution as a single [.zip](#) with build-and-run steps. Download it, open the solution in Visual Studio, press F5 and GlobalDesk comes up on [localhost](#). You need the .NET SDK and the [Wisej-4](#) NuGet package; everything else the prompt asks the assistant to write itself.

The prompt, in full — read it, then edit it if you want a different sample

I am working through Module 6 – "Translation Workflow: ResX Resource Manager and Other Tools" – of the LearnWisej Localization in Wisej.NET course, and I want to build the module sample with you.

Start by reading the module specification. It is a single Markdown file holding the full lesson, the lab brief and the deliverables:

<https://learnwisej.com/assets/courses/localization-in-wisej-net/specs/module-06.md>

Then build a complete Wisej.NET 4 project I can open in Visual Studio:

1. Project "GlobalDesk", C#, net8.0 or later, NuGet package Wisej-4 4.1.x, Kestrel host calling `app.UseWisej()`.
2. Build what the specification's lab asks for: Run one honest translation round on GlobalDesk. Open the solution in ResX Resource Manager, add Italian as a new language column, and use the untranslated filter to list every key that still has no value. Add comments to the three keys whose meaning is ambiguous out of context, starting with the one-word Open caption, then translate or machine-draft the Italian values and mark the product name and the two technical identifiers as invariant so nobody translates them. Export the neutral and Italian columns, re-import them, and review the resulting .resx diff before you commit it as one reviewable change. Finish with a one-page workflow note comparing the Visual Studio editor, ResX Resource Manager and a cloud platform for a team whose translators are not developers.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - Strings.it.resx produced through ResX Resource Manager with no untranslated key left
 - Resource comments on at least three ambiguous keys so a translator has the context
 - Invariant markers on the product name and the identifiers that must not be translated
 - Export, re-import and a reviewed.resx diff committed as one change
 - Workflow note comparing the Visual Studio editor, ResX Resource Manager and a cloud TMS
4. Follow the specification wherever it names a Wisej.NET class, property or event – use exactly that one. Where it leaves a choice open, tell me which API you picked and why.
5. Keep the server authoritative: state and validation live in .NET, and the browser renders and reports. Never move a decision into JavaScript because it is easier there.
6. Make the result demonstrable: I should be able to run it and see the behaviour this module is about, including what happens when something goes wrong.

Before you write any code, restate in your own words what you are going to build and where each piece will live, so I can correct the design while it is still cheap.

When the code is done, package the whole solution as a single .zip I can download, and give me the exact steps to build and run it.

Finally, answer these questions from the specification, using your own implementation as the evidence:

- Your Italian column is full and the application looks right in it-IT, but a colleague claims two keys were never translated. How could both statements be true at once, and which tool feature settles the argument?
- An Italian machine draft returned `Ultimo ordine: {1} il {0}` for a key whose neutral value is `Last order: {0} on {1}`. What breaks, when does it break, and what would have caught it before the commit?
- Your team has five languages, two of them translated by an agency that has never opened Visual Studio. Which of the three workflows would you adopt, and what would have to change before you moved to the next one up?

Ask me about anything that is ambiguous before you start.

Then review what came back

Code that compiles is not yet code that is right. The interesting questions are design questions: is this the Wisej.NET control the job actually calls for, does it follow the lifecycle the framework gives you, is the authoritative state on the server rather than parked in the browser because that was convenient? Read the result as a reviewer, not as a customer.

This module gives you the checklist. Answer each one against the code you just received:

- ✓ Your Italian column is full and the application looks right in it-IT, but a colleague claims two keys were never translated. How could both statements be true at once, and which tool feature settles the argument?
- ✓ An Italian machine draft returned Ultimo ordine: {1} il {0} for a key whose neutral value is Last order: {0} on {1}. What breaks, when does it break, and what would have caught it before the commit?
- ✓ Your team has five languages, two of them translated by an agency that has never opened Visual Studio. Which of the three workflows would you adopt, and what would have to change before you moved to the next one up?

Anything you had to correct is worth writing down. A short note of the conventions and decisions you settled on, pasted into the first message of your next session, is worth more than any single clever prompt.



No reference sample for this course yet — yours is the first

The [LearnWisej samples repository](#) carries worked builds for most courses, but not yet for this one. That means there is nothing to compare against and no shortcut: the specification and your own review are the whole safety net. Keep what you build — and if it turns out well, we would like to see it.

Three prompts to take it further

Point your assistant at the project you just built and try these. Each one is a review-and-extend exercise, not a rewrite.

- ✓ **Add the second case.** *"Extend this GlobalDesk project with a second case that follows the same pattern as the first — another entity, screen or control of the same shape, wired the same way. Factor out what is genuinely shared into a base class or service, but only where it earns its place; two similar things do not always want an abstraction. Show me the diff and justify each thing you extracted."*
- ✓ **Prove it actually works.** *"Go through this project deliverable by deliverable against the module specification at <https://learnwisej.com/assets/courses/localization-in-wisej-net/specs/module-06.md> and show me the evidence that each one is genuinely done — the code path, what to click, and what I should see. Where a deliverable is only partly met, say so plainly instead of arguing it through. Then add whatever is missing."*

- ✓ **Review it like a pull request.** *"Walk me through this GlobalDesk code as if you were reviewing a colleague's pull request and did not write it. Where does state live, what happens end to end on one user action, what would you send back for changes, and which parts would you not be comfortable putting in front of a customer? Be specific and quote the lines you mean — do not summarise the design back to me."*



Carry it into the lab

The hands-on lab that closes this module asks for the same deliverables you just had the assistant produce. Use what you have here as the shape of your answer and the evidence behind it — not as the answer itself. The lab is graded on your reasoning, and that part has to be yours.