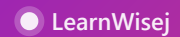


AI Coding Exercise

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.



Same module, built the way you will actually build things.

Every other section of this module explains how "Right-to-Left Languages and Localization-Safe Layouts" works. This one asks you to produce it — by directing an assistant with the module's own specification, then verifying the result runs and does what the lab brief describes. It is the workflow you will use on real work, practised on something whose right answer you already know.

What you are asking for

Make GlobalDesk survive a right-to-left culture. Add Arabic to the language picker and set `rightToLeft` to `auto` in `Default.json`, leave the child controls on `RightToLeft.Inherit`, and set `RightToLeftLayout` on the dashboard and the customer editor so their children mirror. Keep exactly one control left-to-right, the customer code field, and say in a comment why that identifier must not flip. Rebuild the fixed-position part of the layout with docking, anchoring and a container layout so a long translation cannot clip, add a pseudo-localized resource file whose values are lengthened and bracketed, and record what the contacts `DataGridView` actually does under RTL rather than assuming it mirrors. Finish with a five-case layout test list you can rerun on any screen.

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

- ✓ Arabic entry in the language picker with `rightToLeft` set to `auto` in `Default.json`
- ✓ Dashboard and customer editor using `RightToLeftLayout` with their children left on `Inherit`
- ✓ One deliberately left-to-right control with a comment explaining the exception
- ✓ Layout rebuilt with docking, anchoring and a container layout so long translations cannot clip
- ✓ RTL test list covering short text, long text, a narrow window, an RTL culture and the `DataGridView` limitation

Why it is shaped this way

GlobalDesk can already change language while it runs: the picker sets `Application.CurrentCulture`, the `Application.CultureChanged` handler reapplies every resource-backed caption, and the preview panel reformats its samples. Then a tester opens the dashboard under Arabic. Every word is translated and every one of them is in the wrong place: labels hug the left edge of their fields, the navigation column sits where an Arabic reader's eye arrives last, and the long German captions from the previous module already push one button outside a fixed-width panel.

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
RightToLeft	A control property with the values Inherit, No and Yes that tells one control to adapt its own text and internal element direction.
RightToLeft.Inherit	The normal choice for a child control: it follows its parent, so the whole screen changes direction from a single decision at the top.
RightToLeftLayout	A different property that mirrors the horizontal position of a container's direct children when RTL is active, so the visual arrangement flips while your logical layout stays as designed.
The rightToLeft application setting	An entry in Default.json; set to "auto" it makes Wisej.NET derive RTL behavior from the session's culture instead of from culture-name checks scattered through your screens.
Application.RightToLeft	The same decision expressed in code, for sessions that must be switched deliberately rather than by culture detection.
Text expansion	A translated caption is rarely the length of the English one, which is why fixed pixel positions and hand-measured panel widths break first in a localized build.
Documented RTL limits	Wisej.NET documents that DataGridView, ListView and PropertyGrid do not fully support RTL, so a test plan has to name them instead of assuming they mirror.

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 5 is about. You can open [module-05.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 5 - "Right-to-Left Languages and Localization-Safe Layouts" - 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-05.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: Make GlobalDesk survive a right-to-left culture. Add Arabic to the language picker and set `rightToLeft` to auto in `Default.json`, leave the child controls on `RightToLeft.Inherit`, and set `RightToLeftLayout` on the dashboard and the customer editor so their children mirror. Keep exactly one control left-to-right, the customer code field, and say in a comment why that identifier must not flip. Rebuild the fixed-position part of the layout with docking, anchoring and a container layout so a long translation cannot clip, add a pseudo-localized resource file whose values are lengthened and bracketed, and record what the contacts `DataGridView` actually does under RTL rather than assuming it mirrors. Finish with a five-case layout test list you can rerun on any screen.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - Arabic entry in the language picker with `rightToLeft` set to auto in `Default.json`
 - Dashboard and customer editor using `RightToLeftLayout` with their children left on `Inherit`
 - One deliberately left-to-right control with a comment explaining the exception
 - Layout rebuilt with docking, anchoring and a container layout so long translations cannot clip
 - RTL test list covering short text, long text, a narrow window, an RTL culture and the `DataGridView` limitation
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:

- A colleague sets `RightToLeft.Yes` on twenty controls and the Arabic screen looks correct, but the German screen is unchanged. What did they actually configure, and which single property would have done the job for the whole container?
- Your pseudo-localized pass shows three captions still in plain English. Nothing clipped and nothing threw. What does that tell you about those three captions, and which earlier module's work has a gap?
- The contacts grid does not mirror the way the rest of the dashboard does. Which of your

options — accept it and document it, replace the control, or write mirroring by hand — can you defend to a reviewer, and what would you have to verify before promising any of them?

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:

- ✓ A colleague sets `RightToLeft.Yes` on twenty controls and the Arabic screen looks correct, but the German screen is unchanged. What did they actually configure, and which single property would have done the job for the whole container?
- ✓ Your pseudo-localized pass shows three captions still in plain English. Nothing clipped and nothing threw. What does that tell you about those three captions, and which earlier module's work has a gap?
- ✓ The contacts grid does not mirror the way the rest of the dashboard does. Which of your options — accept it and document it, replace the control, or write mirroring by hand — can you defend to a reviewer, and what would you have to verify before promising any of them?

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.

- ✓ **Make it survive a bad day.** *"Review this GlobalDesk project for the failure modes it does not handle yet — bad or missing input, a dependency that is slow or unavailable, work arriving after the user navigated away, and two actions racing each other. For each one, tell me first whether it is a real risk in Wisej.NET 4 and why, and only then write the fix. Where you decide something needs no handling, say so and explain the reasoning rather than adding defensive code."*
- ✓ **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-05.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."*



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.