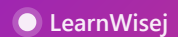


AI Coding Exercise

Build this module's OperationsConsole 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 — including against our own reference build on GitHub.



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

Create the Operations Console control catalog shell: a main Page with a navigation area, a content area, a command area and a status area; placeholder pages for Editors, Layouts, Lists and Trees, DataGridView, Dashboard and Widgets; a ControlSelection.md note inside the project listing which control family you will use for each area; clear control names throughout; and a status area that updates whenever navigation changes.

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

- ✓ Main shell with navigation, content, command and status areas
- ✓ Placeholder pages for Editors, Layouts, Lists and Trees, DataGridView, Dashboard and Widgets
- ✓ ControlSelection.md listing the control family chosen for each area
- ✓ Clear control names and a status area that updates on navigation
- ✓ Screenshot and a short note explaining the control decisions

Why it is shaped this way

Wisej.NET ships more than a hundred controls, and a team without a shared mental model of them ends up with twenty different search screens, five ways of reporting a validation error, and custom JavaScript where a native control would do. This module fixes the vocabulary, decision rules and conventions every later module builds on, and starts the Operations Console shell.

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
server control tree	The hierarchy of .NET control objects the server keeps for each session; your code sets properties and handles events here, never in markup.
browser widget	The JavaScript object Wisej.NET creates to render a server control, kept in sync by serialized state changes, not page reloads.
control family	A group of controls sharing a purpose (editors, containers, lists, grids, content, menus, notifications), where decisions are made.
extender	A component that adds properties or behaviour to existing controls at design time without subclassing them.
Widget and extension	The Widget control hosts a custom or third-party JavaScript widget inside the Wisej.NET event model; an extension is a packaged integration, usually from NuGet.
UserControl	A reusable server-side composition of controls with business meaning and a small public surface.
control selection guide	A living team document fixing standard controls, docking, validation and notification style and grid conventions per screen pattern.

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 1 is about. You can open [module-01.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 OperationsConsole 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 1 – "Control Library Mental Model" – of the LearnWisej Mastering the Control Library 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/mastering-the-control-library/specs/module-01.md>

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

1. Project "OperationsConsole", 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: Create the Operations Console control catalog shell: a main Page with a navigation area, a content area, a command area and a status area; placeholder pages for Editors, Layouts, Lists and Trees, DataGridView, Dashboard and Widgets; a `ControlSelection.md` note inside the project listing which control family you will use for each area; clear control names throughout; and a status area that updates whenever navigation changes.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - Main shell with navigation, content, command and status areas
 - Placeholder pages for Editors, Layouts, Lists and Trees, DataGridView, Dashboard and Widgets
 - `ControlSelection.md` listing the control family chosen for each area
 - Clear control names and a status area that updates on navigation
 - Screenshot and a short note explaining the control decisions
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:

- When you set `statusLabel.Text` in the navigation method, what actually travels to the browser, and why is that different from writing HTML?
- Which shell area would you turn into a `UserControl` first, and what would its public surface be?
- If a later module needs a collapsible navigation area, which native container replaces the docked panel, and why is that preferable to a custom widget?

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:

- ✓ When you set `statusLabel.Text` in the navigation method, what actually travels to the browser, and why is that different from writing HTML?
- ✓ Which shell area would you turn into a `UserControl` first, and what would its public surface be?
- ✓ If a later module needs a collapsible navigation area, which native container replaces the docked panel, and why is that preferable to a custom widget?

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.



We already built this one — compare yours with ours

The Module 1 reference implementation lives in the public samples repository. Build yours first, then open ours: the differences are the interesting part.



[Open the Module 1 sample on GitHub](#)

Or clone the whole set with `git clone https://github.com/iceteagroup/LearnWisej-Samples` and open `Mastering the Control Library Course/Module 1`.



Our sample is a starting point, not a finished product

It was built the same way you are building yours — by prompt, then corrected — and it deliberately stops where the module stops. Expect stand-ins where production code would go, thin error handling and no tests. Treat that as an invitation: use what you learned in this module, plus the assistant, to push it further. Improving code you did not write is the skill this course is actually teaching.

Three prompts to take it further

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

- ✓ **Make it survive a bad day.** *"Review this OperationsConsole 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 OperationsConsole 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/mastering-the-control-library/specs/module-01.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.