

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.

● LearnWisej



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

Every other section of this module explains how "DataGridView Mastery" 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

Build the Operations Console Orders grid: a DataGridView bound to a BindingSource with explicit columns and AutoGenerateColumns under control, a status cell rendered with AllowHtml and CellFormatting, one custom editor or command column, a virtual-mode or cache-backed version for the large-data path, and a composed filter and status strip docked inside or around the grid.

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

- ✓ Orders DataGridView with explicit columns bound to a BindingSource
- ✓ Status cell rendered with AllowHtml and CellFormatting
- ✓ One custom editor or command column
- ✓ Virtual-mode or cache-backed large-data version of the grid
- ✓ Composed filter and status strip docked inside or around the grid

Why it is shaped this way

DataGridView is the control most likely to decide whether a business application feels mature. Orders, invoices, deadlines, logs, users and documents all end up in a grid, and users judge the app by how that grid loads, scrolls, edits and reports status. Wisej.NET offers everything from manual population to virtual mode and cell painting; mastery is choosing the right technique for the data size and the user's task.

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
BindingSource	A server-side intermediary between view-model rows and the grid, giving it a stable data source, currency and sorting without coupling the screen to the database.
AutoGenerateColumns	Creates a column for every public property of the bound type; handy for a prototype, dangerous in production because a model change silently rearranges the screen.
explicit column	A column you define with field binding, header text, width, alignment, format, read-only state, sort mode and editor type, so the UI stays stable when the model changes.
virtual mode	With VirtualMode on, the grid knows only its row count and asks for each visible value through CellValueNeeded, so nothing is created for rows nobody sees.
server-side cache	A page-sized buffer that feeds CellValueNeeded so scrolling does not hit the database per cell; the DataRead event is the documented hook for building and maintaining it.
custom cell editor	A control assigned to a column's Editor property; it must push its value back into the cell, via CellBeginEdit and CellEndEdit if it has no suitable change event.
cell HTML vs cell control	HTML produced in CellFormatting with AllowHtml is display-only markup and nearly free; an embedded control is a real server control and browser widget with a cost per cell.

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 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 5 - "DataGridView Mastery" - 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-05.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: Build the Operations Console Orders grid: a DataGridView bound to a BindingSource with explicit columns and AutoGenerateColumns under control, a status cell rendered with AllowHtml and CellFormatting, one custom editor or command column, a virtual-mode or cache-backed version for the large-data path, and a composed filter and status strip docked inside or around the grid.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - Orders DataGridView with explicit columns bound to a BindingSource
 - Status cell rendered with AllowHtml and CellFormatting
 - One custom editor or command column
 - Virtual-mode or cache-backed large-data version of the grid
 - Composed filter and status strip docked inside or around the grid
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:

- If OrderRow gains two new properties next sprint, what changes on the screen with generated columns, and what changes with your explicit columns?
- Your CellValueNeeded handler works with 200 orders. What happens to the database when a user drags the scrollbar through 200,000, and which part of your design absorbs that?
- Why does the status badge belong in CellFormatting rather than in the OrderRow model, and what would go wrong if a customer name were placed in that HTML without encoding?

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:

- ✓ If OrderRow gains two new properties next sprint, what changes on the screen with generated columns, and what changes with your explicit columns?
- ✓ Your CellValueNeeded handler works with 200 orders. What happens to the database when a user drags the scrollbar through 200,000, and which part of your design absorbs that?
- ✓ Why does the status badge belong in CellFormatting rather than in the OrderRow model, and what would go wrong if a customer name were placed in that HTML without encoding?

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



[Open the Module 5 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 5`.



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.

- ✓ **Swap a piece without moving the seams.** *"Pick the part of this OperationsConsole project that is most obviously a stand-in for something real — a fake data source, an in-memory store, a simplified service — and replace it with a production-shaped implementation. The contract either side of it must not change: same types, same calls, same behaviour under failure. Tell me what the real implementation forced you to handle that the stand-in did not."*
- ✓ **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."*



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.