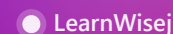


What Wisej.NET Is (and When to Use It)

A clear picture of the framework before you write any code.



Lesson goal

By the end of this reading you'll be able to explain what Wisej.NET is, describe its mental model, and decide when it's the right tool for a project.

1. What Wisej.NET is

Wisej.NET is a framework and development environment for building **real-time web applications** using the .NET languages and the Visual Studio workflow that many desktop developers already know. A Wisej.NET app runs in a browser, but the application logic is written mostly in **C# or VB.NET on the server**.

A useful first mental model: **Wisej.NET feels like building a desktop-style application, but the finished app is delivered through the web browser**. Instead of hand-writing every piece of HTML, CSS and JavaScript, you work with Wisej controls, properties, events and server-side code.



Plain-English definition

Wisej.NET lets a .NET developer build web apps with visual design tools, controls and event-driven C# code — instead of starting from a front-end JavaScript framework.

2. How it works

In a typical Wisej.NET application the browser shows the user interface while the server manages the application objects and event logic. When a user clicks a button, edits a field, or selects a row, the event is handled by C# code on the server. Wisej keeps the browser and server synchronized so the interface updates seamlessly.

Part	What it means for a beginner
Browser	Where the user sees and interacts with the app — pages, buttons, labels, grids and forms.
Server	Runs the .NET application code. Button clicks and most events are handled here.
Designer	The visual surface in Visual Studio where you arrange controls and set properties.
Controls	Reusable UI blocks: <code>Button</code> , <code>Label</code> , <code>TextBox</code> , <code>DataGridView</code> , <code>Panel</code> , <code>Page</code> , <code>Window</code> .
Events	Things that happen — <code>Click</code> , <code>TextChanged</code> , <code>SelectedIndexChanged</code> — which run your code.

3. When to use Wisej.NET

Wisej.NET shines when the project is a **business application** and the team wants fast development with .NET, Visual Studio and a visual designer — especially when the app is form-heavy, data-heavy, or desktop-like.

Good fit	Why Wisej.NET helps
Internal business tools	Dashboards, forms, admin screens and workflow tools, built quickly.
CRUD applications	Controls map naturally onto create / read / update / delete records.
DataGridView-heavy apps	Data tables, record lists and detail views are core Wisej patterns.
Desktop modernization	WinForms-style experience carries over while moving to the web.
Rapid prototypes	The Designer and event model produce working screens fast.

4. When *not* to use it

- Use a traditional website / CMS for mostly public, SEO-focused content pages.
- Use a front-end framework when you specifically need a custom JavaScript component architecture.
- Use a static page generator when there's no real business workflow or database interaction.
- Use native mobile development when deep device integration and app-store distribution are the main goal.

5. The Wisej.NET mental model

Think in terms of **screens, controls, properties and events**. A screen contains controls. Controls have names and properties. Users interact with controls. Events run code — and that code can update the UI, call a service, save data, or navigate.

Example flow

```
User clicks
→ the btnSave_Click event runs
→ it reads values from the text boxes
→ it calls a service method
→ the service saves the data
→ the UI shows a status message
```



Beginner habit

Don't put all your business logic inside the button click. Keep the click readable and call a helper method, service or repository class when the logic grows.

Key vocabulary

- **Project** — the whole app in Visual Studio: code, designer files, references and configuration.
- **Page / Window** — a screen or view shown to the user (Windows are often dialog-style).
- **Control** — a UI object such as a button, label, textbox, grid or checkbox.
- **Property** — a setting on a control: `Text` , `Name` , `Dock` , `Size` , `Visible` .
- **Event** — something that happens (a click, a selection change) that runs code.
- **Code-behind** — the normal `.cs` file where behavior lives.
- **Service class** — a separate class holding business logic and data access, away from the UI.