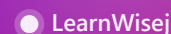


Che cos'è Wisej.NET (e quando usarlo)

Un'idea chiara del framework prima di scrivere una sola riga di codice.



Obiettivo della lezione

Alla fine di questa lettura saprai spiegare che cos'è Wisej.NET, descriverne il modello mentale e decidere quando è lo strumento giusto per un progetto.

1. Che cos'è Wisej.NET

Wisej.NET è un framework e un ambiente di sviluppo per costruire **applicazioni web in tempo reale** con i linguaggi .NET e il flusso di lavoro di Visual Studio che molti sviluppatori desktop conoscono già. Un'app Wisej.NET gira in un browser, ma la logica applicativa è scritta per lo più in **C# o VB.NET sul server**.

Un primo modello mentale utile: **con Wisej.NET sembra di costruire un'applicazione in stile desktop, ma l'app finita viene distribuita tramite il browser web**. Invece di scrivere a mano ogni pezzo di HTML, CSS e JavaScript, lavori con controlli, proprietà ed eventi di Wisej.NET e con codice lato server.



Definizione in parole semplici

Wisej.NET permette a uno sviluppatore .NET di costruire app web con strumenti di progettazione visuale, controlli e codice C# guidato dagli eventi — invece di partire da un framework JavaScript front-end.

2. Come funziona

In una tipica applicazione Wisej.NET il browser mostra l'interfaccia utente mentre il server gestisce gli oggetti dell'applicazione e la logica degli eventi. Quando un utente fa clic su un pulsante, modifica un campo o seleziona una riga, l'evento viene gestito da codice C# sul server. Wisej.NET mantiene browser e server sincronizzati, così l'interfaccia si aggiorna senza interruzioni.

Parte	Che cosa significa per chi inizia
Browser	Dove l'utente vede l'app e interagisce con essa — pagine, pulsanti, label, griglie e form.
Server	Esegue il codice dell'applicazione .NET. I click sui pulsanti e la maggior parte degli eventi vengono gestiti qui.
Designer	La superficie visuale in Visual Studio in cui disponi i controlli e imposti le proprietà.
Controlli	Blocchi di interfaccia riutilizzabili: <code>Button</code> , <code>Label</code> , <code>TextBox</code> , <code>DataGridView</code> , <code>Panel</code> , <code>Page</code> , <code>Window</code> .
Eventi	Cose che succedono — <code>Click</code> , <code>TextChanged</code> , <code>SelectedIndexChanged</code> — e che eseguono il tuo codice.

3. Quando usare Wisej.NET

Wisej.NET dà il meglio quando il progetto è un'**applicazione di business** e il team vuole sviluppare in fretta con .NET, Visual Studio e un designer visuale — soprattutto quando l'app è ricca di form, di dati o simile a un'app desktop.

Casi adatti	Perché Wisej.NET aiuta
Strumenti aziendali interni	Dashboard, form, schermate di amministrazione e strumenti di workflow, costruiti rapidamente.
Applicazioni CRUD	I controlli si adattano in modo naturale alla creazione / lettura / modifica / eliminazione dei record.
App basate sulla DataGridView	Tabelle di dati, elenchi di record e viste di dettaglio sono pattern fondamentali di Wisej.NET.
Modernizzazione di app desktop	L'esperienza in stile WinForms si conserva nel passaggio al web.
Prototipi rapidi	Il Designer e il modello a eventi producono in fretta schermate funzionanti.

4. Quando *non* usarlo

- Usa un sito web tradizionale / un CMS per pagine di contenuto per lo più pubbliche e orientate alla SEO.
- Usa un framework front-end quando ti serve espressamente un'architettura di componenti JavaScript personalizzata.
- Usa un generatore di pagine statiche quando non c'è un vero flusso di business né un'interazione con un database.
- Usa lo sviluppo mobile nativo quando l'obiettivo principale è un'integrazione profonda con il dispositivo e la distribuzione tramite gli app store.

5. Il modello mentale di Wisej.NET

Ragiona in termini di **schermate, controlli, proprietà ed eventi**. Una schermata contiene controlli. I controlli hanno nomi e proprietà. Gli utenti interagiscono con i controlli. Gli eventi eseguono codice — e quel codice può aggiornare l'interfaccia, chiamare un servizio, salvare dati o navigare.

Flusso di esempio

```
L'utente fa clic su
→ viene eseguito l'evento btnSave_Click
→ legge i valori dalle textbox
→ chiama un metodo di servizio
→ il servizio salva i dati
→ l'interfaccia mostra un messaggio di stato
```



Abitudine per chi inizia

Non mettere tutta la logica di business dentro il click del pulsante. Mantieni il click leggibile e chiama un metodo helper, un servizio o una classe repository quando la logica cresce.

Vocabolario chiave

- **Progetto** — l'intera app in Visual Studio: codice, file del Designer, riferimenti e configurazione.
- **Page / Window** — una schermata o vista mostrata all'utente (le Window hanno spesso lo stile di una finestra di dialogo).
- **Controllo** — un oggetto dell'interfaccia come un pulsante, una label, una textbox, una griglia o una checkbox.
- **Proprietà** — un'impostazione di un controllo: `Text`, `Name`, `Dock`, `Size`, `Visible`.
- **Evento** — qualcosa che succede (un click, un cambio di selezione) e che esegue codice.
- **Code-behind** — il normale file `.cs` in cui vive il comportamento.
- **Classe di servizio** — una classe separata che contiene la logica di business e l'accesso ai dati, lontano dall'interfaccia.