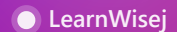


Was Wisej.NET ist (und wann Sie es einsetzen)

Ein klares Bild vom Framework, bevor Sie die erste Zeile Code schreiben.



Lernziel

Nach dieser Lektüre können Sie erklären, was Wisej.NET ist, sein mentales Modell beschreiben und entscheiden, wann es das richtige Werkzeug für ein Projekt ist.

1. Was Wisej.NET ist

Wisej.NET ist ein Framework und eine Entwicklungsumgebung zum Erstellen von **Echtzeit-Webanwendungen** mit den .NET-Sprachen und dem Visual-Studio-Workflow, den viele Desktop-Entwickler bereits kennen. Eine Wisej.NET-App läuft im Browser, die Anwendungslogik schreiben Sie jedoch überwiegend in **C# oder VB.NET auf dem Server**.

Ein hilfreiches erstes mentales Modell: **Mit Wisej.NET entwickeln Sie wie eine Desktop-Anwendung, ausgeliefert wird die fertige App aber über den Webbrowser.** Statt jedes Stück HTML, CSS und JavaScript von Hand zu schreiben, arbeiten Sie mit Wisej.NET-Controls, Eigenschaften, Events und serverseitigem Code.



Einfach erklärt

Mit Wisej.NET erstellt ein .NET-Entwickler Web-Apps mit visuellen Design-Werkzeugen, Controls und ereignisgesteuertem C#-Code – statt mit einem Front-End-JavaScript-Framework zu beginnen.

2. So funktioniert es

In einer typischen Wisej.NET-Anwendung zeigt der Browser die Benutzeroberfläche an, während der Server die Anwendungsobjekte und die Event-Logik verwaltet. Klickt ein Benutzer auf einen Button, bearbeitet ein Feld oder wählt eine Zeile aus, wird das Event von C#-Code auf dem Server verarbeitet. Wisej.NET hält Browser und Server synchron, sodass sich die Oberfläche nahtlos aktualisiert.

Baustein	Was er für Einsteiger bedeutet
Browser	Hier sieht und bedient der Benutzer die App – Seiten, Buttons, Labels, Grids und Formulare.
Server	Führt den .NET-Anwendungscode aus. Button-Klicks und die meisten Events werden hier verarbeitet.
Designer	Die visuelle Oberfläche in Visual Studio, auf der Sie Controls anordnen und Eigenschaften festlegen.
Controls	Wiederverwendbare UI-Bausteine: <code>Button</code> , <code>Label</code> , <code>TextBox</code> , <code>DataGridView</code> , <code>Panel</code> , <code>Page</code> , <code>Window</code> .
Events	Ereignisse wie <code>Click</code> , <code>TextChanged</code> oder <code>SelectedIndexChanged</code> , die Ihren Code ausführen.

3. Wann Sie Wisej.NET einsetzen

Wisej.NET spielt seine Stärken aus, wenn das Projekt eine **Geschäftsanwendung** ist und das Team schnell mit .NET, Visual Studio und einem visuellen Designer entwickeln möchte – besonders bei formularlastigen, datenintensiven oder desktopähnlichen Apps.

Gut geeignet	Warum Wisej.NET hilft
Interne Business-Tools	Dashboards, Formulare, Admin-Masken und Workflow-Tools, schnell erstellt.
CRUD-Anwendungen	Controls lassen sich ganz natürlich auf das Anlegen, Lesen, Aktualisieren und Löschen von Datensätzen abbilden.
Apps mit vielen DataGridViews	Datentabellen, Datensatzlisten und Detailansichten sind zentrale Wisej.NET-Muster.
Desktop-Modernisierung	Das WinForms-Erlebnis bleibt beim Umstieg ins Web erhalten.
Schnelle Prototypen	Designer und Event-Modell liefern zügig funktionierende Masken.

4. Wann Sie es *nicht* einsetzen

- Für überwiegend öffentliche, SEO-orientierte Inhaltsseiten nehmen Sie eine klassische Website bzw. ein CMS.
- Ein Front-End-Framework ist die bessere Wahl, wenn Sie gezielt eine eigene JavaScript-Komponentenarchitektur brauchen.
- Ein statischer Seitengenerator genügt, wenn es keinen echten Geschäftsablauf und keine Datenbankzugriffe gibt.
- Setzen Sie auf native Mobile-Entwicklung, wenn tiefe Geräteintegration und die Verteilung über App-Stores im Vordergrund stehen.

5. Das mentale Modell von Wisej.NET

Denken Sie in **Masken, Controls, Eigenschaften und Events**. Eine Maske enthält Controls. Controls haben Namen und Eigenschaften. Benutzer interagieren mit Controls. Events führen Code aus – und dieser Code kann die Oberfläche aktualisieren, einen Service aufrufen, Daten speichern oder navigieren.

Beispielablauf

```
Benutzer klickt auf
→ das Event btnSave_Click wird ausgeführt
→ es liest die Werte aus den Textfeldern
→ es ruft eine Service-Methode auf
→ der Service speichert die Daten
→ die Oberfläche zeigt eine Statusmeldung
```



Gute Gewohnheit für Einsteiger

Packen Sie nicht die gesamte Geschäftslogik in den Button-Klick. Halten Sie den Klick-Handler lesbar und rufen Sie eine Hilfsmethode, einen Service oder eine Repository-Klasse auf, sobald die Logik wächst.

Wichtige Begriffe

- **Projekt** – die gesamte App in Visual Studio: Code, Designer-Dateien, Verweise und Konfiguration.
- **Page / Window** – eine Maske oder Ansicht, die dem Benutzer angezeigt wird (Windows sind oft dialogartig).
- **Control** – ein UI-Objekt wie Button, Label, Textfeld, Grid oder Checkbox.
- **Eigenschaft** – eine Einstellung an einem Control: `Text`, `Name`, `Dock`, `Size`, `Visible`.
- **Event** – etwas, das passiert (ein Klick, eine geänderte Auswahl) und Code ausführt.
- **Code-Behind** – die normale `.cs`-Datei, in der das Verhalten steckt.
- **Service-Klasse** – eine separate Klasse mit Geschäftslogik und Datenzugriff, getrennt von der Oberfläche.