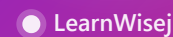


# Widget complessi in data binding: pattern generali con esempi Kendo e DevExtreme

[Guida della lezione](#) · [Guida al lab / esame](#)

## Guida della lezione

Integrare widget dati ricchi senza rendere il corso dipendente da un'unica libreria di un fornitore. Che cosa significa per uno sviluppatore che integra con Wisej.NET — e come affrontare questo modulo.



### Focus di produzione

Integrare widget dati ricchi senza rendere il corso dipendente da un'unica libreria di un fornitore.

## 1. Il problema di integrazione

Librerie avanzate come Kendo UI e DevExtreme offrono griglie, pivot grid, fogli di calcolo e data store potenti, ma ognuna ha la propria API, il proprio ciclo di vita e il proprio modello di operazioni sui dati.

## 2. Concetti chiave

| Concetto                    | Come usarlo                                                                               |
|-----------------------------|-------------------------------------------------------------------------------------------|
| Operazioni sui dati         | Mappa paginazione, ordinamento, filtri, raggruppamento e modifica su gestori lato server. |
| Sorgente dati del fornitore | Considera Kendo/DevExtreme come esempi di un pattern di integrazione generale.            |

## 3. La lezione di produzione

Kendo UI e DevExtreme sono esempi avanzati perché contengono widget ricchi con un vero comportamento sui dati. Non sono la definizione dell'integrazione in Wisej.NET. Gli stessi pattern valgono per qualunque libreria seria di widget JavaScript: caricare le risorse nell'ordine corretto, creare il widget nel contenitore giusto, passare le opzioni, collegare gli eventi, fornire i dati, gestire le operazioni remote ed esporre un'API .NET adatta all'applicazione di destinazione. Per griglie e pivot grid la decisione di design più importante è il contratto dei dati. Un pivot in sola lettura può restituire un array JSON o una risposta in stile OLAP. Una griglia modificabile ha bisogno delle operazioni di load, insert, update e delete. Paginazione e filtri remoti richiedono modelli di richiesta. La serializzazione del layout richiede un oggetto di stato separato. L'esportazione richiede una funzionalità del fornitore lato client oppure una pipeline lato server. Kendo e DevExtreme hanno API diverse, ma le responsabilità dell'adapter Wisej.NET sono le stesse.

## Progettare prima il contratto dei dati

Un widget ricco come una griglia o un pivot è definito dalle operazioni che esegue sui dati, quindi il contratto si progetta prima di scrivere qualunque codice del fornitore. Una vista in sola lettura ha bisogno di un'operazione di load che restituisca righe. Una vista paginata ha bisogno di skip e take, oppure di un indice e di una dimensione di pagina. Una vista ordinata o filtrata ha bisogno di una descrizione dell'ordinamento e del filtro applicati dal widget. Una vista raggruppata restituisce aggregati, e un pivot restituisce una struttura organizzata per campi di riga e di colonna. Una vista modificabile aggiunge insert, update e delete, ciascuna con una chiave e i valori modificati.

Il lab lo rappresenta sul server con una `GridOperationRequest` tipizzata, che contiene skip, take, sort e filter, e con un risultato che contiene le righe e il conteggio totale. Ogni libreria di fornitore, comunque chiami queste cose, può essere mappata su quella richiesta. Progettare prima la richiesta in .NET mantiene il server in controllo di che cosa può contenere un filtro e di quanto può essere grande una pagina.

Lo stato del layout, come larghezze e visibilità delle colonne, è un contratto separato, memorizzato per utente e mai mescolato con le operazioni sui dati.

## Mappare le operazioni del fornitore sugli endpoint

Il DataSource di Kendo e il CustomStore di DevExtreme differiscono nella sintassi, ma entrambi chiamano un URL o una funzione per il load, ed entrambi chiamano di nuovo per insert, update e delete quando le operazioni remote sono abilitate. Il compito del wrapper è tradurre. Per una sorgente guidata da URL, l'InitScript indirizza il fornitore all'URL di postback e il gestore `WebRequest` legge i nomi degli argomenti del fornitore, li converte nella `GridOperationRequest` e restituisce il risultato nella forma che il fornitore si aspetta.

Per una sorgente guidata da funzioni, le funzioni dell'InitScript chiamano un `[WebMethod]` con argomenti tipizzati e restituiscono al fornitore il risultato atteso. In entrambi i casi il lato server è lo stesso gestore con la stessa validazione, e cambiare fornitore cambia solo la traduzione nell'InitScript.

Abilitare le operazioni remote è il momento in cui la griglia diventa una vera integrazione. Fino ad allora il fornitore ordina e pagina in memoria e il server viene interpellato una sola volta; da quel momento in poi, ogni interazione dell'utente è una chiamata al server che deve essere veloce e sicura.

## Opzioni JSON o proprietà tipizzate

Sia Kendo sia DevExtreme si configurano con grandi oggetti di opzioni, e il modo più rapido per portare una griglia sullo schermo è assegnare un oggetto del genere direttamente a Options da C#. È la scelta giusta per il prototipo del pivot in sola lettura del lab, dove l'obiettivo è dimostrare che il widget gira e che il contratto dei dati funziona.

Un controllo griglia riutilizzabile non può restare così. Un contenitore di opzioni non tipizzato non ha validazione, né valori predefiniti, né testo di aiuto, e non permette di sapere quali delle sue cento chiavi l'applicazione usi davvero. Proprietà tipizzate per le impostazioni da cui dipende l'applicazione, come dimensione della pagina, modificabilità e definizioni delle colonne, danno al Designer qualcosa da mostrare e ai revisori qualcosa da controllare. Le restanti opzioni del fornitore possono restare dietro un punto di estensione protected per i rari casi che ne hanno bisogno.

Il compromesso è velocità oggi contro manutenibilità domani. Fai il prototipo con JSON, poi promuovi a proprietà tipizzata ogni opzione che sopravvive, prima che il controllo arrivi ad altri sviluppatori.

## Due fornitori, un solo pattern

Kendo e DevExtreme compaiono in questo corso perché sono molto diffusi e perché le loro differenze rendono visibile il pattern comune. Uno usa un DataSource con impostazioni di transport, l'altro un CustomStore con funzioni di load e modifica. Uno chiama la paginazione skip e take, l'altro la chiama in modo simile ma dà una forma diversa all'espressione di filtro. Una volta mappati entrambi sulla stessa `GridOperationRequest`, le differenze si riducono a poche righe di traduzione in ciascun InitScript.

È questa la lezione da portare in qualunque altra libreria. Le responsabilità dell'adapter non cambiano mai: caricare le risorse in ordine, creare il widget nel contenitore, passare le opzioni, collegare i pochi eventi che contano, fornire i dati tramite un contratto controllato dal server, gestire le operazioni remote ed esporre un'API .NET dimensionata per l'applicazione. Un team che sa farlo per due librerie sa farlo per una terza senza bisogno di un nuovo corso.

Tieni le conoscenze specifiche del fornitore nell'InitScript e nella traduzione degli eventi, e mantieni il contratto del server indipendente dal fornitore, così che un cambio di licenza o di roadmap non costringa mai a riscrivere il codice dell'applicazione.



### Regola di produzione

Tieni ben distinti la proprietà dello stato in Wisej.NET e il comportamento del widget client. Documenta il contratto client/server prima di affidarti all'integrazione in produzione.



### Errore comune da evitare

Copiare nelle schermate il JavaScript delle demo senza documentare ciclo di vita, risorse, contratti degli eventi, pulizia o confini di sicurezza.



### Obiettivi di apprendimento

Al termine di questo modulo sarai in grado di:

- ✓ Progettare contratti dei dati per dati in sola lettura, modificabili, paginati, filtrati, ordinati, raggruppati e in pivot.
- ✓ Mappare le operazioni della sorgente dati del fornitore su gestori di postback o WebMethod di Wisej.NET.
- ✓ Capire perché le opzioni JSON sono rapide ma le proprietà .NET tipizzate sono più sicure per i controlli riutilizzabili.
- ✓ Usare Kendo e DevExtreme come implementazioni di riferimento degli stessi pattern di integrazione generali.

## Guida al lab / esame

---

Che cosa costruirai nel lab pratico, il flusso di lavoro consigliato, i deliverable richiesti e le domande di autovalutazione prima della consegna.



### Lab pratico

Costruisci due esempi avanzati: un widget in stile pivot in sola lettura e un widget in stile griglia modificabile. Uno può usare la semantica DataSource in stile Kendo, l'altro la semantica CustomStore in stile DevExtreme.

## Flusso di lavoro consigliato

- ✓ Apri il progetto esistente del corso di integrazione Wisej.NET ed esegilo in locale.
- ✓ Individua il componente server, il widget client, il file di risorse o l'endpoint coinvolti in questo modulo.
- ✓ Fai una piccola modifica alla volta e provala nel browser prima di andare avanti.
- ✓ Tieni il JavaScript specifico del fornitore dentro l'adapter o il wrapper invece di spargerlo tra le pagine.
- ✓ Aggiorna la UI in modo che si vedano chiaramente successo, errore, avanzamento o cambiamenti nella comunicazione.
- ✓ Raccogli screenshot o note che dimostrano che l'integrazione funziona e spiegano la scelta progettuale.

## Deliverable richiesti

- ✓ Piano di integrazione del pivot in sola lettura
- ✓ Contratto delle operazioni della griglia modificabile
- ✓ Gestori lato server (o stub) per load/update/insert/delete
- ✓ Contratti dei payload degli eventi per clic su cella e aggiornamento di riga

## Domande di autovalutazione

- ✓ Perché le griglie sono più difficili dei gauge?
- ✓ Che cosa cambia quando le operazioni remote sono abilitate?
- ✓ Perché Kendo e DevExtreme vanno insegnati come esempi e non come l'intero corso?