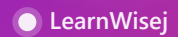


# Komplexe datengebundene Widgets: allgemeine Muster mit Beispielen für Kendo und DevExtreme

Lektionsleitfaden · Lab- und Prüfungsleitfaden



## Lektionsleitfaden

Umfangreiche Daten-Widgets integrieren, ohne den Kurs von einer einzigen Anbieterbibliothek abhängig zu machen. Was das für Entwickler bedeutet, die mit Wisej.NET integrieren – und wie Sie dieses Modul lesen sollten.



### Fokus: Produktivbetrieb

Umfangreiche Daten-Widgets integrieren, ohne den Kurs von einer einzigen Anbieterbibliothek abhängig zu machen.

## 1. Das Integrationsproblem

Anspruchsvolle Bibliotheken wie Kendo UI und DevExtreme bieten leistungsfähige Grids, Pivot-Grids, Tabellenkalkulationen und Datenspeicher, doch jede hat ihre eigene API, ihren eigenen Lebenszyklus und ihr eigenes Modell für Datenoperationen.

## 2. Zentrale Konzepte

| Konzept                   | So setzen Sie es ein                                                               |
|---------------------------|------------------------------------------------------------------------------------|
| Datenoperationen          | Paging, Sortieren, Filtern, Gruppieren und Bearbeiten auf Server-Handler abbilden. |
| Datenquelle des Anbieters | Kendo/DevExtreme als Beispiele für ein allgemeines Integrationsmuster behandeln.   |

## 3. Lektion aus der Praxis

Kendo UI und DevExtreme sind anspruchsvolle Beispiele, weil sie umfangreiche Widgets mit echtem Datenverhalten enthalten. Sie sind aber nicht die Definition der Wisej.NET-Integration. Dieselben Muster gelten für jede ernstzunehmende JavaScript-Widget-Bibliothek: Ressourcen in der richtigen Reihenfolge laden, das Widget im richtigen Container erzeugen, Optionen übergeben, Events verdrahten, Daten liefern, Remote-Operationen verarbeiten und eine .NET-API bereitstellen, die zur Zielanwendung passt. Bei Grids und Pivot-Grids ist der Datenvertrag die wichtigste Designentscheidung. Ein schreibgeschütztes Pivot kann ein JSON-Array oder eine Antwort im OLAP-Stil zurückgeben. Ein bearbeitbares Grid braucht die Operationen Laden, Einfügen, Aktualisieren und Löschen. Remote-Paging und -Filtern erfordern Abfragemodelle. Die Serialisierung des Layouts erfordert ein eigenes Zustandsobjekt. Der Export erfordert entweder clientseitige Funktionen des Anbieters oder eine serverseitige Pipeline. Kendo und DevExtreme haben unterschiedliche APIs, doch die Aufgaben des Wisej.NET-Adapters sind dieselben.

## Zuerst den Datenvertrag entwerfen

Ein umfangreiches Grid- oder Pivot-Widget wird durch die Operationen bestimmt, die es auf Daten ausführt. Deshalb entsteht der Vertrag, bevor auch nur eine Zeile Anbietercode geschrieben wird. Eine schreibgeschützte Ansicht braucht eine Ladeoperation, die Zeilen liefert. Eine seitenweise Ansicht braucht Skip und Take oder einen Seitenindex und eine Seitengröße. Eine sortierte oder gefilterte Ansicht braucht eine Beschreibung der Sortierung und des Filters, die das Widget angewendet hat. Eine gruppierte Ansicht liefert Aggregate, ein Pivot eine Struktur, die nach Zeilen- und Spaltenfeldern gegliedert ist. Eine bearbeitbare Ansicht ergänzt Einfügen, Aktualisieren und Löschen, jeweils mit einem Schlüssel und den geänderten Werten.

Das Lab hält dies auf dem Server als typisierten `GridOperationRequest` fest, der Skip, Take, Sortierung und Filter enthält, sowie als Ergebnis mit den Zeilen und der Gesamtanzahl. Jede Anbieterbibliothek lässt sich – wie auch immer sie diese Dinge nennt – auf diese Anfrage abbilden. Wenn Sie die Anfrage zuerst in .NET entwerfen, behält der Server die Kontrolle darüber, was ein Filter enthalten darf und wie groß eine Seite sein darf.

Der Layout-Zustand, etwa Spaltenbreiten und sichtbare Spalten, ist ein eigener Vertrag, wird pro Benutzer gespeichert und nie mit den Datenoperationen vermischt.

## Anbieter-Operationen auf Endpunkte abbilden

Die DataSource von Kendo und der CustomStore von DevExtreme unterscheiden sich in der Syntax, doch beide rufen zum Laden eine URL oder eine Funktion auf, und beide rufen erneut auf, um einzufügen, zu aktualisieren und zu löschen, wenn Remote-Operationen aktiviert sind. Die Aufgabe des Wrappers ist das Übersetzen. Bei einer URL-gesteuerten Quelle richtet das InitScript den Anbieter auf die Postback-URL aus, und der `WebRequest`-Handler liest die Argumentnamen des Anbieters, wandelt sie in den `GridOperationRequest` um und gibt das Ergebnis in der Form zurück, die der Anbieter erwartet.

Bei einer funktionsgesteuerten Quelle rufen die Funktionen des InitScripts eine `[WebMethod]` mit typisierten Argumenten auf und geben das erwartete Ergebnis an den Anbieter zurück. In beiden Fällen ist die Serverseite derselbe Handler mit derselben Validierung, und ein Wechsel des Anbieters ändert nur die Übersetzung im InitScript.

Mit dem Aktivieren der Remote-Operationen wird das Grid zu einer echten Integration. Bis dahin sortiert und blättert der Anbieter im Speicher, und der Server wird nur ein einziges Mal gefragt; danach ist jede Benutzerinteraktion ein Serveraufruf, der schnell und sicher sein muss.

## JSON-Optionen oder typisierte Eigenschaften

Kendo und DevExtreme werden beide über große Optionsobjekte konfiguriert, und der schnellste Weg, ein Grid auf den Bildschirm zu bringen, ist, ein solches Objekt direkt aus C# den Options zuzuweisen. Für den schreibgeschützten Pivot-Prototyp im Lab ist das genau richtig, denn dort soll nur gezeigt werden, dass das Widget läuft und der Datenvertrag funktioniert.

Ein wiederverwendbares Grid-Control kann so nicht bleiben. Ein untypisierter Optionssack hat keine Validierung, keine Standardwerte, keinen Hilfetext und keine Möglichkeit festzustellen, welche seiner hundert Schlüssel die Anwendung tatsächlich nutzt. Typisierte Eigenschaften für die Einstellungen, von denen die Anwendung abhängt – etwa Seitengröße, Bearbeitbarkeit und Spaltendefinitionen –, geben dem

Designer etwas anzuzeigen und Reviewern etwas zu prüfen. Die übrigen Optionen des Anbieters können hinter einem geschützten Erweiterungspunkt bleiben, für den seltenen Fall, dass sie gebraucht werden.

Die Abwägung lautet: Tempo heute gegen Wartbarkeit morgen. Bauen Sie den Prototyp mit JSON und machen Sie dann jede Option, die übrig bleibt, zu einer typisierten Eigenschaft, bevor das Control an andere Entwickler ausgeliefert wird.

## Zwei Anbieter, ein Muster

Kendo und DevExtreme kommen in diesem Kurs vor, weil sie weit verbreitet sind und weil ihre Unterschiede das gemeinsame Muster sichtbar machen. Die eine nutzt eine DataSource mit Transport-Einstellungen, die andere einen CustomStore mit Lade- und Änderungsfunktionen. Die eine nennt das Paging Skip und Take, die andere ähnlich, gestaltet den Filterausdruck aber anders. Sobald beide auf denselben `GridOperationRequest` abgebildet sind, schrumpfen die Unterschiede auf wenige Zeilen Übersetzung im jeweiligen InitScript.

Genau diese Lektion sollten Sie auf jede andere Bibliothek übertragen. Die Aufgaben des Adapters ändern sich nie: Ressourcen in der richtigen Reihenfolge laden, das Widget im Container erzeugen, die Optionen übergeben, die wenigen wichtigen Events verdrahten, die Daten über einen vom Server kontrollierten Vertrag liefern, Remote-Operationen verarbeiten und eine .NET-API bereitstellen, die zur Anwendung passt. Ein Team, das dies für zwei Bibliotheken beherrscht, schafft es auch für eine dritte – ohne neuen Kurs.

Halten Sie anbieterspezifisches Wissen im InitScript und in der Event-Übersetzung und den Serververtrag frei von Anbieterdetails, damit eine Änderung bei Lizenz oder Roadmap nie dazu zwingt, Anwendungscode neu zu schreiben.



### Gewohnheit für den Produktivbetrieb

Halten Sie klar getrennt, welchen Zustand Wisej.NET besitzt und wie sich das Client-Widget verhält. Dokumentieren Sie den Client-Server-Vertrag, bevor Sie sich im Produktivbetrieb auf die Integration verlassen.



### Typischer Fehler

Demo-JavaScript in Masken kopieren, ohne Lebenszyklus, Ressourcen, Event-Verträge, Aufräumen oder Sicherheitsgrenzen zu dokumentieren.



### Lernziele

Am Ende dieses Moduls können Sie:

- ✓ Datenverträge für schreibgeschützte, bearbeitbare, seitenweise geladene, gefilterte, sortierte, gruppierte und pivotierte Daten entwerfen.
- ✓ Datenquellen-Operationen des Anbieters auf Wisej.NET-Postback-Handler oder WebMethods abbilden.

- ✓ Verstehen, warum JSON-Optionen schnell sind, typisierte .NET-Eigenschaften für wiederverwendbare Controls aber sicherer.
- ✓ Kendo und DevExtreme als Referenzimplementierungen derselben allgemeinen Integrationsmuster nutzen.

## Lab- und Prüfungsleitfaden

---

Was Sie im praktischen Lab erstellen, der empfohlene Arbeitsablauf, die geforderten Ergebnisse und der Selbstcheck vor der Abgabe.



### Praktisches Lab

Erstellen Sie zwei fortgeschrittene Beispiele: ein schreibgeschütztes Widget im Pivot-Stil und ein bearbeitbares Widget im Grid-Stil. Eines kann die DataSource-Semantik im Kendo-Stil verwenden, das andere die CustomStore-Semantik im DevExtreme-Stil.

## Empfohlener Arbeitsablauf

- ✓ Öffnen Sie das vorhandene Projekt des Wisej.NET-Integrationskurses und starten Sie es lokal.
- ✓ Ermitteln Sie die Server-Komponente, das Client-Widget, die Ressourcendatei oder den Endpunkt, um die es in diesem Modul geht.
- ✓ Nehmen Sie jeweils nur eine kleine Änderung vor und testen Sie sie im Browser, bevor Sie weitermachen.
- ✓ Halten Sie anbieterspezifisches JavaScript im Adapter oder Wrapper, statt es über mehrere Seiten zu verstreuen.
- ✓ Passen Sie die UI so an, dass Erfolg, Fehler, Fortschritt oder Änderungen in der Kommunikation klar erkennbar sind.
- ✓ Halten Sie mit Screenshots oder Notizen fest, dass die Integration funktioniert, und begründen Sie die Designentscheidung.

## Geforderte Ergebnisse

- ✓ Integrationsplan für das schreibgeschützte Pivot
- ✓ Operationsvertrag für das bearbeitbare Grid
- ✓ Server-Handler oder Stubs für Load/Update/Insert/Delete
- ✓ Event-Payload-Verträge für Zellenklick und Zeilenaktualisierung

## Selbstcheck vor der Abgabe

- ✓ Warum sind Grids schwieriger als Gauges?
- ✓ Was ändert sich, wenn Remote-Operationen aktiviert sind?
- ✓ Warum sollten Kendo und DevExtreme als Beispiele gelehrt werden und nicht als der ganze Kurs?