WPF Data Grid v20.1
Virtual Data Source Editing Video Script

Video URL: https://www.youtube.com/watch?v=HOfGI6nfWsA&t
Product URL: https://www.devexpress.com/Products/NET/Controls/WPF/Grid/
Free Trial URL: https://www.devexpress.com/Products/Try/

We expect to ship our next major update – v20.1 – in May 2020. Among new WPF-specific features we'll include in this update is data editing support for our WPF Grid's virtual data source.

As I'll demonstrate shortly, data editing support for our grid's Virtual Data Source will be available at the row level. If you're using an infinite or a paged virtual data source, your end-users will be able to modify row data and post changes to the underlying data source asynchronously.

Before I start, a quick word for those who are new to our WPF Data Grid. Our WPF Grid allows you to bind data to any data source – even when the total record count is unknown. Unlike a more traditional data source – when you bind our grid to a virtual data source, the control only requests top records or a specific page of records. In previous versions, data editing was unavailable due to the "virtual" nature of the data source. Well, as you can obviously tell, this limitation no longer exists. We now fully support data editing for virtual data sources.

The DevExpress WPF Data Grid ships with an unbelievably rich data editing library – from date selectors to advanced drop down controls. We'll discuss our industry-leading cell editing options in a future video but for now, let's get into it and show you how you can edit virtual data source records.

To help demonstrate this feature, I'll use a sample that's currently available on our GitHub page. You'll find a link to this sample in this video's description. In this GitHub sample, our WPF Data Grid already uses a virtual data source. As you can see, it connects to a class that imitates a web service and implements basic data operations such as sorting and filtering.

To enable data editing for this sample, I'll need to modify the IssueData class and make its property setters public. Next, I'll have to add a new method to our service class to save changes made in our Data Grid. This method will be asynchronous – this will help prevent UI lock-ups during save operations.

Once I've added the service class method, I'll specify the fields that can be edited at runtime. I'm going to find my priority column in XAML and set its AllowEditing property to True. I will also configure our WPF Data Grid's Table View to display Update and Cancel buttons during row edit operations.

Before I continue – I do want to quickly mention that our WPF Data Grid ships with a View-based architecture. If you've used our Grid in the past, you'll know that the control ships with 3 view options – Tables, Cards and TreeLists. Virtual Data source editing is currently available for our Grid's Table View.

Ok – with those simple changes to the GitHub sample, I'm ready to run the project and view my results. As you can see, the priority column is now editable. Notice that if I make a change to a record and then press the Update button, nothing actually happens. The reason for this is simple – our changes are in memory but have not been passed to the web service.

Let me close the running app and return to my project. This time, I'm going to add a handler for ValidateRow in my grid's Table View. In the event handler, I can call the update row async method to save changes. As you can see, the update row result property accepts a task that allows you to work with asynchronous methods.

If we run the app, and press update, the data grid will call our web service method and display a wait indicator symbol while it awaits service results.

Before I wrap up this tutorial, I want to show you how to easily add validation rules to this GitHub sample. I can do this directly in the ValidateRow method – but I'm going to add my validation rule in the sample's service class instead. I'll do this to demonstrate the ease with which our WPF Data Grid handles server side errors. It's really quite amazing.

I'll add a simple check to update row async method and throw an exception for the subject field if the subject field is empty. As you may recall, we only enabled data editing for the Priority column – so the final step is to return to the Data Grid and enable data editing for the subject field.

Let me run the app once more and edit the subject field. I'm going to delete the contents of the row and press update. Note that my exception message appears on-screen. This validation rule requires that I add a value to the subject field. I'll go ahead and do that so I can leave the modified row. Let me enter a character into the subject field and press Update again.

As you can see, our WPF Data Grid automatically handles exceptions from the ValidateRow event handler and does not cancel editing unless it receives a positive response from the web service. Very cool.

I hope you've enjoyed this short tutorial on the DevExpress WPF Grid and its editing options for Virtual Data Sources. If you liked this video, please give us a thumbs up. If you have questions, please comment below. And please remember to subscribe to this channel for more great DevExpress training videos.