DevExtreme JavaScript UI Components (Angular, jQuery, React, Vue)
Data Grid Excel Data Export API

Video URL: https://youtu.be/5Ul1XvmMX10
Product URL: https://js.devexpress.com/Overview/DataGrid/
Free Trial: https://js.devexpress.com/

As you may already know – our JavaScript product line includes support for a variety of development frameworks including Angular, jQuery, Vue, and React.

In this video, I'll discuss DevExtreme's new and 100% client-side Excel Data Export API. As you would expect from a fully client-side implementation, our Excel Data Export API works in the browser and does not affect web-server resources or expenses.

User feedback helped shape our implementation. Our JavaScript Grid's Excel Data Export API was engineered to address customer demands and a variety of real-world usage scenarios. Ultimately, we chose to build our API around Excel JS – an open-source library that gave us comprehensive customization control over the Excel data export process.

In the first portion of this video, I'll share a basic overview of this feature and what you can expect when you use our new Excel Export API with our JavaScript Data Grid. In the second portion of this video, I'll dig a bit deeper into our implementation and consider advanced use cases. To help demonstrate this feature, I'll be using demos published on the DevExtreme Widgets Gallery webpage. To follow along, simply point your browser to js.devexpress.com/demos, select Data Grid and use the navigation tree to jump to our Export to Excel demo collection.

OK, with that – let's get started.

Oh and before I forget, I'm using our Angular demo collection but this feature also applies to React, Vue, and jQuery.

The first demo I'll tackle is our Overview demo – a list of employees grouped by the State column. As you can see, our JavaScript Grid's toolbar includes an Export command button. Once you click this Export button, our Grid control displays two export options. I can either export all data within the grid or only selected rows. This menu is configurable – so you can control what users see when the button is clicked.

Let's proceed with this demo and select the Export all option. Since this tutorial is all about Excel, let me open Excel and view the generated XLSx file now. As you can see, my exported document mirrors the display of my grid – data is grouped within the spreadsheet by state and sort order is retained. Very cool.

A quick note: You as the developer control the entire export process. In this demo, I did not display a file dialog but you can certainly do so and require users to enter a file name before generating the Excel file.

I'll go ahead an enable editing and show you a few more things within Excel. Notice that grouped rows within the spreadsheet are using a bold font effect. Also note that the generated Excel file allows you to leverage Excel's group row expand/collapse feature. Finally, the exported file employs Excel's Column Filter option. If you configure our JavaScript's Grid with this option, your user will be able to filter data within the spreadsheet document as needed. Again, very cool.

Let's now return to our DevExtreme Data Grid demo collection and run a different sample so I can explain a few more options available to you when using the DevExtreme Excel Export API. I'm going to switch to our Cell Customization demo and initiate export. As you can see, this particular exported file contains cell customizations. We changed row background colors, we exported hyperlinks and yes, they are active. We also applied unique font styles to specific cells. The bottom line – you have complete control over output and the manner in which cells are formatted within Excel.

OK – moving on – let me run the Header and Footer sample to show you a different but very popular usage scenario. This particular demo includes area, population, and GDP information organized by country. The demo also includes both a header and a footer in the form of HTML.

I'll go ahead and export now and load up Excel. And as you can see, both the Header and Footer are included during the export operation. A couple of other things worth mentioning here. In our demo, our DataGrid used column bands to organize information on-screen. The generated Excel file retains band configurations. Second, note that our new Export to Excel API retains original cell data formats. For instance, GDP column cells are exported as percentages.

We're on a roll – let's keep going. I'll now shift focus to multiple worksheets inside a single workbook. Let me load up our Export Multiple Grids sample. In this example, we have two data grids displayed. Users can navigate between the Price and Rating table via a Tab control. Now, once I press the export button, I'm going to except two worksheets within my workbook.

And there it is – in this sample, we generate an Excel file with two separate worksheets. One includes Price data and the second includes Rating info. Easy as can be.

One final sample to cover – I promise, we are near the end. Let me move to our Export Images demo and show you how easy it is to export images with Base64 encoding to an Excel file. I'll press export and open Excel to show you what this looks like. Here's our exported data inside Excel. Again, easy.

Ok, we've covered a few usage scenarios – let's now look at actual code and configuration

options. I'll go back to the Overview demo and scroll down to view its code snippet. Since we started with Angular, I'll stay with Angular but note once more that our JavaScript DataGrid and our Excel Export API supports multiple development platforms.

First things first – to enable the UI elements necessary for Excel export, we need to add a child component called dxo-export and set enabled to true. If you recall, this demo gave us the option to only export selected records, so we also need to set allowExportSelectedRecords to true.

To pass Grid data to our new Excel API, we need to handle the OnExporting event. Let's look at this event handler in the TypeScript code behind for this demo.

Note that we import the ExcelJS library from ExcelJS npm package. We also use a file saver library so users can download exported documents within the browser.

Let's take a look at the actual handler. As you can see, it's pretty straightforward. We create an Excel workbook and then add a single worksheet to it – named Employees. Using our Grid's API, we pass our Data Grid to the component option – the worksheet to the worksheet option and we specify one option for the target Excel document – an option called autoFilterEnabled. Recall, the first Excel file I generated in this tutorial allowed users to filter column data. This option enables that capability.

Once the Excel file is generated, we need to enable download. To do this, we call SaveAs - pass the content as a buffer – specify content type – and name the actual file. And that's all there is to it.

I hope you've enjoyed this introductory tutorial on the DevExtreme's Excel Data Export API. To learn more, please make sure to visit js.devexpress.com/demos at your convenience.

As always, if you liked this video, please give us a thumbs up. If you have questions, please comment below. Please remember to subscribe to this channel for more great DevExpress training videos.

Copyright © 2020 Developer Express Inc.
All trademarks or registered trademarks are property of their respective owners.