My First ExtJS DataGrid Pt 7: Custom Cell Renderers

Posted At : August 22, 2007 11:07 AM | Posted By : Cutter
Related Categories: ExtJS, JQuery, Development, My First ExtJS DataGrid

So, it's been awhile. No, I haven't forgotten you, I've just been busy with a lot of things. One of which has been implementing a new **ExtJS** DataGrid in a project I'm working on. Sure, there's a lot more going on, but that's becoming a nice front end piece. As previously promised, I want to look at a *renderer*.

What Is A Renderer?

Hey, not everything you get back from your paging query will be formatted in the way you want it displayed. I'm going to show you a really simple example, dealing with a boolean value. Let's start off by writing a renderer function. Here's the deal, you have a query column with a boolean value (bIsActive), which is returned as either 1 or 0. You want the grid to display*Yes* or *No*. You could define the function directly in your ColumnModel declaration:

```
...
   ,{
   header:'Active',
   dataIndex:'bIsActive',
   renderer:function(value){
    return (value == 1)?'Yes':'No';}
   }
   ...
```

Easy, right? But, it's not really reusable is it? So, it's better to define the function, then register that function as the renderer for the column. Now, there isn't a ton of documentation on exactly what's going on here, but there are a lot of examples, so I'll try to give you what I've figured out, plus what I believe to be right (if you know it better then let us all know).

A renderer function will take at least one argument by default, that being the *value* of the cell being rendered. You do not explicitly call a renderer, the arguments passed to the renderer are dependent upon the function definition. For something as simple as a Yes/No boolean renderer the *value* is all that's necessary. So you would define a renderer function, and register it, like this:

```
function renderBoolean(value){
    return String.format("{0}",(value==1)?'Yes':'No');
   }

   // And a redefinition of of the renderer in the ColumnModel
   ...
   ,{
   header:'Active',
   dataIndex:'bIsActive',
   renderer:renderBoolean
   }
```

```
    ...
```

Did you see it? No parens on the end of that renderer config property in the ColumnModel, and no arguments explicitly passed. That's all covered by the ExtJS framework. But wait, it gets better. We had to columns in the data store that relate to a person's name (vcFirstName and vcLastName). What if we just wanted to concantenate the two values? Well for that we would probably need the entire record. No problem, you just change things up a bit:

```
function renderName(value,p,r){
    return String.format("{0} {1}", value, r.data['vcLastName']);
   }
                        // Combine our 'name' columns into one
   ...
   ,{
   header:'Name',
   dataIndex:'vcFirstName',
   renderer:renderName
   }
   ...
```

OK, three arguments, with one never used. Well, it wasn't used in this case, but it's necessary, as you can't get the third argument (the data record) passed into the renderer without also passing in the second argument. What's the second argument? Well, that's a great question. I'm fairly sure that it is the element object of the container of the value, but I could be very wrong. In this particular case it doesn't matter, but I'll give you another item renderer in a future post where it will play a part.

You'll also want to note the the JavaScript bind variable syntax being used in the String.format() function, {[value]}. The numbers used here are like array position values, starting at zero, for all of the variables passed in the arguments that follow the string you are formatting (the first argument of the function).

So, that's a quick down and dirty on renderers. No download with this particular post, but the code samples above aren't overly difficult to integrate. You can, absolutely, do some much more complex renderer functions, but this should be a good start.