CFQueryReader Update: Support for ExtJS 5 and ColdFusion 11

Posted At : December 4, 2014 10:53 AM | Posted By : Cutter
Related Categories: CFQueryReader, Ajax, Development, ColdFusion, ExtJS

Due to a donation from Bruce Lomasky (thanks Bruce), I've written a new CFQueryReader for ExtJS 5. CFQueryReader is an Open Source project, **available on GitHub**, that I began years ago to create a proper reader for parsing Adobe's native ColdFusion query JSON return format into something usable by **ExtJS** data stores. As ExtJS has changed over the years, I have updated the reader to accommodate new versions. But, I don't do much ExtJS development anymore. It's a great front-end app development framework, but I've changed jobs a few times since co-authoring the first books on the library, and just don't work with the technology on a day to day basis anymore. Bruce had a need, and offered to pay for the development of an updated reader. So I dove back in to the ExtJS pool, to help him fulfill his requirements and put a new reader out there.

ExtJS went through a major rewrite between versions 3 and 4, requiring an entirely new direction. While much is backward compatible, in the move from version 4 to version 5, there were some big changes to ExtJS' data packages. Sencha has always done a good job with creating a rich, dynamic framework for web application development. I have always been impressed with their commitment to improving the product, continuously modernizing the framework, and creating more of a "software craftsmanship" approach to web application development. That said they haven't always done such a great job with ensuring an easy upgrade path for developers. Changes to their data package completely broke some of the use cases of CFQueryReader, requiring some refactoring to accomadate.

And that's OK. Sencha's changes to their data packages are some welcome changes, especially for our little plugin. In all of the past revisions of CFQueryReader, we've extended a base Array Reader class, and written an override method of the core readRecords() method. While this worked, it was really kinda kludgy. What we really needed was a way to transform the incoming data packet prior to the reader really processing the object. With ExtJS, we now have that ability to do just that.

ExtJS 5 introduced a new configuration option for data readers: transform. This new configuration attribute takes a function within which you can manipulate the return data of a store's ajax request prior to the reader actually processing that data. This gives some underlying flexibility that wasn't really there before, especially when using Ext.Direct, but for now you really just need to know the basics.

ColdFusion upper cases object keys when it serializes to JSON. If you are manually creating the structure, there are ways to fix that:

```
var myStruct = {
        "key1": "some value,
        "Key2": "this one is cased differently",
        "differentKey3": "this one is more different"
    };
```

This is fine, and helps a little bit, but ColdFusion creates an object when it serializes a ColdFusion query object, and you can't control it's casing:

```
ColdFusionJson = {
        COLUMNS: ["KEY1","KEY2","DIFFERENTKEY3"],
        DATA: [
            ["some value","this one is cased differently","this one is more different"]
        ]
    };
```

When you build your reader configurations, we no longer worry about this casing, as we will do a case insensitive matching of keys during the pre-process:

```
reader: {
        type: "cfquery",
        rootProperty: "myQuery",
        totalProperty: "totalCount"
    }
```

NOTE: We will lowercase column names during preprocess, and you should remember that when creating your "dataIndex" attributes of your Model's Field configurations.

CFQueryReader will read ColdFusion query JSON serializations that are:

1. The root of your return

2. Nested within a larger return object (struct)
3. Have been converted with ColdFusion QueryForGrid (not suggested, but supported)

And, because ColdFusion 11 includes the new "struct" type for the queryFormat parameter (even as an argument of your ajax request), CFQueryReader will properly parse that as well.

And, since we can now process the return prior to ExtJS calling it's internal readRecords() method, CFQueryReader no longer extends Ext.data.reader.ArrayReader as it's base class, but the more appropriate Ext.data.reader.JsonReader instead, allowing for a more native access approach under the hood.

I've made the new plugin available, for now, within the **Ext5** branch of the repository. If you have the need, take it for a spin, and let me know if there's anything that might require tweaking.