JQuery Datatables plugin example using a server side data request (coldfusion)

Posted At : 29 April 2010 18:04 | Posted By : Shaun McCran
Related Categories: JQuery, Coldfusion, Javascript, RIA, Json, AJAX

Im my previous article on how to use the JQuery datatables plug I concentrated mainly on the JQuery script, and how to build the AJAX request to receive a JSON response.

In this article I will demonstrate the full application which will include the front end JQuery, building the back end server response and a few tips that I've picked up since implementing the plugin. I am using an MS SQL table filled with UK location data that I also used for a weather webservice, to fill the table.

A full example of this working can be seen here: Data table server side example

The front end - JQuery

This is built using the JQuery datatables plugin. So firstly get the JQuery library from Google, and the Jquery plugin script. For this example we are also using the demo css provided by www.datatables.net.

```
s/cript src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.0/jquery.js"></script
        <sc/ript language="javascript" src="dataTables.js"></script>

                <style type="text/css" title="currentStyle">
                        @import "demo_page.css";
                        @import "demo_table.css";
                            </style>
```

Next to actually build our data table object. We simply list any of the parameters that we want to pass to the dataTable method as name value pairs. I have been using this for a while now, and have settled on the

options below. (I'm only explaining certain values, if you are unsure of them all, use the documentation on www.datatables.net)

The **'bStateSave'** value is very handy as it tells the plugin to use Javascript cookie to remember any user filtering or sorting criteria. In this way page reloads do not reset the data display.

The **'bServerSide'** value tells the dataTable that the data is coming froma server request.

The **'sAjaxSource'** value tells the dataTable what template to query for a Json response.

The **'aoColumns'** value builds an Array which sets up the actual rows in the dataTable. This is where you can set the width, and the headers for the display.

The last few options are dealing with the paging setup. They are text book ripped from www.datatables.net.

```
<s/cript type="text/javascript" charset="utf-8">
        $(document).ready(function() {
            $('#displayData').dataTable( {
                "bProcessing": true,
                "bStateSave": true,
                "bServerSide": true,
                "sAjaxSource": "handler.cfm",
                "aoColumns": [
   {"sName": "id", "sTitle": "ID", "sWidth": "20%", "bSortable": "true"},
 {"sName": "varCode", "sTitle": "Code", "sWidth": "40%", "bSortable": "true"},
"sName": "VarLocation", "sTitle": "Location", "sWidth": "40%", "bSortable": "true"
                ],
                "sPaginationType": "full_numbers",
                "aaSorting": [[1,'asc']],
                "oLanguage": {
                "sLengthMenu": "Page length: _MENU_",
                "sSearch": "Filter:",
                "sZeroRecords": "No matching records found"
                                                        },
```

Next we need to actually send the request to the server. The **'fnServerData'** function collates all the values, and allows you to add any

other data you want here. Stick to the "name: value method" of passing data and you can't go wrong. In this example I am passing in a table value of 'ukLocationCode' and a SQL string. These values can be referenced as POST values in the data handling script.

Lastly I am using the &.ajax function to POST the data. I have left a commented out $.getJSON method to show the GET method. I am using POST as IE tends to cache the data results using GET requests.

```
        "fnServerData": function ( sSource, aoData, fnCallback ) {
                                    aoData.push(
                          { "name": "table", "value": "ukLocationCodes" },
                 { "name": "sql", "value": "SELECT [id], [varCode], [varLocatio
                                                );

                                  $.ajax( {"dataType": 'json',
                                             "type": "POST",
                                             "url": sSource,
                                             "data": aoData,
                                        "success": fnCallback} );


    // $.getJSON( sSource, aoData, function (json) {fnCallback(json)} );
                                                                 }
                                                    } );
                                                } );
                                    </script>
```

The final part is the html display. The only thing to watch for here is that you give the table element the same id value as used in the script above.

```
                            <h2>Data Tables Example</h2>

or a data Tables example. It is handling the data(Json) from an AJAX post, and dis
                 All changes are made inline, so there are no refreshes.</p>
                                        <br/>

:table cellpadding="0" cellspacing="0" border="0" class="display" id="displayData">
                                           <thead>
                                                <tr>
                                          <th align="left">ID</th>
                                         <th align="left">Code</th>
```

```
                                <th align="left">Location</th>
                                        </tr>
                                    </thead>
                                     <tbody>
                                            <tr>
                <td colspan="3" class="dataTables_empty">Loading data from serve
                                        </tr>
                                    </tbody>
                                </table>
```

The back end – Coldfusion response

My server response has been built using Coldfusion, but almost all of the principles here are not language specific. IE if you are doing this in PHP then watch out for the same sticking points.

This script params all the POST values that it is expecting to ensure they exist. It is then performing two database queries. The first to get the total record count for the results. The second to actually get the data to go into the table. The second query uses a few of the values passed to it to determine if there are any filtering or sorting criteria being applied.

Lastly I use a create the Json response from the Query results. I am simply looping over the query records and outputting them in a Json format. Here it is also possible to intercept any specific values and apply custom formatting. In that way they are returned in exactly the right format for your dataTable display.

```
                            <cfsilent>
                        <cfparam name="form.table" default="">
                       <cfparam name="form.sColumns" default="">
                    <cfparam name="form.editButtonText" default="">
                   <cfparam name="form.editButtonTarget" default="">
                        <cfparam name="form.sSearch" default="">
                    <cfparam name="variables.fieldlist" default="">

                            <cfsetting showDebugOutput=false>
                          <cfsetting enablecfoutputonly="true">
                   <cfprocessingdirective suppresswhitespace="true">

               <!--- this comes from the AJAX script in the template --->
                        <cfset variables.fieldlist=form.sColumns>
                                <cfset variables.count=0>
```

```
        <!--- strip off the comma if it is the last element --->
              <cfif right(variables.fieldlist,'1') EQ ","> 
                        <!--- last char is a comma --->
             <cfset variables.listLength = len(variables.fieldlist)>
  <cfset variables.fieldlist = left(variables.fieldlist, variables.listLength-1
                              </cfif>

                      <!--- get count of records --->
            <cfquery name="qGetCount" datasource="A8002CMS">
                          SELECT COUNT(*) AS fullCount
                               FROM #form.table#
                              </cfquery>

             <cfquery name="rResult" datasource="A8002CMS">
                        #preservesinglequotes(form.sql)#
                               FROM #form.table#

                                  WHERE 1 = 1
                            <cfif len(form.sSearch)>
                                        AND (
<cfloop from="1" to="#listLen(variables.fieldlist)#" index="variables.index">
variables.index,',')# LIKE '%#form.sSearch#%' <cfif variables.index LT listLen(va
                              </cfloop>
                                        )
                               </cfif>

                     <cfif isdefined('form.iSortCol_0')>
                                     ORDER BY
      <cfloop from="0" to="#form.iSortingCols-1#" index="variables.i">
form["iSortCol_#variables.i#"]+1)# #form["sSortDir_#variables.i#"]# <cfif variables
                              </cfloop>

                               </cfif>
                              </cfquery>

 strip off the table name from the values, otherwise it will break making the json
set variables.fieldlist = ReplaceNoCase(variables.fieldlist,'#form.table#.','','al

                     <!--- create the JSON response --->
           <cfsavecontent variable="variables.sOutput"><cfoutput>{
                                "sEcho": #form.sEcho#,
                          "iTotalRecords": #qGetCount.fullCount#,
                      "iTotalDisplayRecords": #rResult.recordcount#,
                                 "aaData": [
ow="#form.iDisplayStart+1#" endrow="#form.iDisplayStart+form.iDisplayLength#"><cfs
           [<cfloop list="#variables.fieldlist#" index="variables.i">
                        <!--- custom translations --->
                  "#rResult[variables.i][rResult.currentRow]#"
       <cfif variables.i is not listLast(variables.fieldlist)>, </cfif>
                              </cfloop>]

     <cfif rResult.recordcount LT form.iDisplayStart+form.iDisplayLength>
               <cfif variables.count is not rResult.recordcount>,</cfif>
                                 <cfelse>
                <cfif variables.count LT form.iDisplayLength>,</cfif>
                               </cfif>

                              </cfloop>
                                        ]
```

```
}</cfoutput></cfsavecontent>
  </cfprocessingdirective>
        </cfsilent>
<cfoutput>#variables.sOutput#</cfoutput>
```

Points of note

Make sure that there is no whitespace in the beginning of your Json response. If there is then some browsers will not interpret it (I'm looking at you IE 6/7)

Watch out for trailing commas after your data elements In your Json. Firefox will compensate for them, but IE thinks there is a missing element so will not display any data at all.

Use www.jsonlint.com to validate your Json

Use firebug for firefox, or http://www.charlesproxy.com/ to track the inline AJAX requests and responses. Both these tools are invaluable

Your AJAX requests will still be subject to an Application (.cfm/.cfc) code that they inherit. In one example of this code I had four random lines of whitespace appearing that were actually in an Application. cfm file further down my folder structure.

A full example of this working can be seen here: Data table server side example