Activity Log / Work Plan

In order to solve this issue, first I need to build some test scenario to reproduce it and see what is actually going on.

The initial set up will require to:

Create a test Project to reproduce the issue

Create a database to hold test data similar to the one described in the email.

Test performance and behavior without setting up OutputCache.

Test performance and behavior setting up OutputCache.

For testing memory usage I used Visual Studio’s Diagnostic Toolbox.

Ok, It seems that there is some issue with the varyByParam attribute when it is configured in the profile key of the web.config. I try setting up the OutputCache directly in the action method without using CacheProfile… like this:

[OutputCache(Duration = 3600, VaryByParam = "iDisplayStart;iDisplayLength;sSearch")]  
public JsonResult List(DataTableQuery query)

And the cache seems to work perfectly.

I did a little bit of research on the issue and I found someone on stackoverflow.com with a similar problem:

<http://stackoverflow.com/questions/23764926/asp-net-outputcache-varybyparam-does-not-work-in-web-config>

But what really caught my attention were the comments on the accepted answer. Users [haim770](http://stackoverflow.com/users/1625737/haim770) and [rnd](http://stackoverflow.com/users/2573335/rnd) found out that value VaryByParam seems to be always "\*" in this kind of scenarios.

Now, since VaryByParam is practically set to "\*", OutputCache will keep the output for almost every call that Datatable does to the action method, and incidentally this will increase the memory usage. This is because Datatable includes two parameters in the query string that change on every call: "\_" and "sEcho".

We can solve this OutputCache's issue with two aproaches: One is not using Profiles and specifying the Duration and VaryByParam directly on the attribute, like I did above. If we only need to use this attribute here, I think this solution will suffice. But if we really have a lot of places where we need to use output cache then using profiles is maybe the best choice. We can implement and use a custom attribute that inherits from OutputCacheAttribute and solves this issue with the profile. (I did it on the attached sample project)

Ok, now by doing this, we solve the caching problem, but another issue appeared… Datatable sends with each request the parameter "sEcho" which is intended to identify the request and stablish an order among multiple asyncronous calls to the server. The above is to draw the result of the last request issued (In order to draw the result, Datatable expects to get back from the server the same sEcho that sent on the request).

This new issue makes the "previous" button useless, since when we hit that button, a cached result is returned, but that result, doesn't have the same "sEcho" that was issued by the request, an so, Datatable doesn't render the result.

To solve this, we could include "sEcho" in the list of VaryByParam, but that would jeopardize the memory usage of the output cache since It will increase the amount of duplicated result stored in the cache.

I suggest fixing this issue with a workaround. It may not be completely necessary to send the "sEcho" to the server. Since Datatable lets us use set up the parameter fnServerData to customize the way we request data from the server, We can build here a javascript closure that holds the value of sEcho and reappend it to the resulting data when the request is completed.