**Efficient Paging with WebGrid Web Helper - ASP.NET MVC 3 RC**

**Posted by:** [Malcolm Sheridan](http://www.dotnetcurry.com/Author.aspx?AuthorName=Malcolm%20Sheridan) , on 12/9/2010, in **Category** [ASP.NET MVC](http://www.dotnetcurry.com/BrowseArticles.aspx?CatID=67)

**Views:** 89916

[23](http://topsy.com/trackback?url=http%3A%2F%2Fwww.dotnetcurry.com%2FShowArticle.aspx%3FID%3D618&utm_source=button)[retweet](http://button.topsy.com/retweet?nick=dotnetcurry&url=http%3A%2F%2Fwww.dotnetcurry.com%2FShowArticle.aspx%3FID%3D618&title=Efficient%20Paging%20with%20WebGrid%20Web%20Helper%20-%20ASP.NET%20MVC%203%20RC)

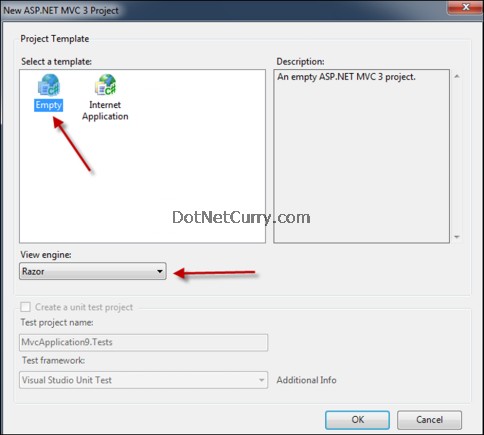
**Abstract:** The following article demonstrates one way of performing efficient paging using the WebGrid WebHelper in ASP.NET MVC 3 RC.

Last week I published an article on how to use the [**WebGrid WebHelper in ASP.NET MVC 3 RC**](http://www.dotnetcurry.com/ShowArticle.aspx?ID=615). This was a beginners article on getting started. The one piece of functionality I didn’t touch on was paging. Out of the box, the WebGrid has a built-in Pager object, which handles the rendering of paging links underneath the WebGrid. The biggest downside of using this paging is smart paging isn’t enabled. What I mean by that is if you have a WebGrid fetches it data from 10,000 records from a database, but it only displays 10 records per page, each time you click on a paging link, you’re not retrieving 10 records you want to display, you’re fetching the 10,000 records each time and only displaying 10. This is terribly inefficient when you hit a large scale system. Well this article you’re reading right now will show you one possible way of creating efficient paging and still using the WebGrid.

Before moving on, you need to download ASP.NET MVC 3 RC. [Click here](http://www.microsoft.com/web/gallery/install.aspx?appid=MVC3) to download and install them using the Microsoft Web Platform Installer.

*The Microsoft Web Helpers are added as a reference by default now in ASP.NET MVC 3 RC. I’ve left the instructions on how to add them in this article in case you need to do this as a separate exercise.*

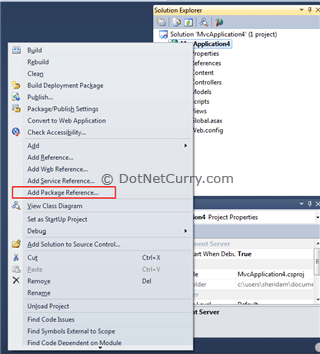
Open studio 2010 and create a new ASP.NET MVC 3 Web Application (Razor) project. The new MVC 3 dialog can be seen below:



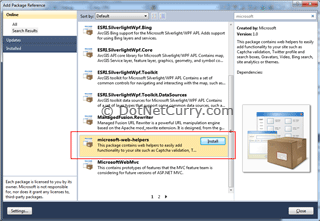
Choose Razor as the view engine and click OK.

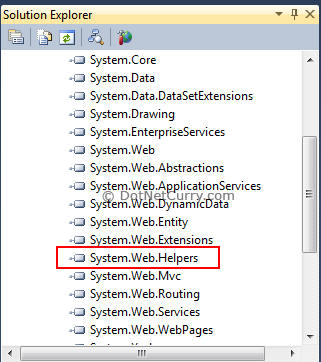
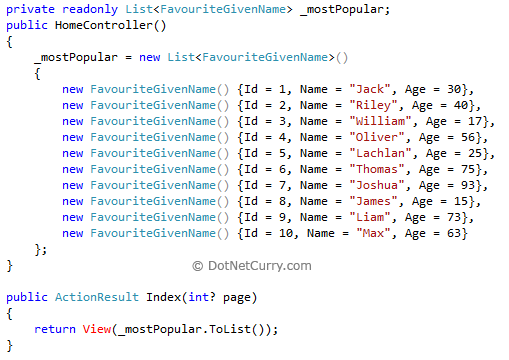
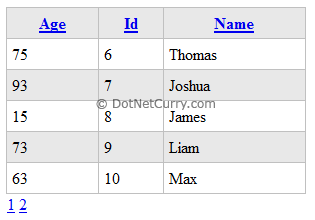
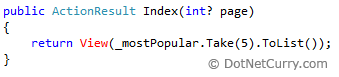
The next step is to download the Microsoft Web Helpers library through NuGet. Follow these steps to do this:

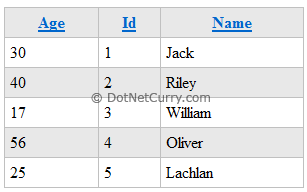
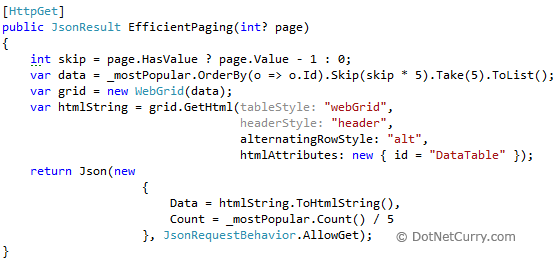
Step 1. Right click the website and choose Add Package Reference.



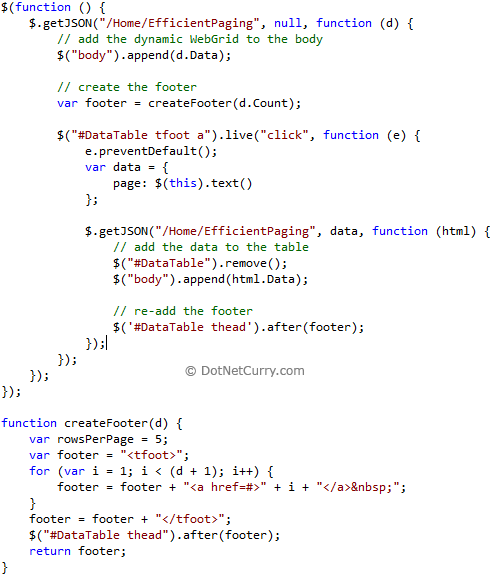
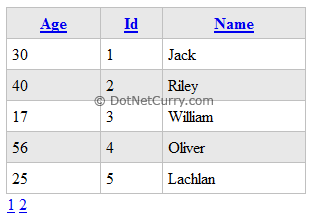
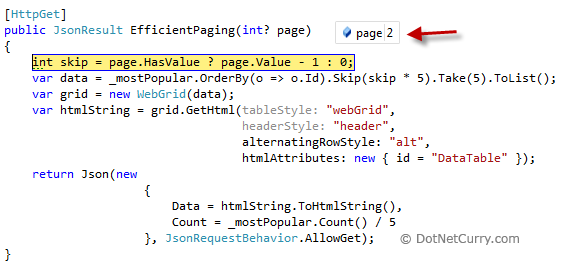
Step 2. The Add Package Reference dialog will open. Search for Microsoft in the search box in the top right corner. Upon completion you'll see **microsoft-web-helpers** option. Select that option and click Install.



Step 3. Click 'Close' to return to the project   
  
Check the references and you'll see that the library has been added to the project.   
  
  
  
Now it's time to start coding! For this example I’ve created a model called FavouriteGivenName. This will store 10 records. The model would normally be a database, but I’m using a smaller set of data to illustrate the paging examples.   
  
**Default Paging in MVC 3 WebGrid**  
  
By default, the WebGrid’s paging functionality will retrieve the entire set of data for each paging request. To see this in action, create a Home controller and add an index action.   
  
  
  
The code above is setting model for the view as a collection of FavouriteGivenName’s. Add the following code to the view:   
  
  
  
The output from the above code is shown below.   
  
  
  
The grid.Pager method sets the paging for the WebGrid. The problem is when you page through the data, all the data is returned. If you try to limit the data being returned, the problem you’ll encounter because you’re only returning a subset of the data is the WebGrid thinks there’s only that amount of data to display, so the paging links will disappear! Not good! So to show you what this looks like, I’m going to only return the first 5 records for the WebGrid, because I only want to display 5 records per page. Here’s my updated action:   
  


Now because I’m returning only 5 instead of 10 records, the paging has disappeared from the WebGrid!   
  
  
  
**Efficient Paging in MVC 3 WebGrid Web Helper**   
  
The solution I came up with is one possible solution. I’m sure there are many others, but this works well. The issue around the WebGrid’s paging is it needs the total count of data that the grid is supposed to display. If you limit the WebGrid to only the records you want displayed, you limit the total count and the paging doesn’t work. To fix this problem you need to separate the two. One set of data for the grid and a separate piece of data that tells the WebGrid how many paging links to display. Unfortunately the paging object is locked, so I had to make my own.   
  
To incorporate the [DRY (Don’t Repeat Yourself)](http://en.wikipedia.org/wiki/Don%27t_repeat_yourself) principal, I’m going to create an action method that returns the WebGrid as one object, and a total record count as another object and wrap both of them in a [JSON](http://en.wikipedia.org/wiki/JSON) object. Here’s the action method.   
  
  
  
The JSON being returned stores two object.

* Data – this stores the data only needed for the grid. It utilises the [Skip](http://msdn.microsoft.com/en-us/library/bb386988.aspx) and [Take](http://msdn.microsoft.com/en-us/library/bb386988.aspx) methods to limit the data being returned. It is returning the WebGrid as HTML. This ensures this code will replicate what the WebGrid would look like if we were declaring it in the Razor mark-up.
* Count – this stores the total records. This will be used for creating the paging links.

The view for this is simple. All the hard work is in the JavaScript. Here’s the JavaScript below that calls the EfficientPaging action and displays the returned WebGrid and creates the paging links.   
  
  
  
The result of this is below.   
  
  
  
It looks the same as the previous example, but now if you page through the data, the page you’re requesting will be sent as a parameter to the action method, and only that data will be fetched from the server.   
  
  
  
Now you have efficient paging and are still using the WebGrid WebHelper! The only difference is you’re creating your own paging links.   
  
Like I mentioned earlier, this is only one possible solution, but for me this works well because I’m still creating only one WebGrid. If you think of other ways to do this, please let me know.   
  
The entire **source code** of this article can be downloaded over [**here**](http://www.dotnetcurry.com/Uploads/mvc/MVCWebGridHelperPaging.zip)

Give me a +1 if you think it was a good article. Thanks!

[23](http://topsy.com/trackback?url=http%3A%2F%2Fwww.dotnetcurry.com%2FShowArticle.aspx%3FID%3D618&utm_source=button)