**Strategies For Displaying Lots of Data**

I sometimes get a request from users like:

*Can you please show a list of the last 50 items that went through the system? I just want to go and see the stream of items most recent in order as they were processed/received.*

The task then is to display that data. You can't *guess* and cap how many records the user wants to see - your guess will inevitably be wrong. Splashing *every* record on the page isn't right, either.

1. **Search** - I try to make this the default and try to convince users that they should search for the items that they want. This strategy helps everyone: it retrieves the most relevant data that the user wants, and is light on your web app. Sometimes users push back on this as it forces them to think about what they want, rather than visually searching all items on the page. I'm even done with the typical [**awful one-to-one-textbox-to-column**](http://i.imgur.com/tJmf9CP.jpg) layout. Just give the user a single textbox (search-engine style) where they can type in some text, and perhaps give them some date range inputs if dates are involved. All this is solved when you implement full-text catalog.
2. **Pagination** - The old method of providing [**page numbers with next and previous buttons**](http://i.imgur.com/YcZ3Tof.png). [**It's possible**](http://stackoverflow.com/questions/446196/how-do-i-do-pagination-in-asp-net-mvc), but just not optimal. For me, it just doesn't feel right.
3. **Infinite Scroll** - the user will pull the data as they need it. This satisfies that user requirement above.

**Infinite Scroll NuGet Package**

I've created a NuGet package for MVC applications that will add infinite scrolling to your tables, (un)ordered lists, divs, or any other repeating element in your MVC views. The user simply scrolls down to the end of the page when they want to see more data; JavaScript takes over from there.

It created a better user experience: the user to retrieves a smaller page initially, decreases the user's clicks, and to provides quick responses when asking to see more data.

**[](https://www.nuget.org/packages/MVC-Infinite-Scroll)**

Install-Package MVC-Infinite-Scroll

The package will install the necessary bits for infinite scrolling, and a full demo controller+view example.

* 1 JavaScript file - infiniteScroll.js
* 1 demo controller
* 2 demo views and 1 image
* sample data in App\_Data\Customers.csv

When you're finished with the demo, you can simply delete the controller and views named InfiniteScrollDemo.

**Open Source On GitHub**

I've put a demo project, the NuGet package, and its source on the [**MVC Infinite Scroll GitHub repo**](https://github.com/philoushka/Mvc-Infinite-Scroll-Grid).

**Infinite Scroll Demo**

Check out the demo at: [**http://mvcinfinitescroll.azurewebsites.net/**](http://mvcinfinitescroll.azurewebsites.net/)

**Deep Dive**

Here's how it works:

1. The user navigates to a controller action method. It determines that the user wants to build a list of data.
2. The controller calls out to get the data. Sort/order it as you like.
3. The sample data is a Dictionary<int,Customer> because I **wanted to show row numbers**. If you don't care about row numbers in your data, use IEnumerable<Customer>.
4. It stores that large chunk of data in Session or Cache as you like.
5. It chooses *n* items from there, and puts it into ViewBag.
6. On the view, there is a <table> with a CSS class infinite-scroll.
7. The view is rendered, and the data that's been put into ViewBag is sent to a partial view that contains the repeated HTML. In the sample's case, it's a table row and related columns.
8. JavaScript is loaded to handle the event of scrolling the page to the bottom.
9. When the page is scrolled to the bottom, an AJAX request is sent to a controller action method, along with apageNumber value. The Session data is reloaded and the next *n* items are taken, offset by the pageNumber.
10. That partial view is rendered with the next *n* items, and the rendered HTML is returned back to the page.
11. The JavaScript then appends the rendered HTML to the <tbody> element of the element with class infinite-scroll.
12. The row appears, and the user can scroll again. Rinse and repeat until there are no more rows to deliver fromSession.