Memory and Performance Issues

Large Collections

Collections with a large number records (500 records are more), consume a large amount of memory to store the corresponding field objects. Because a field object is created for each column of the collection record, the number of field objects could grow quite fast. In addition to this, each field has a number of related objects attached to it that makes the size relatively large.

This leads to the following issues:

1. The table layout manager or stacked layout manager requires a large amount of memory to store the related field objects.
2. These layout managers are stored within the view, that is then placed on the form and stored in session. Note a view may also contain multiple collections.

Potential Improvements

1. Reduce the size of each field object: many of the nested objects on field are initialized for convenience and are not used.Therefore the size of the field object can be reduced by nulling out these objects (possibly by looking at the render flag and if false nulling the object)
2. Have option for not storing the collection group in session. This will prevent some server side operations
3. Look at using a FlyWeight pattern. Since generally there is little difference between the field objects (only property name is different in most cases), space could be saved by using a FlyWeight pattern. Although knowing when a new instance does need to be created could be tricky.
4. Provide client side rendering functionality for cases where the collection is read only.
5. Perform actual server side paging of results (currently all the records are sent to the client and paged in the client)
6. Provided ability to limit displayed records through collection group
7. For search results, do component refresh for results group instead of full submit (therefore only lifecycle will be run on collection group and only collection group will be rendered)

Form Session Storage

The amount of memory required to store forms in session can grow quite rapidly. This is primarily due to the lack of removing the session storage once the form is no longer needed.

Strategy for Storing Forms

The current strategy for storing forms in session is as follows:

- After the request is completed the current form is stored in the session object with a form key (used to pull the form out)
- The form key is taken from the request if present (either by a request URL or Form parameter). If the form key is not present on the request, a new key is generated.

Therefore generally get requests create new forms in the session, while post request will override the form in session. As a consequence of this (and no removal being done), a user can fill all the server memory by continuing to make a get request (such as a new document request).

Potential Improvements

1. Store and retrieve the form from the database instead of the session object. Currently the form is stored to the database at the end of each request so it can be retrieved when the session times out. This would change to not storing the form in session but only the database. The form would be retrieved once at the beginning of each request. Note problems were found with the document form database storage in the KNS and we need to find out more about those.
2. Implement various form clearing points. Some could be:
    a. When an action is requested that navigates back (e.g. cancel, close, return value, return to previous/home)
    b. Catch exit actions client side and send a request to clear (e.g. closing tab/window, navigating the portal)
3. Implement a timeout on session objects. After a configured period of inactivity on that object it will be cleared from session. Note if a request is made later for that object it can still be retrieved from the database.
4. Implement a memory watcher for the session. After a configured amount of memory is being used up the objects begin getting paged to the database.
5. Reduce the size of the form in session. The form can be serialized to a string and then compressed.
6. Limit that number of form objects that can be stored for a given view. For example, allowing only two instances of a document to be open at a time.
7. Option on the View component for not storing the form in session. On by default for InquiryView
8. Option on all components to not store the component in session

General

1. Look into template invocation and reducing number of templates invoked
2. Look into size of view tree and number of recursions
3. Profile building of view for bottlenecks

4. Optimized VM arguments
5. Pre-built views or groups?
6. Compress Data Before Transfer
    a. take advantage of browsers compression functionality allowing the server to send compressed HTTP data to the client. For regular text content like HTML, CSS and JavaScript, this compression technique can reduce your data transfer with as much as 90%. Although compression and decompression slightly increases the CPU usage on both the client and the web application server, it's still much faster than to transfer an excessive data volume.
    b. minimize java script text size. In looking at jquery code, i noticed that the code is very compressed. small variable names, no whitespace. I assume this is to minimize the footprint. While it does make the code very difficult to read/debug, it would be interesting to see the effect on performance. Consider production/debug option to "pack/unpack" js code.

Optimization Results

Test: Book Lookup, 200 records, 6 columns, read-only

Change	Response Time	Page Size	Server Form Size
None	3.73 S	7.23 MB	
Simple table with client render	1.03 S	90 KB	
Simple table with direct datatables render	1.03 S	53 KB	

(See also Performance Assessment - KS / My Plan.)