7/11/2011

* BSR Web tool
  + About 50,000 records converted and combined for 6 regions: CM, CO, MA, NE, WS, OS
  + Web tool created for BSR to Filter/Search for records on the BSR table and display the results in a GridView for subsequent selection.
  + Performance is diminished when such a large number of records is read into the GridView via the SqlDataSource. After much experimentation and research, the issue is with the ASP.NET control of the GridView via the SqlDataSource. The problem is that ASP.NET loads all the records (e.g. 50,000) into the grid even though it only shows 10 records at a time. I was able to implement two methods of caching which greatly improved the performance, but the tool is not yet at its peak performance level that it could be if it did not have to store all 50,000 records in the GridView. The way to solve this issue is to implement “custom paging and sorting”, which requires redesigning the GridView architecture with my own coding instead of using the given controls from ASP.NET. To do this, I will need to use the more flexible ObjectDataSource control instead of the SqlDataSource control.

7/18/2011

* BSR Web tool – newly created performance improvement!!!
  + In the past, performing a function on all 50,000 combined records in the new BSR tool would take 80 – 90 seconds (e.g. paging from one page of data to another or sorting a column, regardless of how many records were selected). Now, the BSR tool takes only 3 seconds to perform a function when options “All Regions” and “All Platforms” are selected!
  + The key was to write my own customized functions instead of utilizing the out-of-the-box controls associated with the ASP.NET GridView. It is a known fact that the GridView’s paging and sorting mechanism is inefficient when there are many records involved, since it reads into memory all of the records (say 50,000) instead of just retrieving the page of records selected (say 10). Now the program does retrieve only those 10, and the performance is greatly improved in obtaining the next page of 10 records and in sorting the entire subset of records (perhaps a few hundred or a few thousand).
* Merging PSR into BSR
  + Now that we have the ability to efficiently process 50,000+ records in a single table, I brought up the idea, again, of merging the PSR tool into the BSR tool.
  + These two tools are VERY similar in purpose, architecture, table fields, etc., more so than any two other tools.
  + Speaking with West representatives from ESP admin, ESP scheduling, and ABM, they were excited about the possibility and agreed that the two tools should be one tool, which will make it better for the users who currently have to determine which tool to use.
  + James is whole-heartedly on board (and excited) with the idea and he is the best person to scope out the territory more, by assessing the feasibility of such a merger and determining the challenges involved. I’ve asked him to look into this further when I am gone on vacation to Panama next week, though he has made good progress with this assessment already.

8/1/2011

* + Performance was lacking when loading the BSR Web tool or changing a region option or changing a platform option on the tool, which took 9 or more seconds.
  + With a change made to the query SELECT statement to include a dummy WHERE clause (in App\_Code/ShemaForTable.cs), changing options for the Region or Platform is performed in a fraction of a second!

<asp:GridView ID="GridView1" runat="server"

AllowPaging="True"

AllowSorting="True"

DataKeyNames="RefNum"

DataSourceID="ObjectDataSource1"

<%-- SelectMethod is called with parameters when selecting --%>

<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"

TypeName="TableForODS"

SelectMethod="GetPagedDataSet"

EnablePaging="true"

SelectCountMethod="GetRowCount"

StartRowIndexParameterName="startRowIndex"

MaximumRowsParameterName="pageSize"

SortParameterName="sortColumns"

OnObjectCreated="ObjectDataSource1\_ObjectCreated"

OnSelecting="ObjectDataSource1\_Selecting"

OnSelected="ObjectDataSource1\_Selected" >

<SelectParameters>

<asp:Parameter Name="table" Type="String" />

<asp:Parameter Name="type" Type="String" />

<asp:Parameter Name="region" Type="String" />

<asp:Parameter Name="platform" Type="String" />

<asp:Parameter Name="filter" Type="String" />

</SelectParameters>

</asp:ObjectDataSource>

<%-- ODS Parameters are set in method ApplyFilter of class

CodeFormView.cs (being passed from page's CodeBehind

method FilterBySchema1\_ButtonClick) when a Filter button

is pressed, thus activating the selection process at

that time. --%>

public DataSet GetPagedDataSet(string table,

string type, string region, string platform, string filter,

int startRowIndex, int pageSize, string sortColumns)

{

if (sortColumns.Length > 0)

\_sortColumns = sortColumns;

// If you want to set the default sort order in the data class,

// here's one place you could do it.

//else

// \_sortColumns = "LastName, FirstName";

return GetDataSet(

AssemblePagedSelectSql(table, type, region, platform, filter,

startRowIndex, pageSize));

}

private string AssemblePagedSelectSql(string table,

string type, string region, string platform, string filter,

int startRowIndex, int pageSize)

{

string whereClause = BuildWhereClause(table, type, region, platform);

StringBuilder sql = new StringBuilder();

// The "core" select statement is: "SELECT RefNum ..."

// The rest of this code inserts a RowNumber column into the result set

// and wraps the entire select with paging conditions.

sql.Append("SELECT \* FROM (");

sql.Append("SELECT \*");

sql.AppendFormat(", ROW\_NUMBER() OVER (ORDER BY {0}) AS RowNum ", \_sortColumns);

sql.Append("FROM dbo." + table + " ");

sql.Append(whereClause + " AND " + filter);

sql.Append(") AS PagedResults ");

//The startRowIndex supplied by the ObjectDataSource is indexed starting at

//zero, whereas the ROW\_NUMBER() value returned by SQL Server 2005 is indexed

//starting at 1.

//startRowIndex value is 0-based, but the rows in the result set start at 1.

//Therefore, the WHERE clause returns those records where

//RowNum is strictly greater than startRowIndex and less than or equal

//to startRowIndex + pageSize. (E.g. startRowIndex is 0, pageSize is 10)

sql.AppendFormat("WHERE RowNum BETWEEN {0} AND {1} ",

(startRowIndex + 1).ToString(), (startRowIndex + pageSize).ToString());

//sql.AppendFormat(" WHERE RowNum > {0} AND RowNum <= {1}",

// startRowIndex.ToString(), (startRowIndex + pageSize).ToString());

return sql.ToString();

}