NGE/Digital Dashboard

Interface Modifications and Editing Guide

Pierre Berryer

Date: February 2, 2001
Version 1.00

NGE/Digital Dashboard Interface Modifications & Editing Guide – 2/2/2001

Observational Notes/Suggestions

Interface Design Components:

Dbview_ie.xsl	Contains the HTML that creates a page layout
Dbmenu.htm	Contains the HTML/JavaScript that creates the dropdown menu layout
Oneams.css	The revised stylesheet that formats all elements for the designated dashboard
Dbview_ie.js	The Javascript that handles client side dynamic functions
Dbview_ie.xml	Contains the content and tool tips for the top right menu links

Additional Interface Design Components:

cust.css	The Cascading stylesheet that formats all elements for the Content, Layout and Settings screens
dbedit.xsl	Dashboard Settings page
layout.xsl	Dashboard Layout page
dbview.xml	Stores the text string values for certain interface link labels and descriptions. This includes the strings for the top right links.

Primary Images: (The links to these are modified in the section's XSL file except where stated)

close.gif	Webpart Control: The close webpart button
dashlogooneams.gif	Top left Logo icon - Link Modified via CSS
dropdown.gif	Top Menu dropdown arrow icon - Black - Link Modified via CSS
dropdownwhite.gif	Top Menu dropdown arrow icon - White - Link Modified via CSS
minus.gif	Webpart Control: The minimize webpart button
pixel.gif	Single clear pixel used for layout spacing
plus.gif	Webpart Control: The restore webpart button
refresh.gif	Webpart Control: The refresh webpart button
settingsicon.gif	Top Left Logo icon for settings/content/layout sections
up.gif	Top Left Icon for navigating back/higher in the Dashboards tree

Comment Styles

CSS Comments: The standard.CSS file contains comments for the classes contained within. To alter this file, find the description that matches the element you would like to modify and edit the attributes for that class. These pages contain comments in the form of:

```
 /*
   comment
   */
```

HTML/XSL Comments:
You can search the code for the start of these in order to quickly get to editable areas. These pages contain HTML comments in the form of :

```
<!-- comment -->
```

Possible Features:

- Customizable Favorites menu with links?
- Lotus Notes Multiple Desktops – A useful ability. Perhaps accomplished by adding a Favorites menu with a sub folder linking to multiple desktops. Each Desktop could then be customized with the user's choice of web parts? Still not as quickly effective as the current Notes method. Ideally keep the alternate desktop 1 click away.
- Mail Alert Display – Arlette's mail icons used to display priority of emails and special alerts set to incoming mail from user specified senders
- To Do List

Dashboard Editing Guide

This site uses CSS (Cascading Style Sheets) to control font faces and sizes, table cell internal spacing, and an assortment of factors. In future edits to the stylesheet, please continue to use the variable font size descriptors as they are done currently.

Modifying width of webpart columns:
The three columns that hold the webparts are controlled via the stylesheet. The classes used are the **DashZone** series. Modifying the width of these classes allows the webpart columns to be resized. This may require multiple stylesheets to be used for each dashboard that has a differently sized column layout.

Modifying the Top Left Logo Text:
Usability Concern: The text for this is stored in the dbview_ie.xsl file. It should really function as a link to the start page or highest hierarchical level in the site. This requires editing the XSL code to dynamically add this link. Search for the comment:

<!-- MODIFICATION: site name

Modifying the Top Left Logo Image:
This image is called via the CSS as a background image. The class that handles it is **DashTitleImage**. The image should be an extended height (taller than the row it is placed in) to allow for the title font to be adjusted by the user without the image ending abruptly. This appears mainly when the image is sized to display only as tall as the cell when the user's browser has their font set to small. In this case, if the font is then set to medium, the logo image ends before touching the bottom of the cell and the table's background color shows through.

Modifying the Top Right Search and Logout links:
The formatting of these links is stored in the CSS file via the **DashCommand** set of classes. The text and link modifications are made in the XSL file. Search for the comment:

<!-- Top Right Links Section --> or **<!-- Jayma:**

Including additional Stylesheets within Webparts:
When webparts accessing external Certain factors must be taken into account. If styles are defined in multiple stylesheets, the styles cascade. They will sequentially add to and eventually override each other. Keep this in mind if merging stylesheets or calling multiple stylesheets within a dashboard. Apparently, individual stylesheets can be made to affect only certain components, as is done with the dropdown menus.

How is a webpart displayed?
- Webparts are included via the server before reaching the browser. They are then shown inline as contents of a table or included within an iframe and called from an external source.
- When a webpart collapses, it adds an extra pixel row to the bottom as the bottom border is raised. Can this be set to simply collapse to the title bar? Purely a visual detail remark.
- Webparts have no margin around their border. When creating a text based webpart, be sure to include margin space along all sides to prevent the text from directly touching the webpart frame.

Creating the black on yellow Popup Help:
Popup help occurs when the tag contains a title="*description*" attribute.

Directory Structure:
The default directory structure places all files in one folder. This may be a problem in the future as the number of components to the system starts increasing. With the ability for a different stylesheet for each dashboard, custom images, or a custom XSL file, subdirectories separating types of files may be beneficial and may outweigh the extra steps involved in maintaining links within a more complex directory structure.

Page Length/Footer Area:
Care should be taken to avoid excessively long web parts and the resulting excessive scroll. The longest column sets the distance at which the designated footer area for the webparts appears. For example, a long

column on the left will mean that the user will have to scroll a long way down before seeing the footer. This is a fairly important usability concern.

Development & Usability Concerns

Folder Backtracking: Modifying the Top Left Up Folder Icon:
Usability Concern: The problem is that if a user is nested several levels (dashboards) down, the backtracking process could become tedious. This would be most visible when going back more than two levels, backtracking to server intensive dashboards, or using slower connections. These modifications may require some serious development efforts/modifications to the dbview_ie.xsl file. This navigation tool is, however, a potentially major usability issue.

- Dropdown "Breadcrumbs" Menu: A potential solution would be to make the backtrack dashboard link be a dropdown menu itself with a list of the path taken to get where the user is (again either dynamically or a general optimal path).
- Breadcrumbs: Another potential solution to this could be using the top webpart area to store a "Breadcrumbs" listing. This would show the user a series of links to each key dashboard level used to get where they are. This can either be a dynamic or, more commonly, a standard listing that could be used by default. This breadcrumb is a feature common to many sites. An example of what they look like is:
  Home > X-Apps > TimeX > Administrator > Account Editor

Dropdown menu:

- The dropdown menu is displayed using the dbmenu.htm file and its included JavaScript.
- **Usability Concern:** currently the dropdown menus act as a list of links that must be clicked on directly to function. This differs from the standard Windows/IE dropdown menu which allows a user to highlight the line of a menu item and click anywhere in the selected area. This area is indicated via a highlighted box that extends the width of the menu.
- XSL passes parameters later to the JavaScript to control entries, stylesheet, and size of the menu. The JavaScript then uses the Internet Explorer methods of the Table to automatically insert a standard row and cell. This is done by modifying a named table object with the insertRow() and insertCell() methods. To achieve a more conventional highlighted menu, these rows would need to contain the JavaScript that would highlight them (by changing their class) when the user's mouse is over the cell. Also, the cell itself, not just the text contained within, would need to respond to the user's click and link to the correct page. This may require a different JavaScript approach to building the tables.
  - An example of the HTML code that would display the highlight and link properly can be found in: **MouseOver Table Cell.txt**
- Curved bottom corners would need logic to determine first/last rows and whether or not to curve the top corners.
- Dropdown menus may contain styles with pixel size locked fonts. Check for other styles for this problem as well.
- Each dropdown menu can be linked with a different stylesheet. This could allow the creation of special menus that display differently to facilitate user interaction. An example of the need for this might be in the user's favorites menu or a Breadcrumbs menu.

Dashboards build HTML page in one table:
The current dbview_ie.xsl builds each page contained in one table. This can prevent certain pages from displaying before all the HTML has been received. This can slow down display of complex pages unnecessarily. An alternate method would be to design the page to load in chunks allowing the page to begin displaying before the HTML for entire page has been received. If these chunks were stored in individual layers, the order of content appearance could be controlled in detail. This is not related to Webparts loading from sources external to the current page's HTML via the iframes tag (i.e. another website). Once the HTML for a page has loaded, it displays even while waiting for the iframe to load data from the external source.

Search Link Popup form:
The links are currently created using XML as the source for link, text, and title. The search/logout are empty links until their functionality can be implemented. They can be hard coded into the XSL doc as standard HTML or they can be recreated in the same way that the Content/Settings/Layout links are made. To see a demonstration of a popup search layer in which to enter text, load the file **searchpopup.mht.** This should be implemented using similar methods as those used to display the dropdown menus. This should allow the popup

layer to appear in the same relative location on the screen no matter the size of the browser. The method being used on the concept source-site (www.hermanmiller.com) uses JavaScript to determine the current size of the browser then calculate how far from the left the layer should appear. There is probably a much simpler method of accomplishing this that may require only small amounts of JavaScript modification or addition to the current dashboard files, rather than an entirely new set of functions.

Coding Notes:

- Ampersand (&) breaks XSL. Use &amp; to display this character in HTML.
- XML/XSL tags should self terminate when used without a paired closing tag. (i.e. <SingleTag />)
- CSS/XSL sets arbitrary (and potentially incorrect) widths for cells, etc. Should be checked, however, it may not produce visual anomalies.

Design Notes - Applicable during creation of webparts as well

Standard body Text:
Default text is set via the CSS modifying the <BODY> tag. Outside of that, text should generally use a class assigned to it and commented accordingly. The provided standard.css is a bit excessive at times but serves as a good example of what should be handles via the CSS.

Usage of <SPAN>:
Avoid the <SPAN> tag unless you are attempting to apply a style to only a portion of text that has already been enclosed within other class-modified tags. <SPAN> tags cannot be nested within each other.

Font Size:
Dashboard's stylesheets ship with font sizes set using the em unit. This is a relative unit size and can vary with the users choice of default text size (chosen in the IE menu via View -> Text Size). When creating styles for new HTML, this unit can be used or the simpler set of sizes such as xx-small, x-small, smaller, small, medium, larger, larger, , x-large, xx-large. If modifying the existing stylesheets, the existing unit type should be maintained to avoid future confusion and more easily maintain look and feel. Also, avoid using locked font sizes such as points, pixels, picas, or inches unless absolutely necessary. Locked sizes currently prevent users from adjusting their browser to suit their needs, a problem for those with poor or tired eyesite. Also, to maintain CSS benefits, avoid using the <FONT> tag and instead assign classes.

Page Width:
This site defaults to 100% screen width and is set for users to use a 1024x768 minimum screen resolution. Be aware of this when designing webparts and choosing widths for the columns. While this is currently the minimum system requirement, if possible, try and use percentages or to avoid creating an interface that cannot scale down to a smaller window within the user's desktop without horizontal scrolling.

Alignment:
Table contents should generally be Top aligned when there is the chance that one of the row's cells will wrap the text around, creating a double high row. In the case of form field labels, they should be consistent, being either top or bottom aligned.

Cell-Spacing vs Cell-Padding:
Generally, avoid cell spacing unless you're going for a very specific look. Padding will put buffer space between the edge of the cell and its contents using the same color as the rest of the cell. Cellspacing will create space between the <TD> cells of a table letting background colors show through.

Page bottom:
A standard should be chosen for the amount of white space shown at the bottom of a page. A series of 2 line breaks (<BR><BR>) is generally sufficient. This should be extended as necessary if using an intra page link to jump to an anchor at the bottom of the page (i.e. URLs that take you to *#anchorName* on the same page). Without sufficient white space, the page cannot jump down far enough to display the anchor area at the top.

Dreamweaver Bugs:

CSS Problems:
Large CSS files may not function properly within the DW 3 CSS Palette. The bug results in missing styles when editing the sheet this way. If attempting to create styles via this WYSIWIG, you must create the styles in a temporary external stylesheet then manually copy and paste them into the dashboard's stylesheet. Also, DW3 does not allow you to comment a stylesheet or read comments contained within. Any new comments must be entered via a text editor.