AWS, CSS and id selectors

1 Document History

Date	Version	Change Description	
4/29/2014	V1.0	Initial Entry	
4/30/2014	V1.1	More initial entry	
4/30/2014	V1.2	Editing	
4/31/2014	V1.3	Add Bubba's comments	

2 Table of Contents

3 Introduction

The pbuilder utility, a component of NET+OS's Advanced Web Server disposes of id attributes contained in pbuilder comment tags. Given that id attributes are used to direct CSS directives to HTML elements, having id attributes lost is a problem. This white paper addresses this problem.

3.1 Problem Solved

It is a fact that id attributes included in pbuilder comment tags will not be passed to the resultant .c file. If your development of HTML files includes CSS directives, this could be a problem. id attributes can be used for directing a CSS directive to some HTML code. There is nothing that can be done to mitigate the problem of id attributes being discarded. What this paper does is describe a couple of ways to work around the loss of the id attribute.

3.2 Audience

This paper is written for developers with experience developing web applications in the NET+OS development environment, that contain not only HTML elements but also contain CSS.

3.3 Scope

The scope of this paper is quite limited. This paper addresses the disposal of id attributes from pbuilder comment tags, by the pbuilder utility. This paper discusses the problem and then a few ways to fix it.

This paper does not discuss any of the following:

- Developing with the NET+OS development environment
- Using HTML
- Using CSS
- Developing with C or C++
- Debugging
- TCP/IP or network programming
- Anything else not described in the initial paragraph

3.4 Theory of Operation

This paper uses HTML and CSS code samples in describing methods for working around id attributes being discarded by the pbuilder utility.

4 Definitions

Given the number of items, both internal and external to NET+OS, we thought it was important to describe some of them before we get started. The next few sections describe some of the terms you should know about.

4.1 Advanced Web Server (AWS)

If you are reading this paper then you are probably developing an embedded application using Digi International's NET+OS development environment. The NET+OS development environment is made up of a number of components. One of these components is the Advanced Web Server or AWS. This is an embedded web server allowing your application to serve web pages and device data to a web browser, along with fielding requests from a browser. This might include updating device data. A necessary component of the AWS is the pbuilder utility. We will discuss the pbuilder utility in a few sections.

4.2 Hypertext Markup Language (HTML)

To quote from the Wikipedia entry:"HTML or Hypertext Markup Language is the standard markup language used to create web pages". HTML is made up of pairs (generally) of tags such as <html></html> and <body></body>. In this case <html> opens the tag and </html> closes the tag. A web page is made up of numbers of these tags along with text.

4.3 Cascading Style Sheets (CSS)

Cascading style sheets are additional directives allowing the style of a markup language file to be modified. The directives can be included in the markup language or can be included in a separate file. For simplicity, the examples used in this paper are included in the markup language file. Briefly, CSS style sheet directives look something like the following:

```
p {color:green;}
.mydiv {background-color:purple;}
```

4.4 Pbuilder utility (wizard)

Web pages are written in HTML. Cascading style sheets are written in their own language. Images can be in jpeg, tiff and many other formats. Programs run within the NET+OS environment are generally written in C, C++ or assembly. The purpose of the pbuilder utility is to convert the HTML, CSS or other file formats into a C file. Those HTML, CSS or other files formats can then be included into a NET+OS application.

As a note, in the title we used the term utility and parenthetically included the term wizard. When the pbuilder is run from within ESP, pbuilder is referred to as the pbuilder wizard. When pbuilder is run standalone from the command line, it is referred to as the pbuilder utility. At the end of the day they are (more or less) the same thing. The main difference is that running the pbuilder utility from the command line, you do not get the advantage of the merge tool. When running pbuilder from within ESP, you do get to use the merge tool.

4.5 Pbuilder comment tags

As stated earlier, web pages are written in HTML. So you could write all of your web pages in HTML and build them into a NET+OS application. The problem is that they would not be very useful because when using pure HTML, you would not have access to device data. In most cases, the reason for building the web pages as part of the NET+OS application, is to give your browser and the underlying HTML files access to your device's data. Pbuilder comment tags add additional functionality to your web pages, by adding callback functions that the advanced web server will use for accessing device data. Pbuilder comment tags generally come in pairs. The beginning tag might look something like <!--RpFormHeader method="get" -->. The ending tag is <!--RpEnd -->.

For additional information about comment tags and APIs that are part of AWS, please see the Advanced Web Server Toolkit document included in the Documentation directory of your installation tree. It is entitled Advanced_Web_Server_Toolkit.pdf.

5 Problem Details

This section presents a small sample web page, including CSS directives, and uses this page to describe the problem we want to solve.

5.1.1 Sample web page

```
<html>
<head>
<title>Test page</title>
<style type="text/css">
#field1 {color:red;
         background-color:yellow;
        }
</style>
</head>
<body>
<!-- RpFormHeader method="get" -->
<form method="get" >
<!-- RpEnd -->
<span style="background-color:white; ">
Class within the element:
</span>
<!-- RpFormInput name="fieldOne" size="20" maxlength="20"
RpGetType=Direct
    RpGetPtr=theFieldOne RpSetType=Direct RpSetPtr=theFieldOne
id="field1" -->
<input type="text" name="fieldOne" size="20" maxlength="20" id="field1"
>
<!-- RpEnd -->

<!-- RpEndForm -->
</form>
<!-- RpEnd -->
</body>
</html>
```

The HTML code above represents a very simple HTML file, including the pbuilder comments tags and CSS directives as described above. The CSS following directive:

```
#field1 {color:red;
         background-color:yellow;
        }
```

requests that elements containing the id of field1 have red colored text and a background color of yellow. So that is fairly straight forward. There are two sets of HTML that we need to describe as follows:

```
<!-- RpFormInput name="fieldOne" size="20" maxlength="20"
RpGetType=Direct
    RpGetPtr=theFieldOne RpSetType=Direct RpSetPtr=theFieldOne
id="field1" -->

<!-- RpEnd -->
```

RpFormInput and RpEnd are the beginning and ending comment tags describing a text input field to a form. As we described earlier, the CSS has a directive starting with #field1.

```
<input type="text" name="fieldOne" size="20" maxlength="20" id="field1"
>
```

This is the HTML describing the same thing. So if you were not running this through the pbuilder utility, you'd point your browser at the HTML file and the browser would see "pure" HTML only. It sees the comment tags as comments. On the other hand, the pbuilder utility would "compile" the comment tags and ignore the "pure" HTML.

You will notice that both examples contain the attribute pair id="field1". This attribute value pair equates to the CSS directive starting with #field1. With this the browser knows to apply the CSS directive to this HTML element.

So if you are familiar with HTML and CSS you'll say to yourself, "all this looks clean, so what is the issue?". The issue is, after you create your HTML file and write your application C code, you will run your HTML, CSS and image files through the pbuilder utility. If you looked at the C code generated by the pbuilder utility, you would notice that the id=field1 attribute value pair is missing from the pbuilder generated HTML. If you had developed a set of HTML files, containing CSS formatting directives that were associated with HTML code using id attributes, all that formatting information would be lost from your application. Without the id attribute in the final form HTML, there would be no way for the browser to associate the CSS with the HTML elements. Hopefully you are now saying to yourself, "Aha now I see the problem".

6 Potential Solutions

We will explore three possible solutions. We have tried them all and they all present positive results.

6.1 Use class attribute in place of id attribute

This first solution is arguably the simplest. As stated earlier the pbuilder utility will not pass id attributes to the C file it generates. The pbuilder utility will, though, pass class attributes to the C file it generates. So change your id attributes to class attributes. This involves two steps. You will have to change both the CSS entry and the pbuilder comment tag.

6.1.1 Changes to the CSS entry

Change the CSS entry from this:

```
#field1 {color:red;
         background-color:yellow;
        }
```

To this:

```
.field1 {color:red;
         background-color:yellow;
        }
```

In CSS grammar a '#' signifies an id attribute while a '.' signifies a class attribute.

6.1.2 Changes to the pbuilder comment tag

Change the pbuilder comment tag from this:

```
 <!-- RpFormInput name="fieldOne" size="20" maxlength="20"
RpGetType=Direct
    RpGetPtr=theFieldOne RpSetType=Direct RpSetPtr=theFieldOne
id="field1" -->

<!-- RpEnd -->
```

To this:

```
<!-- RpFormInput name="fieldOne" size="20" maxlength="20"
RpGetType=Direct
    RpGetPtr=theFieldOne RpSetType=Direct RpSetPtr=theFieldOne
class="field1" -->

<!-- RpEnd -->
```

Notice we changed the "id=field1" attribute value pair to "class=field1".

6.2 Use a div element with a class attribute

The next solution we present involves the use of the div element. The div element allows you to break your HTML into blocks. We will use this to identify blocks and apply attributes to those blocks.

6.2.1 Changes to the CSS entry

First, the new CSS directive we need looks like the following:

```
div.field3 input {color:white;
         background-color:gray;
         }
```

div means look for a div element. .field3 means look for a class attribute named field3. So div.field3 means look for a div element with an attribute of field3 and apply the CSS directives as needed. In our case, color:white means make the text color white. background-color:gray means set the background color or the input text field gray.

6.2.2 Changes to the pbuilder comment tag

The associated pbuilder comment tag looks like this:

```
<div class="field3" >
Class within a div element:
<!-- RpFormInput name="fieldThree" size="20" maxlength="20"
RpGetType=Direct
    RpGetPtr=theFieldThree RpSetType=Direct RpSetPtr=theFieldThree -->

<!-- RpEnd -->
</div>
```

In this example, we have removed the class or id attribute from the pbuilder comment tag. In its place we have surrounded the comment tags with a div element with a class attribute whose value is field3. Since our CSS calls for a div with an attribute of class and a value of field three, our CSS will be applied to this for input.

6.3 Use a div element with an id attribute

The third and final solution is similar to the last one, except instead of giving the div element a class attribute, we will give the div element an id attribute. We will also use a slightly different CSS directive.

6.3.1 Changes to the CSS entry

The CSS directive we will use looks like the following:

```
div#field5 input[type="text"] {color:gray;
         background-color:red;
         }
```

div refers to a div element. #field5 refers to an id whose value is field5. Please note that a class is prefixed with a '.', while an id is prefixed with a '#'. The input means to apply it to the form input field of the HTML. The [type="text"] modifies the form input to a form input whose type is text. As before color:gray means to set the text in the form input box to gray. Background-color means to set the background of the form input to red.

6.3.2 Changes to the pbuilder comment tag

To go along with the CSS entry above, HTML and pbuilder comment tags would look like the following:

```
<div id="field5" >
id within a div:
<!-- RpFormInput name="fieldFive" size="20" maxlength="20"
RpGetType=Direct
    RpGetPtr=theFieldFive RpSetType=Direct RpSetPtr=theFieldfive -->

<!-- RpEnd -->
</div>
```

Notice that the div element has an id attribute whose value is field5 and thus would be associated with the CSS directive above. The attributes of the CSS are applied to anything appropriate between the and the tags.

7 Odds and ends

This section discusses some additional topics that did not fit in with the main points of the paper, but may be useful, just the same.

7.1 The merge tool and HTML file changes

When you create an HTML file(s), representing a web page(s) and run it/them through the pbuilder utility, two main files are created. One is the .c file and one if the _v.c file. The .c files contain structures and HTML boilerplate. It is generally recommended that you do not edit this file. The _v.c file, contains stub functions that give your web pages access to your device data. You fill in the _v.c file (you define the stub function internals), thus you must edit the _v.c file. If you add additional fields to your HTML file(s) and rerun your application through the pbuilder wizard, the pbuilder wizard will invoke the merge tool. The merge tools allows you to add the new/additional stub functions to your existing _v.c file without losing the C code you have previously added (when you filled in the stub functions). On the other hand, if all you do is change CSS directives, ids, classes and div elements, you are NOT adding new stub functions. Due to this, the pbuilder wizard will not invoke the merge tool. This is not a bug. Because you are advised not to edit the .c file, the .c file is generally overwritten each time you run the pbuilder. Since the file is overwritten, there is nothing for the merge tool to merge.

7.2 Debugging note

At this point we will digress for a moment to give you a debugging tip. Let's assume you have developed a NET+OS-based web application. You have run your HTML et al through the pbuilder utility, have compiled and built the application and you are downloading the application to your device. You then surf to your device and notice that all of your CSS-based formatting is missing. Further, you notice that if you point your browser directly at your HTML files, all of your formatting returns. What is going on? A good place to start is to use the "view source" feature of your browser. Most browsers have this though it may be accessed through different buttons, pull-downs or other means. Print out the HTML source for both the pure HTML view and the pbuilder utility HTML view. Compare the two. In our case it should be self evident that the page missing the formatting is also missing the id attributes.

8 Example Application Explanation

The application that accompanies this white paper has an example of each of the methods described in the paper. You should feel free to experiment with the sample application. It is simple but describes the essence of what you need to know to deal with the lack of an id attribute in pbuilder comment tags. We hope you find this useful.

You will find the example application here

9 Conclusion

Yes it is true that id attributes are not passed through the pbuilder utility. We believe, though, that we have provided three viable options for solving this issue and allowing you to include the richness of CSS in your web-based applications.

10 Appendix

10.1 Glossary of terms

- Attribute

Dictionary.com defines an attribute as follows: "A word or phrase that is syntactically subordinate to another and serves to limit, identify, particularize, describe or supplement the meaning of the form with which it is in construction". W3Schools defines HTML attributes as "Providing additional information about HTML elements". In this case, an attribute is a piece of data added to either some HTML or CSS that, as stated, adds additional information to it. Generally the attribute is presented in the form of the attribute followed by an equals sign (showing assignment) followed by the value assigned to the attribute. So in the case of "class=field1", class is the attribute and field1 is the value. The value is generally displayed in double quotes.

- AWS

Advanced web server. An embedded web server included with the NET+OS development environment.

- Comment tag

An element that looks like a combination of an HTML comment and an HTML element. They are used in combination with the pbuilder utility to provide HTML elements and callback names to be used in NET+OS-based web application.

- CSS

Cascading style sheet, as described in Wikipedia, "is a style sheet used for describing the look and formatting of a document written in a markup language." Further, "CSS is designed primarily to enable the separation of document content from document presentation, including elements such as layout, colors and fonts".

- HTML

Hypertext Markup Language according to Wikipedia, "is the standard markup language used to create web pages. HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets".

- NET+OS

An embedded operating system and development environment developed and distributed by Digi International.

- Pbuilder (utility/wizard)

A software utility distributed as part of the advanced web server which is a component of the NET+OS development environment. pbuilder's purpose is to act as an HTML (and CSS and image) to C interpreter. It is available standalone from the command line and as part of a pbuilder wizard contain within Digi's ESP IDE.

- Selectors

According to the CSS selector reference section of w3schools.com, "In CSS, selectors are patterns used to select the element(s) you want to style".

10.2 Citations

HTML from Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/HTML . Creative Common Attribution Share-alike License.

Cascading Style Sheets, form Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Cascading_Style_Sheets. Wikemedia Foundation

Attribute. Dictionary.com. http://dictionary.reference.com/browse/attribute. Copyright 2014 Dictionary.com

HTML Attributes. http://www.w3schools.com/html/html_attributes.asp. Copyright 1999-2014 Refsnes Data.

CSS Selector Reference. http://www.w3schools.com/cssref/css_selectors.asp Copyright 1999 - 2014 Refsnes Data