by Tom Muck • *www.dwteam.com*

Dreamweaver's Built-in Power Extension: Find/Replace

How many times have you said "I wish there was an extension to remove <insert your tag here> tags". Or "I wish there was an extension to remove table formatting". Or how about this one: "I wish there was an extension to change <font> tags to CSS". This article is going to explore one of the most underused areas of Dreamweaver and UltraDev: the Find/Replace dialog. This dialog box in DW can perform a thousand-and-one functions – and will even work site-wide or computer-wide.

When you are editing a document in MS Word, Notepad, or any other text editor of choice, the find/replace dialog can be useful for finding and replacing words or combinations of words. In Dreamweaver, however, the dialog box takes on a whole new persona. You can search through text, or you can search through source code. You can search for individual tags, or you can use regular expressions. If you haven't experienced the power of regular expressions, you're in for a surprise.

Another great feature about DW's find/replace dialog box is that the queries can be saved and reloaded at a later date. The Configuration folder of DW is known well by DW users as the place where all of the functionality of DW is stored. All extensions reside here. One area of extensibility that hasn't been fully utilized by developers is the find/replace query. These extensions are XML files with a .dwr file extension that reside in the Queries folder. The extensions can be created by anyone. They are simply XML representations of the find/replace query that you execute from within the find/replace dialog box. By clicking the save icon you can save the query. By clicking the Open icon, you can open an existing query.

Find/Replace Source

Let's start with a simple one. We want to find all references to Copyright © 2001 in the site. This is a simple case of scanning the source code for the following string:

```
Copyright &copy; 2001
```

For the settings, you'll want to choose the following:

Example 1

Find In	Entire Local Site
Search For	Source Code
Find	Copyright &copy; 2001
Replace With	Copyright &copy; 2002
Match Case	unchecked
Ignore Whitespace differences	checked
Use Regular Expressions	unchecked

That was almost too easy, but the one key thing to remember about this query is that Ignore Whitespace Differences has to be checked. When DW formats the HTML in your document, it there may be times when line breaks are inserted in places you wouldn't expect. The above example might appear like this in your document:

```
Copyright
&copy; 2001
```

If you hadn't checked the box, this occurrence would not have been replaced.

What happens if your copyright notice looks like this in the source code?

```
Copyright &copy; <b>2001</b>
```

Now, the same query won't work. You might think that you can search Text instead of Source, but when you do that, you have to use actual text and not the HTML encoded text:

Example 2

Find In	Entire Local Site
Search For	Text
Find	Copyright © 2001
Replace With	Copyright © 2002
Match Case	unchecked
Ignore Whitespace differences	checked
Use Regular Expressions	unchecked

One concern here is that when you replace the text, the bold tags will be lost on the replace as well. What can we do? This is one area where regular expressions can come into play:

Example 3

Find In	Entire Local Site
Search For	Source code
Find	(copyright[^&]*&copy;[^2]*)(2001)
Replace With	$1 2002
Match Case	unchecked
Use Regular Expressions	checked

If you are unfamiliar with regular expressions, this one needs some explanation. Regular expressions use pattern matching, much like when you perform a Find Files function on your computer using an asterisk (*.txt) only much more powerful. For a full set of regular expression patterns, consult a JavaScript book, such as *JavaScript: The Definitive Guide*. For now, I'll explain the regular expression in Example 3:

```
(copyright[^&]*&copy;[^2]*)(2001)
```

The first thing you should take note of is that there are two groups:

```
(copyright[^&]*&copy;[^2]*)  is the first group
```

```
(2001) is the second group
```

The first group is the part of the code that will stay the same. We can represent this as $1 in our Replace dialog. When you have groupings in regular expressions, you refer to the groups as $1, $2, $3, $4, etc.

This way, you can retain sections of your code while replacing others. The code is broken down as follows:

`copyright`	self-explanatory. This will be an exact match of the word
`[^&]*`	anything and everything up to a & character
`&copy;`	again, this is an exact match of the html representation of the copyright sign
`[^2]*`	anything up to a 2 character

The second grouping consists of the number 2001. This will be replaced in our query by using the first saved group ($1) and substituting the number 2002 for the remainder. By doing this, we are effectively replacing all occurrences of the copyright statement, and keeping any existing tags in place. On a huge site, this could be a timesaver.

Strip Table Formatting

Here's another example of a Find/replace query using a regular expression. Suppose you want to clean all formatting from your tables in your site, or in selected pages in your site. Obviously you want to keep all of the content within the tables in place, but there has to be an easy way to remove the extraneous attributes. We could use the Specific Tag functionality of the find/replace dialog, but you would have to strip attributes out one at a time, or create a complex list of attributes. Let's build a regular expression to do the job in one step.

One of the features of regular expressions is the ability to choose one match OR another match. For example, to find all table, tr, td, or th tags, you could use this:

```
<table|tr|td|th>
```

So far, this will find all opening tags with no attributes. Let's add an expression to find closing tags as well:

```
<\/?table|tr|td|th>
```

Now the expression tells us to find all tags that may have an optional slash character (/) for a closing tag. Since the slash is a reserved character, we escape it with a backslash. The question mark tells us that there may be zero or one of the preceding single character. This will now find all opening and closing tags.

Now, let's add an expression to find the tags with attributes:

```
<\/?table|tr|td|th[^>]*>
```

We've added this expression [^>]* to say "give me anything up to a closing > character." The expression isn't finished, however, because we need to retain the actual tag when we do the replacement. To do that, we'll need a grouping level:

```
<(\/?table|tr|td|th)[^>]*>
```

We put the parenthesis around the optional closing tag slash symbol and the entire tag name. This ensures that we will be able to replace opening and closing tags, and be able to remember which are which. Now for the replacement string:

```
<$1>
```

That's it. If you run this on the sample page, all of the attributes are stripped out of the table cells, rows, headers, and the table itself. Click the Save icon on the find/replace dialog box to save this query for future use.

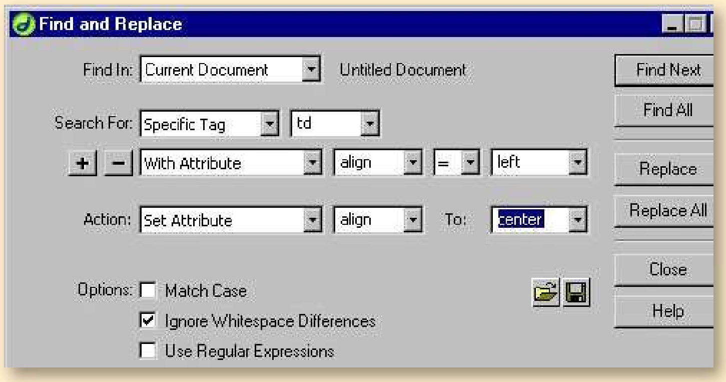
A much simpler query to replace all <th> tags with <td> tags using regular expressions could be written like this:

```
Find   (<\/?)th([^>]>)
Replace with  $1td$2
```

Notice that the opening and closing parts of the tag are grouped ($1 and $2) so that the only thing that is being replaced is the "th".

Using the Tag Replace Functionality

One of the most powerful aspects of the find/replace dialog is the tag functionality:



Find In: Current Document Untitled Document
Search For: Specific Tag td
With Attribute align = left
Action: Set Attribute align To: center
Options: ☐ Match Case
☑ Ignore Whitespace Differences
☐ Use Regular Expressions

Figure 1: Find/replace using tags

The query in the preceding section could have been done using the tag replace functionality of the find/replace dialog. These queries can be saved as well. Simply set it up as follows:

Example 5

Find In	Current Document
Search For	Specific Tag th
+ -	(click - to remove all choices)
Action	Change Tag
To	Td

Changing a tag is just one of the many types of functionalities available for working with tags. Some of the others include:

Search for Specific Tag	
With attribute	attribute name
Without attribute	attribute name
Containing	text
Not Containing	text
Inside tag	tag name
Not inside tag	tag name

You can also group these together, such as	
Search for specific tag	font
With attribute	size
Without attribute	color

Action	
Set Attribute	color
To	#000066

Some of the other actions available for tag matches are as follows
Replace tag and contents
Replace contents only
Remove tag and contents
Strip tag
Change tag
Set attribute
Remove attribute
Add before start tag
Add after end tag
Add after start tag
Add before end tag

Site-Wide and Folder-Wide Searches

The find/replace dialog can search your document, or you can select documents in your site window to allow searching of specific documents. Even better, you can perform site-wide find/replace queries. One thing that you have to be aware of, however, is that these types of matches cannot be undone. If you strip all formatting from a particular tag and later realize that several of your pages didn't need the functionality removed, you will have to edit the pages by hand to restore the functionality - undo works only on the current document.

Find/replace also works on folders in your hard drive. You can choose the Folder option from the Find In dropdown list and then browse to any folder in your hard drive. It can even be your entire C: drive - find/replace will scan for text using more powerful options than you have with the standard operating system find dialog box.

Conclusion

You can see that the find/replace dialog box has a lot of possible options beyond simple searching and replacing of words and code. You can use it to reformat entire documents or sites with a few simple keystrokes. You can also save complex queries and trade them with other developers, or package them as extensions. Exploring the functionality can be an eye-opening experience.