Table Extractor Guide

Table extractor is a php class that can extract almost any table from any html document/page, and then convert that html table into a php array, read on for a description of all the features, and ho to use them. The script can extract almost any table from any html page, use the table column headers as the keys in the array, strip unnecessary html tags out of the source, create extra columns based on data in others using regular expressions and a lot more.

The script does have the following limitations:

- It does not support tables that have other tables nested inside them (there's no error checking, so you will probably just get a complete mess returned). This is on my list of features to add, when I have time.
- It does not check that the structure of the table is actually valid, if invalid html is sent (eg. missing </tr> tags) then it will probably screw up.
- Although I have tested it myself, I make no grantees that it's perfect, in fact I'm sure its not. If you find problems, or a particular table it does not work with, please let me know (e-mail link at the bottom of the page).

NB. All the examples here have been outputted to a browser using my PHP Dump class, so that it's easy to see what the returned arrays contain.

In these examples I was working with a table that looked like this:

Name	Price	Stock	Info
New Product 1	£10	12	More Info...
New Product 2	£20	45	More Info...
New Product 3	£30	1	More Info...
New Product 4	£40	7	More Info...
New Product 5	£50	68	More Info...
New Product 6	£60	2	More Info...
New Product 7	£70	57	More Info...
New Product 8	£80	4	More Info...

The first step in using the class is to include the class file, and the create a new tableExtractor object:

```
include 'tableExtractor.class.php';
$tx = new tableExtractor;
```

Now you have to give it the source html code that contains the table you want to extract. This could either be from an existing string, a local html file, or a remote webpage.

```
$tx->source = file_get_contents('sample1.html');
```

The next thing to do is to tell the script which table you want to extract. To do this you have to give it an anchor point. This is a string that only appears in the source once, and appears just before the table you want to extract. The script will search through the source from this point, and extract the first table it comes to.

```
$tx->anchor = '<h2>Our Products</h2>';
```

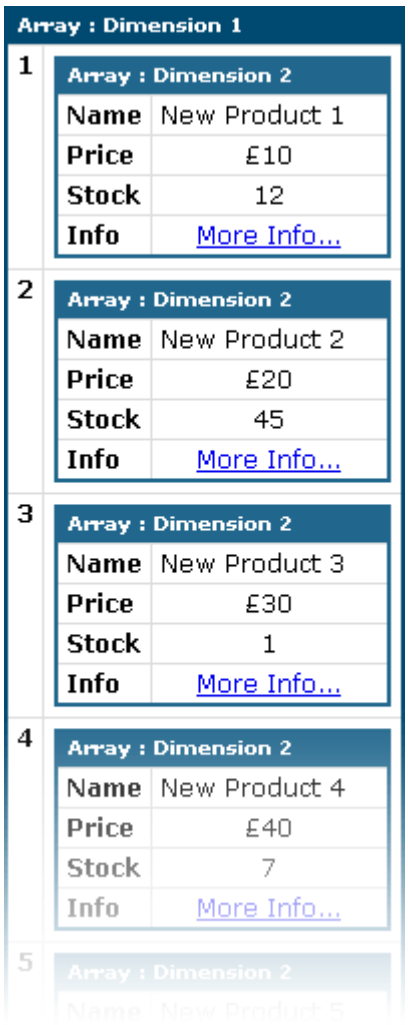
You can also specify an anchor that is within the table you want to extract, to do this use the option above and set anchorWithin to true:

```
$tx->anchorWithin = true;
```

The final step is to extract the table.

```
$tableArray = $tx->extractTable();
```

You can then do whatever you want with this array, I have sent it to my PHP Dump script so that I can preview its contents:



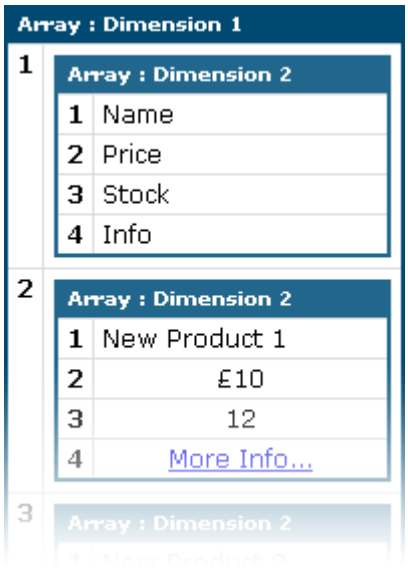
For basic, standard functionality, that's about it. You now have the table, but in an array. Notice that the column headers are used for the keys in the array, if you have a table that doesn't have column headers, see the next section.

Column Headers

By default, the values in the first row of the table are used as the keys for the 2nd dimension array. If you don't want the script to do this, if for example you have a table that has no column headers, just add this line before the call to the extractTable() method:

```
$tx->headerRow = false;
```

That will make the script treat the first row as a normal row. This is the output now:



Array : Dimension 1

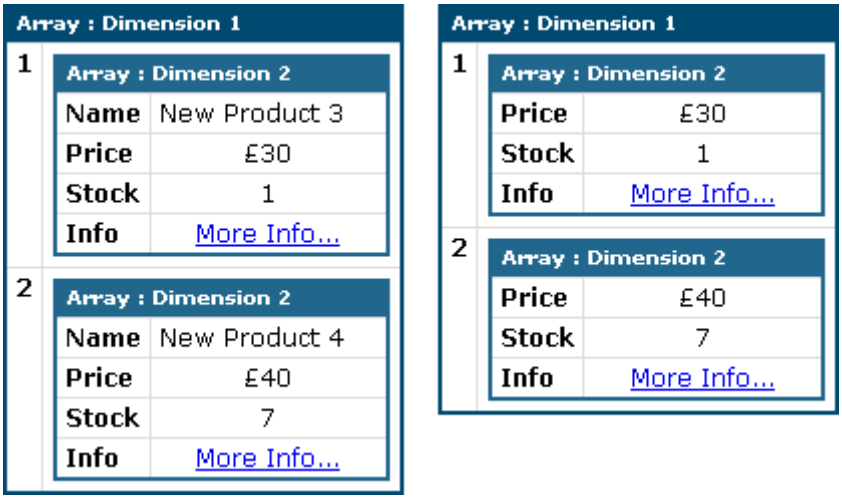
	Array : Dimension 2	
1	1	Name
	2	Price
	3	Stock
	4	Info
2	Array : Dimension 2	
	1	New Product 1
	2	£10
	3	12
	4	More Info...
3	Array : Dimension 2	

Limiting Rows & Columns

There are also a number of variables available to set the start row, start column, maximum rows and maximum columns. These are what you need to set:

```
$tx->startRow = 3;
$tx->maxRows = 2;
$tx->startCol = 2;
$tx->maxCols = 0;
```

They are all pretty self explanatory. If the maximum rows and columns options are set to 0, the script will return all rows/columns from the start row/column. Here is an example of the table with limited rows, and then with limited columns as well:



Array : Dimension 1

	Array : Dimension 2	
1	Name	New Product 3
	Price	£30
	Stock	1
	Info	More Info...
2	Array : Dimension 2	
	Name	New Product 4
	Price	£40
	Stock	7
	Info	More Info...

Array : Dimension 1

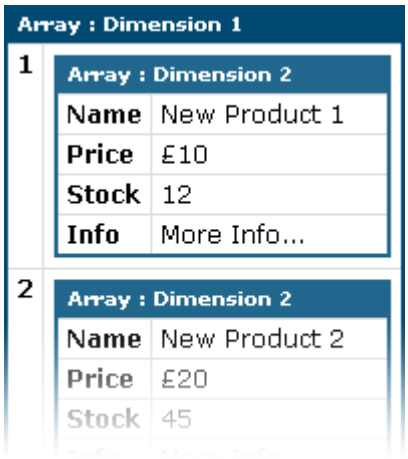
	Array : Dimension 2	
1	Price	£30
	Stock	1
	Info	More Info...
2	Array : Dimension 2	
	Price	£40
	Stock	7
	Info	More Info...

Stripping HTML Tags

If you just want the text data, Its quite likely that you wont want all the extra tags that are stored in the table (such as a tags and img tags). To remove these, add this line to your code:

```
$tx->stripTags = true;
```

That will strip all html tags from the final data:



Array : Dimension 1		
1	Array : Dimension 2	
	Name	New Product 1
	Price	£10
	Stock	12
	Info	More Info...
2	Array : Dimension 2	
	Name	New Product 2
	Price	£20
	Stock	45

Extra Columns

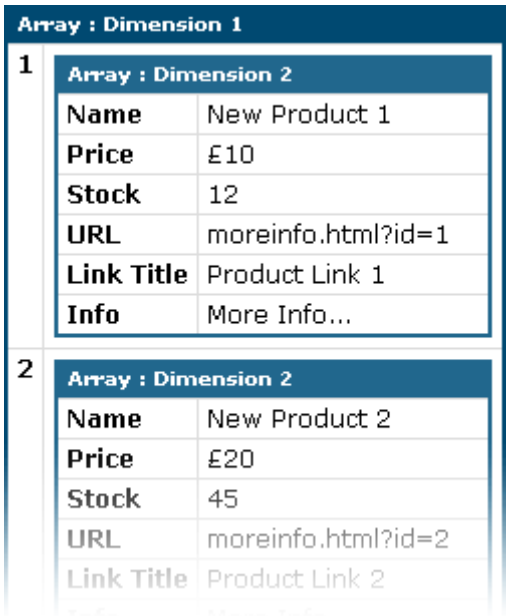
This is the clever bit. You might be thinking “I want to remove all the tags, but I also need the URL that the link for that record goes to, how can I do that if all the html tags have been stripped out?”. This is where what I've called “extra columns” comes in.

With the extra columns feature you can create new columns in your returned array, whose values are based on data from other columns. This data is extracted from a particular column based on a regular expression. What's more, the data is extracted before the tags are stripped, so you can get values from html tags even though they wont appear in the final array. The syntax for this is as follows:

```
$extraCols[] = array('column'=>4, 'names'=>array('URL', 'Link Title'), 'regex'=>'/<a href="([^>"\']*)"
title="([^>"\']*)">/is');
$tx->extraCols = $extraCols;
```

First off, $extraCols is an array, so you can extract data from as many different columns as you want. The first element “column” is the numeric index of the column you want to extract the data from (NB. The first columns index is 1 not 0). The next element “names” is either a string with the name of the new column you're creating, or an array of names, if you want to create multiple extra columns based on this original column. The final element is “regex”, this is the regular expression that will be run on the original column, and the matches will be saved into the appropriate extra columns.

I hope that makes sense, this is an example of the table with those extra columns added, but the original html tags stripped:



Array : Dimension 1

1

Array : Dimension 2

Name	New Product 1
Price	£10
Stock	12
URL	moreinfo.html?id=1
Link Title	Product Link 1
Info	More Info...

2

Array : Dimension 2

Name	New Product 2
Price	£20
Stock	45
URL	moreinfo.html?id=2
Link Title	Product Link 2

Colspans and Rowspans

The final feature I want to discuss is how the script deals with colspans and rowspans. Basically, if the script finds any cells that span multiple columns or rows, it will split them into the correct number of columns/rows, and the duplicate the data in each new cell. NB. This was a nightmare to figure out and I'm pretty sure the code that does it could be improved, if anyone has any ideas for improvements then please let me know. Below is an example of a table with colspans and rowspans, and then how the returned array looks:

<table>
<tr><th>ID</th><th>Col 1</th><th>Col 2</th><th>Col 3</th></tr>
<tr><td>1</td><td colspan="3">Col Span</td></tr>
<tr><td>2</td><td rowspan="2">Row Span</td><td rowspan="2">Row & Col Span</td></tr>
<tr><td>3</td></tr>
</table>

Array : Dimension 1

	Array : Dimension 2	
1	**ID**	1
	Col 1	Col Span
	Col 2	Col Span
	Col 3	Col Span
2	Array : Dimension 2	
	ID	2
	Col 1	Row Span
	Col 2	Row & Col Span
	Col 3	Row & Col Span
3	Array : Dimension 2	
	ID	3
	Col 1	Row Span
	Col 2	Row & Col Span
	Col 3	Row & Col Span

One final note, once the script has run, you can use $tx->rowCount to get the number of rows in the table, the header row is taken into account. That's about it. I am sure that a lot of the code for this could be improved, so if anyone has any suggestions please let me know. Also, if you have any feedback at all, good or bad, either post a comment below or send me an e-mail. Thanks, I hope you like it.