Search term highlighting

By Cal Henderson

Note: This article continues on from where Searching with PHP and MySQL left off and uses some of the code. If you haven't read it yet, I suggest you do so first.

So you've enabled searching on your site, or you're detecting when users arrive from Google. If would be pretty nifty to highlight the search terms they used, right?

There are a number of scripts out there to do this, but they seem to usually go about it the wrong way and forget to use my favourite regular expression feature - the 'e' (evaluation) flag.

We can't do it too simply - if the content to be highlighted is html, rather than plain text, then we don't want to highlight the contents of the tags. For example, a search for 'hello' shouldn't highlight the following:

```
<img src="hello.jpg" /> Foo!
```

So we only really need to highlight blocks of text outside of tags. A block of text either starts with the beginning of the string, or with a tag. It ends with either the end of the string, or a tag. Let's make a regular expression to match those blocks.

```
$start = '(^|<(?:.*?)>)';
$end   = '($|<(?:.*?)>)';

$text = preg_replace("/$start(.*?)$end/s", "something", $text);
```

Now, we want to highlight some terms in each block of text that matches. Let's use the 'e' flag to delegate this to another function:

```
# from the previous article...

$terms = search_split_terms($search_string);
$terms_rx = search_rx_escape_terms($terms);

$content = search_highlight($content, $terms_rx);


function search_highlight($text, $terms_rx){

 $start = '(^|<(?:.*?)>)';
 $end   = '($|<(?:.*?)>)';

 return preg_replace(
  "/$start(.*?)$end/se",
  "StripSlashes('\\1').".
   "search_highlight_inner(StripSlashes('\\2'), \$terms_rx).".
   "StripSlashes('\\3')",
  $text
 );
}
```

So now each block of text is passed to search_highlight_inner() for processing. Let's write that function.

```
 function search_highlight_inner($text, $terms_rx){

 $colors = search_get_highlight_colors();

 foreach($terms_rx as $term_rx){
  $color = array_shift($colors);

  $text = preg_replace(
    "/($term_rx)/ise",
    "search_highlight_do(StripSlashes('\\1'), \$color)",
    $text
   );
 }

 return $text;
}

function search_get_highlight_colors(){

 return array(
  array('#ffff66','#000000'),
  array('#A0FFFF','#000000'),
  array('#99ff99','#000000'),
  array('#ff9999','#000000'),
  array('#ff66ff','#000000'),
  array('#880000','#ffffff'),
  array('#00aa00','#ffffff'),
  array('#886800','#ffffff'),
  array('#004699','#ffffff'),
  array('#990099','#ffffff'),
 );
}
```

This function loops, once per term, and calls search_highlight_do() for each term, along with a color pair to use for highlighting. The color pairs used in the function above are taken from Google's highlighter. Let's finally write search_highlight_do(), do perform the actual highlighting.

```
 function search_highlight_do($fragment, $color){

 return "<span style=\"background-color: $color[0]; ".
  "color: $color[1]; font-weight: bold;\">".
  "$fragment</span>";
}
```

It very simply takes the passed fragment and wraps a span element around it.

So now we've highlighted our search terms. Excellent! Let's also generate a pretty list of the search terms, complete with highlighting, to show the user which terms are highlight with which colors.

```
 $terms_html = search_html_escape_terms($terms);
```

```
function search_pretty_terms_highlighted($terms_html){

 $colors = search_get_highlight_colors();
 $temp = array();

 foreach($terms_html as $term_html){
  $color = array_shift($colors);
  $temp[] = search_highlight_do($term_html, $color);
 }

 return search_pretty_terms($temp);
}
```

As always, we build on earlier functions to reduce code complexity and room for error.

Now we have a pretty fully functioning search library with term highlighting. What's missing? Part three might just reveal all :)

(end of article)