

How to build a PHP Search Engine

Introduction or a tale of how easy a search engine could work

Five years ago your site may have had five or six pages. However, over the time it has grown and one day you wake up with 2 GB of content, more than a thousand links and no way to sort them by relevance. You need a way for you and your visitors to get information about your services and products faster. A search engine could help you and your site visitors find that precious piece of information on your extensively growing site.

Anyway, let us explain how a search engine works in general. Typically, a search engine works by sending out a 'spider' to fetch as many documents as possible. Then another program, called an indexer, reads these documents and creates an index based on the words contained in each document. Each search engine uses a proprietary algorithm to create its indices such that, ideally, only relevant results are returned for each query.

In the following tutorial we are going to create a PHP search engine with a MySQL database. We are building our search engine on PHP 5.2.3. PHP 5.2.3 includes the latest additions such as embedded SQLite database, support for the MySQL 4.1 client library with the introduction of the mysqli extension, etc.



PHP Search Engine

Why should we use MySQL? Well, this pretty gem will help us organize, fetch and store our data in a quite simple way, so that we will not need to create our own spider and indexer. MySQL would do the hard work for us!

Note: For information on how to install and configure MySQL on Windows you can refer to Gareth Downes-Powell's article <u>"Installing MySQL on a Windows PC"</u>.

For installing PHP 5.x.x on Windows you might want to refer to Allan Kent's Article <u>"Installing PHP 5 under Windows"</u>.

Ok, assuming that you have already installed PHP, MySQL with the appropriate server platform of your choice, IIS or Apache, you have a fully operational web server. Now, let's get into the details.

For the search engine we are going to build in this tutorial, I chose to use a table with 3 columns named ID, title, and article, where I have stored the information the user will be searching for. You can enter some data to your table or even make your own table. The ID field is the primary key for the table, while title and article are fields in which we plan to search. To make it easier for you, I've prepared a query to create the table. You just need to copy and paste the following code into the MySQL interface in a database. In order to do that, access your PHPMyAdmin from your favourite browser, then select your database and table respectively. There you should notice a tab named "SQL". You have to paste the following code there:



Under the "Browse" tab you can see the entries in your table. In order to insert new entries go to "Insert" tab and add new values to the fields Title and Article. Surely, you can extend your table and add additional fields, or change the type of the current fields, in a whole to change the entire table for your own convenience.

How are we going to proceed? We will start creating the search form, then setting up our connection to the database, go through the search function itself, and finally test our search engine. In addition, we are going to design additional features such as paging and displaying the search time.

Beginning of the search page...

Firstly let's create our search page with a form where site visitors will enter keywords to search content. Let's open our favourite web editor and create a new page. Then insert a new form called searchbox and a button with value Search. Finally save the page as Search.php. You may want to use the POST method instead of the GET method for your form since it is more secure. For this tutorial I chose using the GET method since this allows me to check if the search engine is working properly.

As you can see this is just simple code for a form with a text field and a button.

Connecting...

Now let's make the connection string. Open a new PHP document, name it connectionstring.php and save it in the same directory in which Search.php is located. For example, you can name it MySearchEngineWithPHP. Later on you will upload the entire folder to your server. Add the following code to connectionstring.php:



Replace "database", "usrname" and "mypassword" with the respective data that you use for the MySQL database.

Time to search!

The following code is the core of our search engine. We will calculate the search time using a function, make a database connection with another function, and finally we will check for matches with an appropriate algorithm running a query to select a table to look for matches. Add the following code to your Search.php page:

```
<?PHP
function getmicrotime()
    list($usec, $sec) = explode(" ", microtime());
    return ((float)$usec + (float)$sec);
$connection_string = dirname(__FILE__) . "/connectionstring.php";
require_once($connection_string);
mysql_select_db("test") or die ( 'Unable to select database.' );
$RESULTS LIMIT=10;
if(isset($_GET['search_term']) && isset($_GET['search_button']))
      $search_term = $_GET['search_term'];
    if(!isset($first_pos))
        $first_pos = "0";
    $start_search = getmicrotime();
    $sql_query = mysql_query("SELECT * FROM news WHERE MATCH(title,article)
AGAINST('$search_term')");
many mathces (too many matches cause returning of 0 results)
    if($results = mysql_num_rows($sql_query) != 0)
                $sql = "SELECT * FROM news WHERE MATCH(title,article)
AGAINST(\$search_term') LIMIT \$first_pos, \$RESULTS_LIMIT";
                  $sql_result_query = mysql_query($sql);
```



Let's take a closer look at the code. On the second line we initialize function to calculate the time for the search.

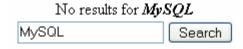
The require_once function initializes a connection to the database – we simply call the connectionstring.php file that we created earlier. With the line "mysql_select_db("test") or die ('Unable to select database.');" we select our database table called test in our case. However, if for some reason a database is not selected the "die" statement will bring a message that a database is not able to be selected. \$RESULTS_LIMIT variable is used to specify how many results to display per page.

Next we use an "if clause" to check if any phrase or word is set in the search field form. This is done with \$search_query. Then we check if our marker for page listing exists with \$first_pos variable – if not we start from the first page. Then we need to initialize a \$QL query – simply with \$sql_query. As you may know mysql_query is a keyword in MySQL. Unfortunately, using that keyword has a major drawback. MySQL does not display results of the phrases that are found in the database too often as it would display too many results. That is why we do a second check. The algorithm counts the number of results with the \$results variable and if it is zero we run second query with a standard field search of the table with LIKE %...%.

NOTE: The function mysql_real_escape_string is a standard MySQL function used to check if the entered value is legal, preventing the injection of malicious MySQL scripts into your database. For example, somebody could enter an SQL query in your form to change your database. Using mysql_real_escape_string will make your search engine and database more secure.

Displaying the results

Now let's display the results. We are going to use a usual loop where we have saved the first results via the \$results variable. We are arranging everything nicely in tables. That way our results will be separated and ordered. If no results are found, the page will display the following message:



However, if results are found, the page should look like this:



Results for *mysql*MySQL MySQL Security 1001 tips for MySQL

OK, so back to Search.php where we have inserted our form. Modify the code as follows:

```
<?PHP
if($results != 0)
?>
  Results for <?PHP echo "<i><b><font
color=#000000>".$search_term."</font></b></i> "; ?>
   Results <b>
    <?PHP echo ($first_pos+1)." - ";
    if(($RESULTS_LIMIT + $first_pos) < $results) echo ($RESULTS_LIMIT + $first_pos);</pre>
    else echo $results ; ?>
   </b>
    out of <b><?PHP echo $results; ?></b>
    for(<b><?PHP echo sprintf("%01.2f", $time_search); ?></b>)
    seconds 
 <form action="" method="GET">
     <input name="search_term" type="text"
value="<?PHP echo $search term; ?>" size="40">
      <input name="search_button" type="submit" value="Search"> 
   </form>
 <?PHP
   while($row = mysql_fetch_array($sql_result_query))
   ?>
    <?PHP echo $row['title']; ?>
    <?PHP
   }
   ?>
<?PHP
elseif($sql_query)
?>
No results for
                                <?PHP echo "<i><book

color=#000000>".$search_term."</font></b></i> "; ?>
```



Sorting into pages

Our final step is to show the number of result pages. We are calculating how many results the SQL query has returned. We start by calculating the first position in order to place it in relevance. If the maximum number of results is exceeded the algorithm creates a forward button for the user to switch to the next page. We already have defined the maximum number per page via \$RESULTS_LIMIT. If the user goes to the second page of the results he can go back and check previous results with a back button. Well, these so called buttons are not really buttons – they are simple ">>" and "<<" signs. Of course feel free to add your own buttons, or even some flash images.

I am also using a PHP function stripslashes in order to escape some quotes. Why? I use it to clean the results from the quotes in the MySQL query. That way the algorithm will not fail if any quotes are inserted.

We can do all this with the following code:

```
<?PHP
if($first_pos > 0)
  $back=$first_pos-$RESULTS_LIMIT;
  if(\$back < 0)
    back = 0;
  echo "<a href='search.php?search_term=".stripslashes($search_term)."&first_pos=$back'
></a>";
if($results>$RESULTS_LIMIT)
  $sites=intval($results/$RESULTS_LIMIT);
  if($results%$RESULTS_LIMIT)
    $sites++;
for ($i=1;$i<=$sites;$i++)
  $fwd=($i-1)*$RESULTS_LIMIT;
  if($fwd == $first_pos)
      echo "<a
href='search.php?search_term=".stripslashes($search_term)."&first_pos=$fwd
`><b>$i</b></a>
```



```
else
{
    echo "<a
href='search.php?search_term=".stripslashes($search_term)."&first_pos=$fwd '>$i</a> |
    ";
    ";
    }
}
if(isset($first_pos) && $first_pos < $results-$RESULTS_LIMIT)
{
    $fwd=$first_pos+$RESULTS_LIMIT;
    echo "<a href='search.php?search_term=".stripslashes($search_term)."&first_pos=$fwd '
> >></a>";
    $fwd=$results-$RESULTS_LIMIT;
}
?>
```

This is the final piece of code in this article.

The next code block contains the entire code for the search engine. I've inserted comments so that you can track the process:

```
<?PHP
function getmicrotime()
    list($usec, $sec) = explode(" ", microtime());
    return ((float)$usec + (float)$sec);
//initializing connection to the database
$connection_string = dirname(__FILE__) . "/connectionstring.php";
require once($connection string);
//selecting table
mysql_select_db("test") or die ( 'Unable to select database.' );
//max number of results on the page
$RESULTS LIMIT=10;
if(isset($_GET['search_term']) && isset($_GET['search_button']))
      $search_term = $_GET['search_term'];
    if(!isset($first_pos))
    {
        $first_pos = "0";
    $start_search = getmicrotime();
      //initializing MySQL Quary
    $sql_query = mysql_query("SELECT * FROM news WHERE MATCH(title,article)
AGAINST('$search_term')");
    //additional check. Insurance method to re-search the database again in case of too
many matches (too many matches cause returning of 0 results)
    if($results = mysql_num_rows($sql_query) != 0)
                $sql = "SELECT * FROM news WHERE MATCH(title,article)
AGAINST(`$search_term') LIMIT $first_pos, $RESULTS_LIMIT";
                  $sql_result_query = mysql_query($sql);
```



```
else
               $sql = "SELECT * FROM news WHERE (title LIKE
`%".mysql_real_escape_string($search_term)."%' OR article LIKE `%".$search_term."%') ``;
               $sql_query = mysql_query($sql);
               $results = mysql_num_rows($sql_query);
               $sql_result_query = mysql_query("SELECT * FROM news WHERE (title LIKE
`%".$search_term."%' OR article LIKE `%".$search_term."%') LIMIT $first_pos,
$RESULTS_LIMIT ");
   $stop_search = getmicrotime();
     //calculating the search time
   $time search = ($stop search - $start search);
?>
<?PHP
if($results != 0)
?>
  <!-- Displaying of the results -->
Results for <?PHP echo "<i><b><font
color=#000000>".$search_term."</font></b></i> "; ?>
   Results <b>
     <?PHP echo ($first_pos+1)." - ";
     if(($RESULTS_LIMIT + $first_pos) < $results) echo ($RESULTS_LIMIT + $first_pos);</pre>
    else echo $results ; ?>
    out of <b><?PHP echo $results; ?></b>
    for(<b><?PHP echo sprintf("%01.2f", $time_search); ?></b>)
     seconds 
 <form action="" method="GET">
      <input name="search_term" type="text"</pre>
value="<?PHP echo $search_term; ?>" size="40">
      </form>
 <?PHP
   while($row = mysql_fetch_array($sql_result_query))
   {
   ?>
     <?PHP echo $row['title']; ?>
     <?PHP
   ?>
<?PHP
//if nothing is found then displays a form and a message that there are nor results for
the specified term
elseif($sql_query)
```



```
?>
No results for <?PHP echo "<i><b><font
color=#000000>".$search_term."</font></b></i> "; ?>
   <form action="" method="GET">
      <input name="search_term" type="text" value="<?PHP echo $search_term; ?>">
         <input name="search_button" type="submit" value="Search">
      </form>
   <?PHP
?>
<?php
    if (!isset($_GET['search_term'])) { ?>
   <form action="" method="GET">
      <input name="search_term" type="text" value="<?PHP echo $search_term; ?>">
         <input name="search_button" type="submit" value="Search">
      </form>
   <?php
    ?>
 <?PHP
//displaying the number of pages where the results are sittuated
if($first_pos > 0)
 $back=$first_pos-$RESULTS_LIMIT;
 if(\$back < 0)
   back = 0;
 echo "<a href='search.php?search_term=".stripslashes($search_term)."&first_pos=$back'
></a>";
if($results>$RESULTS_LIMIT)
 $sites=intval($results/$RESULTS_LIMIT);
 if($results%$RESULTS_LIMIT)
   $sites++;
for ($i=1;$i<=$sites;$i++)
 $fwd=($i-1)*$RESULTS_LIMIT;
 if($fwd == $first_pos)
```



```
echo "<a
href='search.php?search term=".stripslashes($search term)."&first pos=$fwd
'><b>$i</b></a> ";
 else
      echo "<a
href='search.php?search_term=".stripslashes($search_term)."&first_pos=$fwd '>$i</a>
";
if(isset($first_pos) && $first_pos < $results-$RESULTS_LIMIT)</pre>
  $fwd=$first pos+$RESULTS LIMIT;
  echo "<a href='search.php?search_term=".stripslashes($search_term)."&first_pos=$fwd '
 >></a>";
  $fwd=$results-$RESULTS_LIMIT;
?>
```

Do not forget to upload the files connectionstring.php and Search.php in the same directory on your server.

Preview your page and test it. Just open it in your favourite web browser and type in some words. Make sure that your table is filled with some of these words otherwise it will return you that there are no results found.

Conclusion

That's all we need. Our PHP search engine is running smoothly, fast and accurate! We might add some JavaScript functions from an external file, as well some brief CSS rules to make it fancier and give it a pleasant look. In addition, you may want to try combining it with one of our extensions – Ajax Autocomplete. Once you bind the extension to the input form in the code we have just created, it will add auto-complete functionality to the PHP Search Engine. When a user starts entering a phrase a list of suggested search terms will be displayed. Very nice, right?