Warning: extract() [function.extract]: First argument should be an array in **/home/servers/www.devshed.com-mambo/www/includes/templating.php** on line **34**

MYSQL RSS

Completing a Search Engine with MySQL and PHP 5

By: Alejandro Gervasio
Rating: ★★★☆☆ / 10
2007-08-13

Sun Microsystems previews Version 5.4 of its MySQL database April 21 at the MySQL Conference & Expo in Santa Clara, Calif. The announcement of MySQL 5.4 comes 24 hours after Oracle announced plans to buy Sun for $7.4 billion. Read More!

- MySQL Security Tips (2009-11-05)
- Mastering WHILE Loops for PHP and MySQL (2009-10-26)
- Using the Yahoo Site Explorer Inbound Links API (2009-11-11)
- SEO Essentials: the Proper Web Server and Platform (2009-10-27)
- Building a MySQL Abstraction Class with Method Chaining (2009-10-29)
- What is Mobile SEO? (2009-11-13)
- Using Lynx for SEO Analysis (2009-11-10)
- Mastering Lynx (Open Source Text Browser) for Search Engine Optimization (2009-11-09)

Building database-driven web sites is one of the most popular trends today in web site development. However, this approach implies that potential visitors must be provided with a straightforward mechanism that allows them to search through web site content. This three-part series walks you through the process of building an expandable search engine by using the combined functionality of MySQL and PHP 5.

Introduction

As you'll possibly recall, over the course of the preceding article of the series I defined a couple of basic MySQL-processing classes. These classes came in handy not only for performing different search queries against one or more selected databases, but for displaying the corresponding results using chunks of paginated data. I used the object-oriented paradigm to develop the search engine, but this isn't mandatory since the same level of functionality can also be achieved by utilizing a procedural approach.

Having clarified that crucial point, aside from refreshing some important concepts concerning the development of this MySQL-based search engine, I think now is a good time to discuss the topics that I plan to address in this last installment of the series. This way you'll know what to expect from this tutorial.

In the previous article I showed you how to spawn the results returned by a specific search query across different web pages, so in this last part of the series I'm going to complete this search application by fixing a concrete issue associated specifically with paginating MySQL data sets. Put in a simple way, each time that a user performs a search and then clicks on the paginated links to browse the corresponding results, the original search string passed to the application is simply no longer available in subsequent web page requests. This is something that can quickly break the functionality of the search engine.

Of course, there are many ways to solve this problem, such as appending the search terms to the query string generated by the corresponding paginated links. However, in this case I'm going to take a slightly different approach, and use a simple session mechanism to maintain the value of the search string across different web pages. Naturally, you're free to use the method that best suits your personal needs.

All right, with the preliminaries already out of our way, let's take the last step involved in building this MySQL-based search engine. Let's get started!

As usual with many of my articles on PHP web development, before I proceed to implement the session mechanism for maintaining the value of a given search string across different web pages, I'd like to list all of the source files that comprise this search application as they were originally defined in the previous article of the series.

That being said, here are the signatures for the source files:

(definition of "form.htm" file)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN'
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
<title>MySQL-based Search Engine</title>
<link href="default.css" rel="stylesheet" type="text/css'
media="screen" />
<script language="javascript" type="text/javascript">
window.onload=function(){
   if(document.getElementById && document.getElementsByTagName &&
document.createElement){
     var sfield=document.getElementsByTagName('form')[0].elements
[0];
     if(!sfield){return};
     sfield.onfocus=function(){this.value=''};
     sfield.onblur=function(){
       if(!this.value){this.value='Enter your search term here'};
     }
   }
}
</script>
</head>
<body>
  <h1>MySQL-based Search Engine</h1>
  <div class="maincontainer">
    <form method="get" action="processform.php">
      <input type="text" name="searchterm" title="Enter your
search term here" value="Enter your search term here'
class="searchbox" />
      <input type="submit" name="search" title="Search Now!
"value="Search" class="searchbutton" />
    </form>
  </div>
</body>
</html>
```

(definition of "default.css" file)

```
body{
   background: #ccc;
   margin: 0;
   padding: 0;
}

h1{
   width: 375px;
   padding: 10px;
   margin-left: auto;
   margin-right: auto;
   background: #339;
   font: normal 18px Arial, Helvetica, sans-serif;
   color: #fff;
   border: 1px solid #000;
   text-align: center;
}

h2{
   font: bold 18px  Arial, Helvetica, sans-serif;
   color: #339;
}

p{
   font: normal 10pt Arial, Helvetica, sans-serif;
   color: #000;
}

a:link,a:visited{
   font: normal 10pt Arial, Helvetica, sans-serif;
   color: #00f;
   text-decoration: none;
}

a:hover{
   color: #f00;
   text-decoration: underline;
}

.maincontainer{
   width: 375px;
   padding: 10px;
   margin-left: auto;
   margin-right: auto;
   background: #f0f0f0;
   border: 1px solid #000;
}

.rowcontainer{
   padding: 10px;
   margin-bottom: 10px;
   background: #ccf;
}

.searchbox{
   width: 200px;
   font: normal 12px Arial, Helvetica, sans-serif;
   color: #000;
}

.searchbutton{
   width: 80px;
   font: bold 12px Arial, Helvetica, sans-serif;
   color: #000;
}
```

(definition of "mysql.php" file)

```
<?php
// define 'MySQL' class
class MySQL{
   private $conId;
   private $host;
   private $user;
   private $password;
   private $database;
   private $result;
   const OPTIONS=4;
   public function __construct($options=array()){
     if(count($options)!=self::OPTIONS){
       throw new Exception('Invalid number of connection
parameters');
     }
     foreach($options as $parameter=>$value){
       if(!$value){
         throw new Exception('Invalid parameter '.$parameter);
       }
       $this->{$parameter}=$value;
     }
     $this->connectDB();
   }
   // connect to MySQL
   private function connectDB(){
     if(!$this->conId=mysql_connect($this->host,$this-
>user,$this->password)){
       throw new Exception('Error connecting to the server');
     }
     if(!mysql_select_db($this->database,$this->conId)){
       throw new Exception('Error selecting database');
     }
   }
   // run query
   public function query($query){
     if(!$this->result=mysql_query($query,$this->conId)){
       throw new Exception('Error performing query '.$query);
     }
     return new Result($this,$this->result,$query);
   }
   public function escapeString($value){
     return mysql_escape_string($value);
   }
}
// define 'Result' class
class Result {
   private $mysql;
   private $result;
   private $query;
   private $rowTemplate='default.tpl';
   private $numRecs=4;
   public function __construct($mysql,$result,$query){
     $this->mysql=$mysql;
     $this->result=$result;
     $this->query=$query;
   }
   // fetch row
   public function fetchRow(){
     return mysql_fetch_assoc($this->result);
   }
   // count rows
```

```
    public function countRows(){
      if(!$rows=mysql_num_rows($this->result)){
        return false;
      }
      return $rows;
    }
    // count affected rows
    public function countAffectedRows(){
      if(!$rows=mysql_affected_rows($this->mysql->conId)){
        throw new Exception('Error counting affected rows');
      }
      return $rows;
    }
    // get ID form last-inserted row
    public function getInsertID(){
      if(!$id=mysql_insert_id($this->mysql->conId)){
        throw new Exception('Error getting ID');
      }
      return $id;
    }
    // seek row
    public function seekRow($row=0){
      if(!is_int($row)||$row<0){
        throw new Exception('Invalid result set offset');
      }
      if(!mysql_data_seek($this->result,$row)){
        throw new Exception('Error seeking data');
      }
    }
    public function countFields(){
      if(!$fields=mysql_num_fields($this->result)){
        throw new Exception('Error counting fields.');
      }
      return $fields;
    }
    public function fetchPagedRows($page){
      $numPages=ceil($this->countRows()/$this->numRecs);
      if(empty($page)||$page>$numPages){
        $page=1;
      }
      $result=$this->mysql->query($this->query.' LIMIT '.($page-1)
*$this->numRecs.','.$this->numRecs);
      $output='';
      while($row=$result->fetchRow()){
        $rowTemplate=file_get_contents($this->rowTemplate);
        foreach($row as $key=>$value){
          $rowTemplate=str_replace('{'.$key.'}',$value,$rowTemplate);
        }
        $output.=$rowTemplate;
      }
      $output.='<p>';
      if($page>1){
        $output.='<a href="'.$_SERVER['PHP_SELF'].'?&page='.
($page-1).'">&lt;&lt;</a> ';
      }
      for($i=1;$i<=$numPages;$i++){
        $output.=$i!=$page?'<a href="'.$_SERVER['PHP_SELF'].'?
page='.$i.'">'.$i.'</a> ':$i.' ';
      }
      if($page<$numPages){
        $output.=' <a href="'.$_SERVER['PHP_SELF'].'?&page='.
($page+1).'">&gt;&gt;</a>';
      }
```

```
      $output.='</p>';
      return $output;
    }
}
?>
```

(definition of 'default.tpl' file)

```
<div class="rowcontainer">
  <p><strong>First Name:</strong> {firstname}</p>
  <p><strong>Last Name:</strong> {lastname}</p>
  <p><strong>Comments:</strong> {comments}</p>
</div>
```

(definition of "sessionhandler.php" file)

```
<?php
class SessionHandler{
   public function __construct(){
     session_start();
   }
   public function setVariable($value='default',$varname='default'){
     $_SESSION[$varname]=$value;
   }
   public function getVariable($varname='default'){
     if(!$_SESSION[$varname]){
       return false;
     }
     return $_SESSION[$varname];
   }
   public function destroy(){
     session_start();
     session_unset();
     session_destroy();
   }
}
?>
```

As you can see, all the above source files perform well-differentiated tasks, in this way implementing the distinct application modules that comprise the search engine. These range from displaying a simple web form for entering diverse search terms to executing the search queries against one or more MySQL databases.

Besides, you should notice that I defined a template file called "default.tpl" which is used by the previous "Result" PHP class to format the results returned by a query. This record formatting process can be done directly from inside the class.

So far, so good right? At this stage I have shown you the complete signatures corresponding to all the source files that make up this MySQL-based search engine. So what is the next step?

Well, considering that all the database results returned by a specific query must be displayed on a separate web page, in the following section I'm going to define yet another PHP class. It will be tasked with building basic web documents, which makes it quite useful in conjunction with the rest of the source files that you saw earlier.

Want to see how this brand new PHP class will be built? Jump ahead and read the next few lines.

As I stated in the prior section, the last task required to complete the development of this search application consists of building a simple yet effective web page generating class. This class will be responsible for creating the web documents required to display the results returned by a specified search query.

Of course, the definition of this brand new PHP class is entirely optional in this case, since the respective database results can be also shown by using a procedural approach instead of the object-oriented one that I'm explaining here. Nevertheless, I'd like to show you the signature for this class, regardless of the method that you may want to use for displaying the search results.

Given that, here's how this web page generating class looks:

```
<?php
class WebPage{
```

```
    private $title;
    public function __construct($title='MySQL-based Search
Engine'){
      $this->title=$title;
    }
    public function displayHeader(){
      return '<html><head><title>'.$this->title.'</title><link
href="default.css" rel="stylesheet" type="text/css" /></head>';
    }
    public function displayBody($content){
      return '<body>'.$content.'</body>';
    }
    public function displayFooter(){
      return '</html>';
    }
}
?>
```

Quite simple, right? As you can see, the signature for the above "WebPage" class doesn't bear too much discussion. It performs a few simple tasks, such as building the header, body and footer sections of a typical web document. Therefore, assuming that you won't have major problems understanding the way that this class works, it's time to move forward and develop a hands-on example, where you'll be able to see how this MySQL-driven search engine does its business.

To learn how this practical example will be created, please click on the link below and keep reading.

In consonance with the concepts deployed in the previous section, I'm going to set up an example where all the PHP classes defined earlier will be used in conjunction to make this search engine work as expected.

In this case, I'm going to use a simple "USERS" MySQL database table, populated with some basic records to illustrate the functionality of the search engine in question. It's quite possible, however, that in a real situation you'll need to perform the pertinent search queries against multiple tables.

Okay, assuming that the aforementioned "USERS" table is the same one that I used in the first article of the series, it is filled in with the following data:

Id	firstname	lastname	email	comments
1	Alejandro	Gervasio	alejandro@domain.com	MySQL is great for building a search engine
2	John	Williams	john@domain.com	PHP is a server side scripting language
3	Susan	Norton	sue@domain.com	JavaScript is good to manipulate documents
4	Julie	Wilson	julie@domain.com	MySQL is the best open source database server

Here's a basic example that demonstrates how this extensible search application functions:

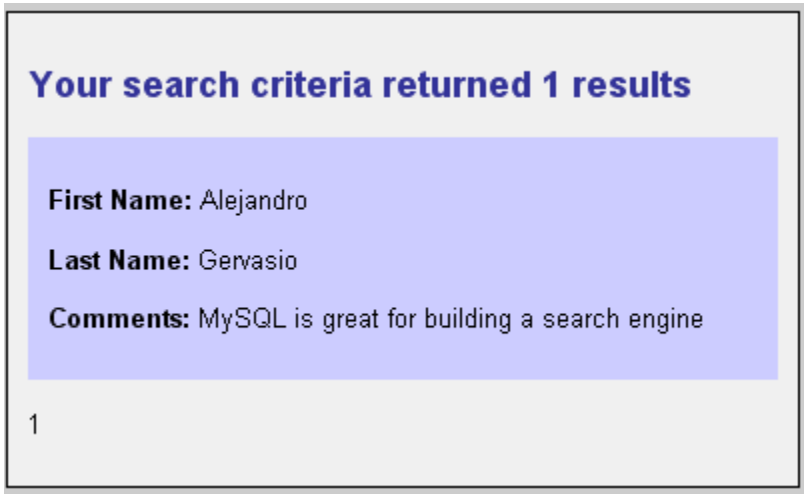
```
try{
    // include classes
    require_once 'sessionhandler.php';
    require_once 'mysql.php';
    require_once 'webpage.php';
    // create new session handler object
    $sh=new SessionHandler();
    // connect to MySQL
    $db=new MySQL(array('host'=>'host','user'=>'user',
```

```
'password'=>'password','database'=>'database'));
   // create new web page object
   $wp=new WebPage();
   // check if search term has been saved to session variable
   if(!$sh->getVariable('searchterm')){
     $searchterm=$db->escapeString($_GET['searchterm']);
     $sh->setVariable($_GET['searchterm'],'searchterm');
   }
   else{
     // get search term from session variable
     $searchterm=$sh->getVariable('searchterm');
   }
   // display header
   echo $wp->displayHeader();
   $result=$db->query("SELECT firstname, lastname,comments FROM
users WHERE MATCH(firstname,lastname,comments) AGAINST
('$searchterm' IN BOOLEAN MODE)");
   if(!$result->countRows()){
     echo $wp->displayBody('<div class="maincontainer"><h2>No
results were found. Go back and try a new search.</h2></div>');
     }
   else{
     // display search results
     echo $wp->displayBody('<div class="maincontainer"><h2>Your
search criteria returned '.$result->countRows().'
results</h2>'.$result->fetchPagedRows($_GET['page']).'</div>');
   }
   // display footer
   echo $wp->displayFooter();
}
catch(Exception $e){
   echo $e->getMessage();
   exit();
}
```

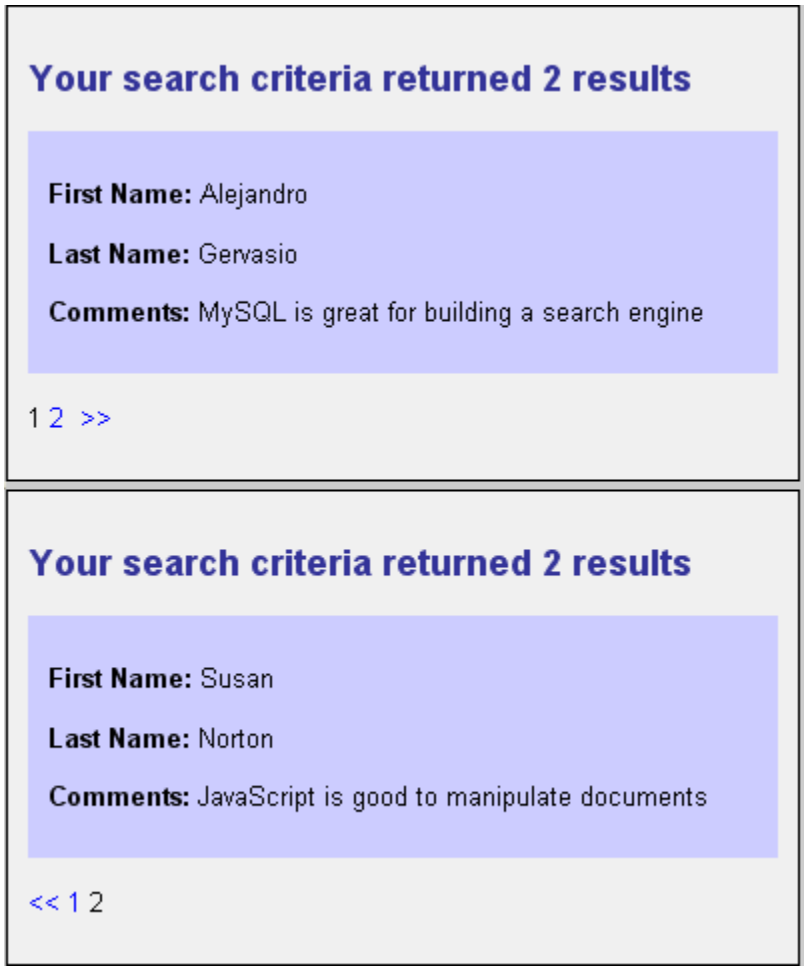
As shown above, putting the search engine to work is a fairly easy process. It requires only using all the classes that were previously defined, in addition to performing the respective search queries by taking advantage of the full-text and Boolean capabilities offered by MySQL.

Finally, take a look at the following screen shots. They show the different database results returned by the search engine, according to certain search strings entered in the corresponding web form:

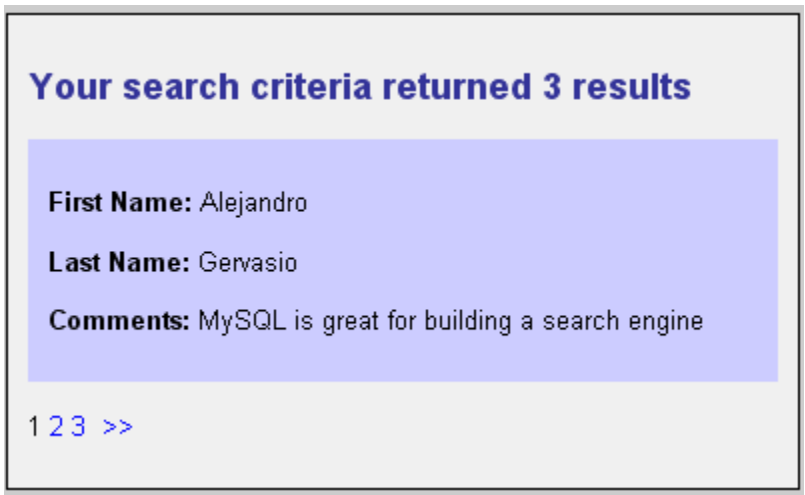
(results returned by entering the search string "Alejandro")



(results returned by entering the search string "Alejandro+Susan")



(results returned by entering the search string "Alejandro+Susan+John")



As you can see in the previous images, building a search engine using the powerful MySQL/PHP 5 combination is indeed a no-brainer process that can be tackled with minor hassles. At this point, you have at your disposal all the required source files to incorporate this application into your own web site, and provide users with a simple mechanism to search and find your nicely-crafted contents.

Final thoughts

Unfortunately, this is the end of the series. Nonetheless, I think that the whole experience has been educational, since it illustrated in a friendly fashion how to build an expandable search engine by using the capabilities provided by MySQL and PHP 5.

See you in the next PHP tutorial!