

Creating a search feature with PHP & MySQL

Developing a robust, interactive and engaging Web site involves many different avenues, such as interactive pop-out menu's using dynamic JavaScript, Cascading Style Sheets (CSS), complex maps that allows visitors to rollover individual sections for detailed information, forms designed and formatted with CSS and are programmed to collect and send visitor feedback to a specified recipient. Other features could include: database driven pages that display a current member directory, a customized blog section that enables administrators to manage postings and allow random users in coordination with CAPTCHA techniques to post remarks to an article. Arguably, one of the most popular features of any database driven site is a searchable form feature that allows anyone to search for current staff members of an organization and find additional information, such as their email address or phone number.

In this article, you'll learn how to create a searchable form feature that will query a database table and display current staff member information. During the analysis you'll learn how to do the following: create a database table that will hold current staff listings, create a search form and use PHP, in coordination with Structured Query Language (SQL) to capture information entered by the visitor and append the information to display the results we want to show. Additionally, you'll learn about design patterns with the searchable form including security through code to ensure that only certain input is entered before performing operations on your database table and ways to pass state information through the address bar to display additional information about staff members.

Download Project Files

<http://midwestwebdesign.net/tutorials/search/search.zip>

*Note: The numbers in brackets indicate a web address that is given at the end of this article.

What you'll need

In order to complete this article, you'll need the following tools and technologies:

One My SQL database

PHP support

A text editor

Creating our database

If you're unsure how to create a database through your host, please contact your system administrator for further information. Once you've created a database, you'll need a way to connect to it, create a table, and enter data for appropriate entities. One popular database administration tool for My SQL is PHP My Admin [1], or one that I'm particular fond of is SQLyog [2], which currently supports a free version (Community Edition) which is sufficient for the purposes of this article.

Creating our table

Our table needs to be created with the following columns and their corresponding data types and lengths:

Column Name	Data Type	Length	Null or Not Null	Primary key?	Auto Increment
ID	INT	1	Not Null	Yes	Yes
FirstName	Varchar	50	Not Null	NO	NO
LastName	Varchar	50	Not Null	NO	NO
Email	Varchar	50	Not Null	NO	NO
PhoneNumber	Varchar	15	Not Null	NO	NO

While an in-depth analysis of database design is beyond the scope of this article, let's briefly discuss the concepts of columns, data types and lengths. A database table is composed of columns and rows, much like an Excel spread sheet. A column gives us a unique way to identify a certain entity, such as a first name. Data types are merely what type of data the column stores, such as a number, variable characters, date, time, etc. Lengths allocate a specified amount of memory (storage) to the table column. In our case, we're using variable characters that allow more flexibility. In other words, if the first or last name doesn't occupy exactly 50 units of data in our column, only that which is occupied will be allocated and stored. Furthermore, there should be no null (empty) entities for any staff members. Lastly, it should be noted that the reason our first row is highlighted in yellow is to identify that the ID column is our primary key. A primary key in database design ensures that each record is guaranteed to be unique. This column is also set to auto increment, ensuring that each record has a unique number assigned to it by the database engine.

Enter staff members into the table

Once you have created the columns for our database table, proceed to fill our columns with data. Six records should suffice for the purposes of this article. A replica of mine is below:

ID	FirstName	LastName	Email	PhoneNumber
----	-----------	----------	-------	-------------

2	Ryan	Butler	ryanbutler@domain.com	417-854-8547
3	Brent	Callahan	brentcallahan@domain.com	417-854-6587

Create the form

In order to create the search form, you'll need to open a text editor of your choice. One that is free is PSPad [3]. You can use any text editor you want; though it's recommended to choose one that has syntax highlighting to make PHP debugging and programming a bit easier. When creating the search form page, make sure to save it with a .php extension, otherwise, the PHP code will not execute. Once the form is saved, begin with the following markup:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Search Contacts</title>
</head>

<body>
<h3>Search Contacts Details</h3>
<p>You may search either by first or last name</p>
<form method="post" action="search.php?go" id="searchform">
<input type="text" name="name">
<input type="submit" name="submit" value="Search">
</form>
</body>
</html>
```

If you're familiar with HTML, everything should look familiar until you reach the opening form tag. Inside the opening form tag, the key line of code is the action attribute. We set the action of the form to the name of our file and then append on a query string of "go". We'll discuss why this is important in the next section (search_start.php).

http://midwestwebdesign.net/tutorials/search/search_start.php

Check to see if criteria's have been met

At this point, it's wise to understand how the initial search functionality will work. When a visitor enters a first or last name and then presses the submit button, the form will post back to itself and append a query string of "go" on the end. At this point, we'll check the address bar for a query string of go and if it exists, we'll perform additional programming logic to make applicable staff members to appear. Before we perform database operations or information is displayed back to the visitor, we need to verify three things: (1) Has the form been submitted, (2) Does the query string contain a value of go, and (3) Does the search

criteria entered only contain a capital or lower case letter? If none of these is true, then no operations are needed. First, let's add a PHP code block right below the closing `</form>` tag:

```
</form>
<?php
//do something here in code
?>
</body>
</html>
```

First, we start a PHP code block:

```
<?php
```

Followed by a closing PHP code block:

```
?>
```

Any PHP code inside this area will be executed by the server. Moving on, we check to see if the form has been submitted:

```
<?php
if(isset($_POST['submit'])){
//do something here in code
}
else{
echo "<p>Please enter a search query</p>";
}
?>
```

We use the built-in function `isset`, which is a Boolean check and pass in the super global array `$_POST` value of `submit`. A Boolean value in programming is a true/false value. Therefore, if the function returns true, then the form has been submitted and we need to do additional programming logic. If the function returns false, we display an error message. If you haven't saved your file yet, now would be a good time to do so.

http://midwestwebdesign.net/tutorials/search/search_submit.php

Next, we need to check whether the query string has a value of `go`, which is achieved by using this code:

```
<?php
if(isset($_POST['submit'])){
if(isset($_GET['go'])){
```

```

else{
echo "<p>Please enter a search query</p>";
}
}
}
?>

```

We simply nest another conditional logic statement inside our original, except this time, we use the super global array name of `$_GET`, with a parameter of "go" to check our address bar for this value. Save your file.

http://midwestwebdesign.net/tutorials/search/search_go.php

Finally, we need to ensure that visitors are only allowed to enter a capital or lower case letter as the first character in our search field. Why this is done is explained in the SQL injection section at the end of the article. We also need a way to collect the search criteria entered by the visitor. The best way to check visitor input, and really the only way, is through a regular expression, which is shown below:

```

<?php
if(isset($_POST['submit'])){
if(isset($_GET['go'])){
if(preg_match("/^[A-Za-z]+/", $_POST['name'])){
$name=$_POST['name'];
}
}
}
else{
echo "<p>Please enter a search query</p>";
}
}
?>

```

Again, as before, we nest another conditional logic statement inside our original two. This time, we use a regular expression to check our input. We use the built-in `preg_match` function along with two parameters, which is the pattern to match, in our case, at least one capital or lowercase letter, followed by the name of our form field we want to check against, in our case, name. Lastly, in order to collect the search criteria entered by the visitor, we create a variable called name by using the dollar sign (\$) in PHP and assign it the POST value of name from our form, which is used in conjunction with our SQL query, seen later. At this point, even though our form won't return a result set, we've ensured that: (1) The form has been submitted, (2) The query string has a value of go, and (3) The visitor has entered a capital or lowercase letter as the first character before any database or SQL operations are allowed to continue. Save your file.

http://midwestwebdesign.net/tutorials/search/search_expression.php

Connect, select, query and return result set from our database table

In order to return a result set from our database table, we need to first connect to our database server. We use the following code:


```

$db=mysql_connect ("mysqlb11.webcontrolcenter.com", "<username>", "<password>") or die ('I cannot
connect to the database because: ' . mysql_error());

//select the database to use
$mydb=mysql_select_db("midwestwebdesign");

}
else{
echo "<p>Please enter a search query</p>";
}
}
}
}
?>

```

We create a variable called mydb and assign to My SQL's built-in function, mysql_select_db and pass in the name of our database we created earlier. Next, we query our database table using SQL, along with our name variable, which contains the search criteria entered by our visitor:

```

<?php
if(isset($_POST['submit'])){
if(isset($_GET['go'])){
if(preg_match("/[A-Z | a-z]+/", $_POST['name'])){
$name=$_POST['name'];

//connect to the database
$db=mysql_connect ("mysqlb11.webcontrolcenter.com", "<username>", "<password>") or die ('I cannot
connect to the database because: ' . mysql_error());

//select the database to use
$mydb=mysql_select_db("midwestwebdesign");

//query the database table
$sql="SELECT ID, FirstName, LastName FROM Contacts WHERE FirstName LIKE '%' . $name . '%"
OR LastName LIKE '%' . $name . '%"";
}
else{
echo "<p>Please enter a search query</p>";
}
}
}
}
?>

```

When querying our database table, we first create a variable called sql, and assign it to a literal string containing our SQL query. In our case, we use the keyword SELECT to grab the id, first and last name columns from our contacts table. Then we use the keyword WHERE, along with our first and last name columns to narrow our search field. Using the LIKE keyword, we pass in the percentage sign (%), which is a wildcard character that returns zero or more characters and our name variable from the search field.

As a result, the LIKE keyword (in conjunction with our wildcard character) will find any name that matches in our database table to that entered by our visitor. In other words, the SQL query performed could be translated as follows: "Select first and last names from our contacts table where either the first or last name matches that entered by our visitor." If you haven't saved your file yet, now would be a good time to do so.

http://midwestwebdesign.net/tutorials/search/search_query.php

Next, we need to store the results of our SQL query in a variable and run it against the **mysql_query** function as illustrated below:

```
<?php

if(isset($_POST['submit'])) {
if(isset($_GET['go'])) {
if(preg_match("/[A-Z | a-z]+/", $_POST['name'])) {
$name=$_POST['name'];

//connect to the database
$db=mysql_connect ("mysqlb11.webcontrolcenter.com", "<username>", "<password>") or die ('I cannot
connect to the database because: ' . mysql_error());

//select the database to use
$mydb=mysql_select_db("midwestwebdesign");

//query the database table
$sql="SELECT ID, FirstName, LastName FROM Contacts WHERE FirstName LIKE '%" . $name . "%'
OR LastName LIKE '%" . $name . "%";

//run the query against the mysql query function
$result=mysql_query($sql);
}
else {
echo "<p>Please enter a search query</p>";
}
}
}
?>
```

As you can see, we create a variable called result, and assign it to the mysql_query function, passing in our query variable. Now our query is stored in the result variable. In order to display our result set in PHP, we'll create a loop and then parse out the first and last name using an unordered list as shown below:

```
<?php
```

```

if(isset($_POST['submit'])){
if(isset($_GET['go'])){
if(preg_match("/^[ a-zA-Z]+/", $_POST['name'])){
$name=$_POST['name'];

//connect to the database
$db=mysql_connect ("mysqlb11.webcontrolcenter.com", "<username>", "<password>") or die ('I cannot
connect to the database because: ' . mysql_error());

//select the database to use
$mydb=mysql_select_db("midwestwebdesign");

//query the database table
$sql="SELECT ID, FirstName, LastName FROM Contacts WHERE FirstName LIKE '%" . $name . "%'
OR LastName LIKE '%" . $name . "%";

//run the query against the mysql query function
$result=mysql_query($sql);

//create while loop and loop through result set
while($row=mysql_fetch_array($result)){

    $FirstName =$row['FirstName'];
    $LastName=$row['LastName'];
    $ID=$row['ID'];

//display the result of the array

echo "<ul>\n";
echo "<li>" . "<a href='\"search.php?id=$ID\">\" . $FirstName . " " . $LastName . "</a></li>\n";
echo "</ul>";
}
}
else{
echo "<p>Please enter a search query</p>";
}
}
}
?>

```

First, we create a while loop, and inside we create a variable called **row** and assign it to the **mysql_fetch_array** function, which takes one parameter, our result variable containing our SQL query. Inside the while loop, we assign each column from the row variable to a new variable with an identical name. Then, we display the values inside an unordered list. Two things worth noting are: (1) inside the while loop, you don't have to create and assign a variable to the row array, you could parse out the values directly from the row array, (2) you'll notice in the anchor tag that we pass in the name of our file along with the id or primary key. The reason for this is mainly because in most search applications, you don't display everything initially. Since we're only displaying first and last name, by appending the ID on the

end of our anchor tag, we can then use the ID for an additional query that will display further information regarding our staff member. Save your file and test your form at this point.

http://midwestwebdesign.net/tutorials/search/search_display.php

Removing bullets

As you can see from viewing the example file, our matches are coming back in an unordered list, but bullets in this case aren't needed. To remove the bullets, add this CSS rule to the top of your file, inside the opening and closing <head> tags:

```
<style type="text/css" media="screen">
ul li {
  list-style-type:none;
}
</style>
```

This rule simply says to target all list items inside of an unordered list and set their bullet type to none.

Searching by letter

Since searching by letter is merely adjusting a couple lines of code, let's go ahead and add in functionality that allows our visitor to search our staff member directory for first or last names containing a certain letter. In order to do this, add this code right after the closing <form> tag:

```
</form>
<p><a href="?by=A">A</a> | <a href="?by=B">B</a> | <a href="?by=K">K</a></p>
<?php
```

We simply hard-code anchor tags with a query string of by, and set it equal to a particular letter. Depending on how large your database of staff members is you may want to add additional ones. To achieve the search functionality by letter we need to add the following code right after the closing curly brace of our original script as shown below:

```
}//end of search form script

if(isset($_GET['by'])){
$letter=$_GET['by'];

//connect to the database
$db=mysql_connect ("mysqlb11.webcontrolcenter.com", "<username>", "<password>") or die ('I cannot
connect to the database because: ' . mysql_error());

//select the database to use
$mydb=mysql_select_db("midwestwebdesign");

//query the database table
```

```

$sql="SELECT ID, FirstName, LastName FROM Contacts WHERE FirstName LIKE "%" . $letter . "%'
OR LastName LIKE %" . $letter . "%";

//run the query against the mysql query function
$result=mysql_query($sql);
//count results
$numrows=mysql_num_rows($result);
echo "<p>" . $numrows . " results found for " . $letter . "</p>";
//create while loop and loop through result set
while($row=mysql_fetch_array($result)){
$FirstName=$row['FirstName'];
    $LastName=$row['LastName'];
    $ID=$row['ID'];
//display the result of the array
echo "<ul>\n";
echo "<li>" . "<a href='\"search.php?id=$ID\">" . $FirstName . " " . $LastName . "</a></li>\n";
echo "</ul>";
}
}

```

The four snippets of code we changed are:

1. We use the isset Boolean check and pass in the \$_GET array and check for a value of by,
2. We create a variable called letter and assign it to the \$_GET array value,
3. Append the letter variable in our SQL statement and
4. Concatenate the letter variable inside the statement where the counted number of rows is returned. Save your file and review the results.

http://midwestwebdesign.net/tutorials/search/search_byletter.php

Searching by specific employee

In order to display the rest of the staff member's information that's passed by the unique id inside our link, we just need to add the following code right after the closing curly brace of our letter script as shown below:

```

} //end of our letter search script

if(isset($_GET['id'])){
    $contactid=$_GET['id'];

//connect to the database
$db=mysql_connect ("mysqlb11.webcontrolcenter.com", "<username>", "<password>") or die ('I cannot
connect to the database because: ' . mysql_error());

//select the database to use
$mydb=mysql_select_db("midwestwebdesign");

```

```

//query the database table
$sql="SELECT * FROM Contacts WHERE ID=" . $contactid;

//run the query against the mysql query function
$result=mysql_query($sql);

//create while loop and loop through result set
while($row=mysql_fetch_array($result)){

    $FirstName=$row['FirstName'];
    $LastName=$row['LastName'];
    $PhoneNumber=$row['PhoneNumber'];
    $Email=$row['Email'];

//display the result of the array

echo "<ul>\n";
echo "<li>" . $FirstName . " " . $LastName . "</li>\n";
echo "<li>" . $PhoneNumber . "</li>\n";
echo "<li>" . "<a href=mailto:" . $Email . ">" . $Email . "</a></li>\n";
echo "</ul>";
}
}
}

```

The four snippets of code we changed are:

1. We use the isset Boolean check and pass in the \$_GET array and check for a value of id,
2. We create a variable called contactid and assign it the \$_GET array value,
3. Select everything from our table, which is indicated by an asterisk (*). An asterisk is shorthand notation in SQL that's translated to "give me all columns and rows from the table". In order to know which information to display we append the contactid variable at the end of our SQL statement, and
4. Display the additional information about each staff member.

Save your file and review the results.

http://midwestwebdesign.net/tutorials/search/search_byid.php

As you review the changes, you should notice our functionality is working as intended. When you enter a first or last name in our form field or select a letter, only the staff member's name appears as a hyperlink. As you move our mouse over the link, in the status bar, you should notice that a unique ID has been passed inside our link and appended on our query string. When you click a specific staff member link, the address bar changes and the rest of the employee information is shown according for the unique ID passed.

SQL Injection

The reason we added a regular expression to our search form field was to ensure that no one could manipulate our SQL query to perform abnormal operations on our data or table such as delete operations, at least, not very easily. Such abnormal operations are called SQL injection attacks. These attacks are basically techniques that hackers use to try and manipulate your applications to perform operations they weren't meant to do and as a result, they cause adverse affects on applications and more importantly, database tables. For example, if we were to allow apostrophes in our search form field, a malicious attacker visiting our form could try typing the following:

```
'DROP TABLE <myTable>
```

By executing this query, it could cause My SQL or any database engine to execute a command that would delete your entire staff member table. As mentioned before our regular expression ensures that the first character coming from the \$_POST value of our form field has to be either a capital or lowercase letter. As a result, we're able to protect against malicious queries that might be attempted from our search field.

Summary

In this article you learned how to do the following:

- Create a database and corresponding table
- Use a database management system (DBMS) to create columns and enter data for the corresponding entities
- Create a search form, along with PHP code to:
 - Check for form submissions
 - Check for appropriate query string variables
 - Check for acceptable input from visitors
 - Connect, query and display results from a database table
- How to protect your application and database table from SQL injection attacks

Taking the knowledge gained from this article, developers can easily adapt or modify the code presented to perform other operations or enhance functionality for any particular instance of a search feature.

If you have questions, please follow the link below.

<http://midwestwebdesign.net/tutorials/contact.aspx>

[1] http://www.phpmyadmin.net/home_page/index.php

[2] <http://www.webyog.com/en/downloads.php>

[3] <http://www.pspad.com/>