

Creating a dynamic menu with PHP

When developing websites for large businesses or educational institutions, navigation is likely to change, and change often at that. Navigation also needs to be consistent for the entire website with links to each page from the current page. With large websites, creating a static menu is preferable during the development and prototyping stage, but in production, it's to the developer's advantage to create a dynamic menu. Dynamic menus can be created using either a server-side include or a database table. Using these methods, links are changed from one source, and reflected to the website.

In this article, we'll learn how to create a dynamic menu using a database table. This table will have three columns which contain a primary key, file name & description. We will then use PHP to connect, query & display the results from our database table. Before finishing, we'll create a server-side include which contains our PHP code to make our menu portable for each page in our website.

If you would like to follow creating a dynamic menu's step-by-step development or even try your hand at creating a menu such as this, you'll find the project files link helpful.

See Finished Page

http://midwestwebdesign.net/tutorials/dynamicmenu/index_nav_finished.php

Download Project Files

<http://midwestwebdeisgn.net/tutorials/dynamicmenu/dynamicmenu.zip>

What's required?

You need the following technologies:

- MySQL database
- PHP

You also need to be comfortable with adding, creating, modifying and deleting records inside a database table.

Create the database

If you are unsure how to create a database or table with your host, please contact your system administrator to find out the steps involved. Once this information is determined, do the following.

- Create a database
- Give the database a name, for our purposes, MyDatabase will work
- Create a user and give it privileges to the database
- Create the table

Creating the database table

For the purpose of this article, name the database table **Navigation**. The table will have three columns:

- Navigation ID
- File name
- Description

Each columns purpose is described below:

NavigationID- This column is the primary key. For each table inside a database, create one column that is your primary key. This makes each record unique for each table you create.

File Name- This column holds the file name for each page used in the site. For example, the home page will have a link to about or contact us. In this column, type **about.php** and **contact.php**. If there is additional file names later, add them to this column.

Description- This column will hold the descriptive text for each link. For example, instead of showing contact.php when the mouse moves over the link, **Contact Us** is shown.

The table structure should resemble the illustration below:

NavigationID	FileName	Description
1	index.php	Home
2	about.php	About Us
3	contact.php	Contact Us

Column Requirements for the table

Make sure you set the following requirements for each column:

NavigationID:

- Primary key
 - Data type: INT
- Auto-increments
 - Required
- Not null

FileName:

- Data type
 - VARCHAR
- Required

- Length
 - Set to 50
 - Can be changed to suit your needs if longer field is needed
- Not null

Description:

- Data type
 - VARCHAR
- Required
- Length
 - Set to 50
 - Can be changed to suit your needs if longer field is needed
- Not null

Create the file

In a text editor, create a new file named **index_nav.php**.

Note: When saving PHP files in Notepad, make sure the file name is surrounded by double quotes, otherwise, `index.txt.php`, will be the result, which does not work.

Connect to the database

In order to connect to the database, add the following code between the opening and closing `<body>` tags:

```
<?php
// -connect to our database
$db=mysql_connect("localhost", "<username>", "<password>") or die
('Cant connect to the database because: ') . mysql_error();
// -select the database that we want to use
mysql_select_db("MyDatabase");
?>
```

Let's examine the code in greater detail:

- First create a variable named **db** by using the dollar sign
- Assign the variable to the **mysql_connect** function. This function takes three parameters:
 - First, the name of the database server is dependent on your host, unless developed on your computer, then it's always **localhost**
 - Next, provide a user name, followed by the password associated with the user
- In case the connection to the database server is not available, use the die function:
 - Using the die function, we immediately terminate further processing of our script. We pass two parameters:
 - A string message indicating we can't connect

- A second parameter, **mysql_error**, which provides basic trouble shooting information, such as incorrect user name and password or access permissions

Next we have the following:

- `mysql_select_db`
 - Is a built in function, which tells MySQL which database to use

The query

When working with server-side scripts and databases, a way of extracting data from the database table is needed and then used with the server-side script to parse regular HTML code to the browser. Data is extracted from the database using Structured Query Language (SQL). While an in-depth analysis of database concept, theory and SQL is beyond the scope of this article, we'll learn enough to complete our menu. In order to inform PHP where the data is coming from, form the following SQL statement right above the closing PHP block:

```
$query="SELECT FileName, Description FROM Navigation";
```

Let's examine the code in greater detail:

- First, create a variable named query by using the dollar (\$) sign and assign it the SQL query:
 - In SQL, use the SELECT keyword to determine which columns to extract data:
 - FileName and Description in this example
 - Use the FROM keyword to determine which table holds this data, in this example, Navigation
- Terminate the PHP command with a semi-colon

The result of the SQL statement is in the query variable. If you haven't saved your file yet, now would be a good time to do so.

Processing the query result

In order to use the result found in the query variable, add the following code right below the last command:

```
$result=mysql_query($query);
```

A variable named result is created by using the dollar (\$) sign. Assign it the **mysql_query** function and pass the query variable as an argument. This function allows the database to process the results of the query on the specified table and holds the value in the result variable.

Create links

Since the database table could have hundreds, possibly thousands of records we need to create the links inside a loop, specifically a while loop. The while loop is added below:

```
while($row=mysql_fetch_array($result))
{
$FileName=$row['FileName'];
$Description=$row['Description'];
//display the results of the query in HTML format using paragraphs
print "<p><a href=\"\"\".\".$FileName.\"/\".\".Description.</a></p>";
}
```

Let's examine the code in greater detail:

- First, the while loop takes one parameter, which is a condition. In the condition do the following:
 - Create a variable named row and assign it the result of the mysql_fetch_array function, which uses the result variable as a parameter. This function takes each record from the table and places it in a zero-based array.
 - Create a variable named FileName and assign it the result of the file name array element
 - Create a variable named Description and assign it the result of the description array element
- Parse/print out a paragraph tag. Inside the paragraph tag, there is the following:
 - An anchor tag which contains the parsed value of the FileName and Description variable. Close the anchor tag.

Save your file and preview the results.

http://midwestwebdesign.net/tutorials/dynamicmenu/index_nav.php

Creating a table

Currently, the page is a bit bland even if it does accomplish what is needed. Let's put the navigation inside a table and style the links. First, add the table:

```
<?php
print "<table cellpadding=\"0\" id=\"menu\">\n";
//connect to our database
$db=mysql_connect("localhost", "<username>", "<password>") or die
('Cant connect to the database because: ') . mysql_error();
//select the database that we want to use
mysql_select_db("MyDatabase");
//declare a query variable to extract information from the navigation
table
$query="SELECT FileName, Description FROM Navigation";

//run the result of the query on the mysql_query function
```

```

$result=mysql_query($query);

//-loop through the results of the query

while($row=mysql_fetch_array($result))

{

$FileName=$row['FileName'];

$Description=$row['Description'];

print "<tr>\n<td>\n<a href=\"". $FileName. "\">".
$Description."</a>\n</td>\n</tr>\n";

}

print "</table>";

?>

```

After the starting PHP code block, parse out a table. Inside the loop, loop through the array and parse out a table row, cell and anchor tag for each record in the database table. Outside the loop, close the table, before closing the PHP code block.

Styling the table

Create a new file named **style.css**. For the sake of space, just copy/paste the CSS rules from the supplied style sheet (style.css) found in the project files link, starting at rules table#menu a – table#menu a:active.

Save your file and preview the results.

http://midwestwebdesign.net/tutorials/dynamicmenu/index_nav_table.php

Making our navigation a server-side include

If you want to make your menu portable, then a server-side include is just what you need. Making your menu a server-side include does the following:

- Eliminates the need to have the entire PHP script inside each file
- With server-side includes, you call the include in the file where it's needed
- If changes are needed, make them in the server-side include

To create the server-side include, follow these steps:

- In **index_nav_table.php**, highlight the php code between the opening and closing <body> tag
- Right click and select **Copy**
- Right click and select **Cut**

Continuing, follow these steps to create the server-side include file:

- Create a new file named **navigation.php**
- Remove all code
- Right click and select **Paste**
- Save your file

Finally, call the server-side include in **index_nav_table.php**:

```
<?php include('navigation.php');?>
```

Save your file and preview the results.

Summary

In this article, you learned:

- How to create a database and a table
- How to add column names, data types and requirements
- Connecting to our database via PHP
- Using Structured Query Language (SQL) to query the specified table
- Use a while loop to iterate through the record set in the database table to create links
- Create a table to hold the menu
- Use basic CSS to style the table
- Create a server-side include file to contain only your navigation to enable modifications to proceed easily and quickly

Using a text based menu, you can incorporate this menu in any website you want as long as you have the required technologies. Using CSS, you can format links to your imagination.

If you have questions, please follow the link below.

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