

Tab-delimited text files w/ PHP

Often in web development, developers are asked to work with many different file formats for a variety of reasons. For instance, most print publications are designed in another program besides Dreamweaver; much of the time it's Page Maker. To make this publication web friendly, you will need to convert it to Adobe PDF so visitors can view the publication, since Adobe Reader is freely available.

Often when displaying rows of information such as course availability for a school or names and addresses of class alumni from your high school on the web you need a way to dynamically generate the output from a data source. The data source can range from database tables to a simple tab delimited text file. The data source is then read by a server-side script in a web page and the result is regular HTML output to the browser. In this article, we will examine how to create a tab-delimited text file with Excel and then use PHP to read the text file in a web page to display the results.

If you would like to follow this project's step-by-step development or even try your hand at creating a web page which reads a tab-delimited text file, you'll find the following project file links helpful.

See Finished Page

<http://midwestwebdesign.net/tutorials/tabdelimited/names.php>

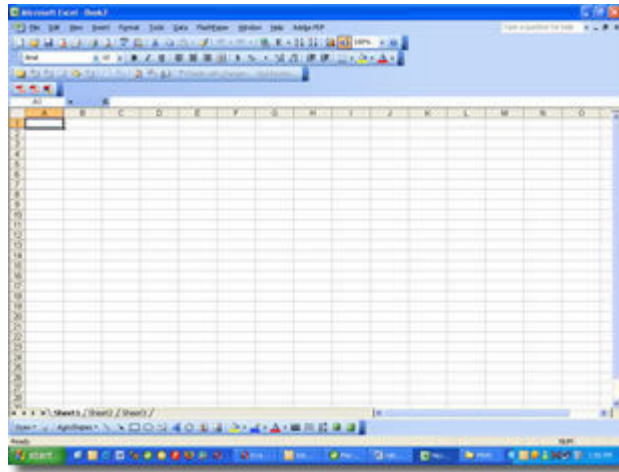
Download Project Files

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

Create tab-delimited text file

For this article's purpose, we'll be using Excel. If you don't have Excel, just use **names.txt** which is provided from the project files link. Follow these steps to create your tab-delimited text file:

- From the Desktop, locate the Start button
- Select Programs>Microsoft Office>Microsoft Excel
- The following window will show:



If you look at the finished text file, we need a table which has four (columns) and four (rows). In cells A-1 through A-4 enter the following, in order, starting at A-1:

Ryan
Megan
Brent
Melissa

For cells B-1 through B-4 enter the following in order starting at B-1:

Butler
Butler
Butler
Butler

For cells C-1 through C-4 enter the following in order starting at C-1:

Husband
Wife
Son
Daughter

For cells D-1 through D-4 enter the following in order starting at D-1:

24
26
2
5

Save your spreadsheet.

Before finishing, save the file in Excel's native format (.xls). (**Note: Office 2007 has a different file format.**) Follow these steps:

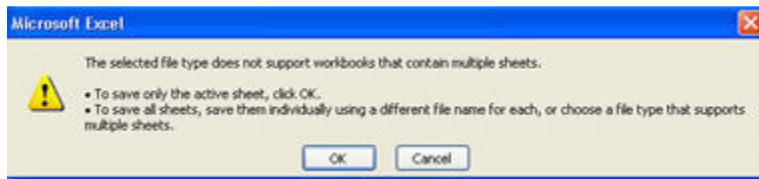
- From the main menu
- Select **File>Save**
- Save window opens
- Choose an appropriate location and file name

Next, follow these steps to create the tab-delimited file:

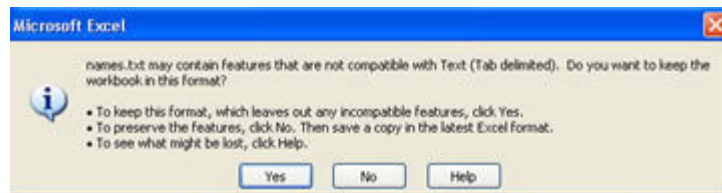
- From the main menu
- Select **File>Save As**
- Save window shows
- In the **Save as type** drop down list choose **Text (Tab delimited)**:



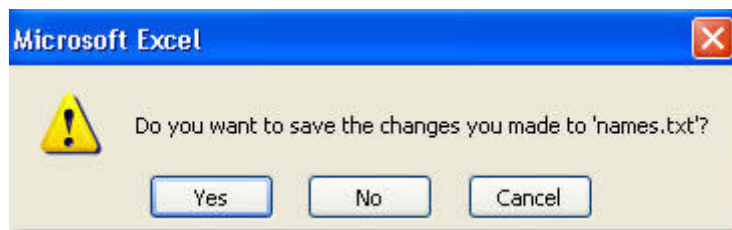
Left click Save, which prompts this warning:



Left click OK, which prompts this message:

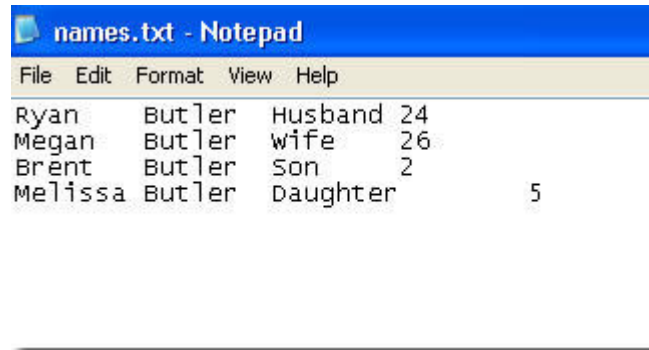


Left click yes. If you try to exit Excel, you receive this warning:



This error is a little misleading; it's referring to any changes which were made to the original spreadsheet that needs to be made to the tab-delimited text file. Since there are no changes, left click **No**.

Navigate to the directory where you saved the text file, you should see a structure similar to this:



Do note, you can at any time add values to the original spreadsheet and then re-save your text file. If you have not already, close Excel, we're done with the program.

Create our PHP script

In your text editor, create a new file named **names.php** and insert the following code:

```
<!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>List of Names</title>
</head>

<body>

<?php
?>

</body>

</html>
```

Between the opening and closing `<body>` tags, we start a PHP code block.

Reading our text file

Luckily for us, PHP has many built in functions to handle reading data from a tab-delimited text file. The first of these functions is **fopen**, which is used to open a file. Once we open a file, we'll assign it to a variable which can be used as a storage location for all data received from the text file. Add the following code to our file:

```
<?php
//create a variable, and assign it the data stream from names.txt
$theFile=fopen("names.txt","r");
?>
```

Let's examine the code in greater detail:

- First, we create a variable name **theFile** by using the dollar (\$) sign
- We assign the variable to the **fopen** function which takes two parameters:
 - First, is the name of our tab-delimited file, in our case **names.txt**
 - Second, which mode we want to use. This parameter specifies how we plan to use the data we receive from this text file. Possible options include:

Access Mode	What it does
r	Read-only
r+	Reading-writing
w	Write only and create if it does not exist. In addition, the data in the file is erased and you will begin writing data at the beginning of the file.
w+	Reading and writing and creating the file if it does not exist.
a	Write only, however, the data in the file is preserved and you begin writing data at the end of the file.
a+	Reading and writing, create the file if it does not exist and place the file position pointer at the end of the file.

Since no writing is required, we use the read-only option.

Check to see if the file opened

Just as the case with databases, we want to make sure we can open the file, if not, display an error message to our visitors. Add the following code:

```

<?php
//create a variable, and assign it the data stream from names.txt
$theFile=fopen("names.txt","r");
//run an error check on the variable to make sure we received a data
stream
if(!$theFile){
print "<p>Couldn't open the data file. Try again later.</p>";
}
?>

```

Let's examine the code in greater detail:

- First, we perform a conditional check:
 - We pass the variable (theFile) which we created to hold our result set
- Then we use the negation operator (!), which is a Boolean value

In PHP, the negation operator accepts a value (true or false) and returns the opposite value. Even though previewing this page at the moment will show nothing, it would be a good idea to save the file. As a result, the condition checks to see if we were able to open the file (true), otherwise, display an error message.

Displaying the results

Since we need to display the records from our text file in a tabular format in our web page, we'll create a table. First, we create the table:

```

<body>
<table id="family" cellspacing="0">
<tr>
<td>First name</td>
<td>Last name</td>
<td>Relationship</td>
<td>Years of Age</td>
</tr>
<?php
//create a variable, and assign it the data stream from names.txt
$theFile=fopen("names.txt","r");
//run an error check on the variable to make sure we received a data
stream
if(!$theFile){
print "<p>Couldn't open the data file. Try again later.</p>";
}
?>
</table>
</body>

```

Let's examine the code in greater detail:

- In the opening body tag:
 - We create a table with an ID of **family** and set cell spacing to zero

- Then we create the structure which is required for our tab-delimited text file:
 - One Row
 - Four Columns

After the closing table row, we have nested our PHP code block before the closing table tag. Let's also create the following CSS rules:

```
<style type="text/css" media="screen">
table#family{
width:400px;
margin:0 auto;
border-collapse:collapse;
border:1px solid #ccc;
}
</style>
```

We create a new CSS rule with an ID of family. This rule will target any table with an ID of family. The property-values are:

- Width
 - Set a fixed width of 400 pixels
- Margin
 - Top and bottom set to zero; left and right set to auto
- Border-collapse
 - Set to collapse will collapse adjacent cell borders
- Border
 - Sets a 1pixel, solid gray border around our table

Do note that setting left and right margins to auto will center our table in the browser. Continuing, add the following CSS rule:

```
table#family td{
border:1px solid #ccc;
text-align:center;
}
```

We set a descendent selector. A descendent selector describes an element which is a descendent (child) element of a parent element. In our example, our td element is a descendent of our table. More importantly, this rule targets any table cell which resides in a table with an ID of family. The property-values are:

- Border
 - Sets a 1 pixel, solid gray border on table cells
- Text-align
 - Sets text alignment of cells to center

Iterate through the result set

Since the result set is in a variable, we need a way to iterate (read) through the text file for each record. When programming, we use loops for this task. The loop, in this case "while", will grab the content and read through the text file until all records have been read. Inside the loop, we explode the content into a string array and then parse a regular table row and cell for each record.

First, let's create the "while" loop to grab and read through the text file for each record:

```
<?php
//create a variable, and assign it the data stream from names.txt
$theFile=fopen("names.txt","r");
//run an error check on the variable to make sure we received a data
stream
if(!$theFile){
print "<p>Couldn't open the data file. Try again later.</p>";
}
else{
while (!feof($theFile)) {
}
?>
```

Let's examine the code in detail:

- First, we create a "while" loop and pass two parameters:
 - First is the negation operator, followed by the **feof** function, which means **for file end of file**. This function instructs the loop to keep looping until the end of the file
 - Secondly, we pass in our variable which holds the result set

Here's how the loop works: The negation operator is a Boolean value, which accept true or false. By passing the feof function and our variable result set as a condition in the "while" loop, we instruct the loop to iterate (read) through the result set until there are no records to read.

Explode the result set

Since our variable contains one string of text from our file, we need to split it into a series of strings using the explode function as shown below:

```

<?php
//create a variable, and assign it the data stream from names.txt
$theFile=fopen("names.txt","r");
//run an error check on the variable to make sure we received a data
stream
if(!$theFile){
    print "<p>Couldn't open the data file. Try again later.</p>";
}
else{
    while(!feof($theFile)){
        $data=explode("\t", fgets($theFile));
    }
}
?>

```

Let's examine the code in greater detail:

- First, we create a variable, data by using the dollar (\$) sign:
 - We assign the variable to the explode function:
 - The explode function will return an array of strings and accepts two parameters:
 - First, we pass a delimiter, in our case, a tabbed character escape sequence (“\t”) which tells us how to handle our array of strings, in our case, tabs
- Secondly, we use another function, **fgets**, which is used to one line of text at a time from our text file

Save your file.

Displaying our records

Now that we have our records in a usable format, we need to create a table row and cell to display them. Inside this row, we will use another loop, in this case "for" which will use a condition to determine if we have less than four columns. If this is true, then we will pass our data variable in conjunction with our counter variable, expressed as an array to display data for each record inside a cell.

First, let's parse a regular table row:

```

<?php
//create a variable, and assign it the data stream from names.txt
$theFile=fopen("names.txt","r");
//run an error check on the variable to make sure we received a data
stream
if(!$theFile){
    print "<p>Couldn't open the data file. Try again later.</p>";
}
else{
    while(!feof($theFile)){
        $data=explode("\t", fgets($theFile));
        print "<tr>\n";
    }
}

```

Let's examine the code in greater detail:

- First, we use the print function to parse a regular table row
- Next, we use a character escape sequence (“\n”) to provide a new line character, which would be present in source view of our browser. This makes reading HTML output easiest to debug and read

Next, we add our loop and parse a regular table cell in conjunction with our data variable:

```
<?php
//create a variable, and assign it the data stream from names.txt
$theFile=fopen("names.txt","r");
//run an error check on the variable to make sure we received a data
stream
if(!$theFile){
    print "<p>Couldn't open the data file. Try again later.</p>";
}
else{
    while(!feof($theFile)){
        $data=explode("\t", fgets($theFile));
        print "<tr>\n";
        for($i=0; $i<4; $i++){
            print "<td>" . $data[$i] . "</td>\n";
        }
    }
}
?>
```

Let's examine the code in greater detail:

- First, we create a loop which also creates a condition:
 - In the condition, we create a variable, **i** by using the dollar (\$) sign
 - Use a logical operator, less than symbol (<) to make sure our counter variable is less than four columns
 - Increment our counter variable each time through the loop, so we can read the next record

Continuing, inside the loop we do the following:

- Parse a regular table cell using the print function
- Concatenate our data variable by using a period (.) which holds our array of strings from our variable and pass the counter variable as a parameter in order to know which record to display
- Concatenate a closing table cell as well as a character escape sequence, (“\n”), to provide a new line return which is only visible in source view of a browser

Finally, outside the loop, we have the following:

```

<?php
//create a variable, and assign it the data stream from names.txt
$theFile=fopen("names.txt","r");
//run an error check on the variable to make sure we received a data
stream
if(!$theFile){
    print "<p>Couldn't open the data file. Try again later.</p>";
}
else{
    while(!feof($theFile)){
        $data=explode("\t", fgets($theFile));
        print "<tr>\n";
        for($i=0; $i<4; $i++){
            print "<td>" . $data[$i] . "</td>\n";
        }
        print "</tr>\n";
    }
}
?>

```

Let's examine the code in greater detail:

- We parse a closing table row using the print function and use a character escape sequence (\n) to provide a new line return which is only visible in the browsers source view

Closing the data source

Since we have completed reading and parsing our records to the browser, we need to close the connection to the data source:

```

<?php
//create a variable, and assign it the data stream from names.txt
using the fopen function
$theFile=fopen("names.txt","r");
//run an error check on the variable to make sure we received a data
stream
if(!$theFile){
    print "<p>Couldn't open the data file. Try again later.</p>";
}
else{
    while(!feof($theFile)){
        $data=explode("\t", fgets($theFile));
        print "<tr>\n";
        for($i=0; $i<4; $i++){
            print "<td>" . $data[$i] . "</td>\n";
        }
        print "</tr>\n";
    }
}
//done with the file, close it
fclose($theFile);
}
?>

```

We use another function, **fclose**, which takes one parameter, our original variable which opened the connection to our data source. Save your finished file and preview the results.

http://midwestwebdesign.net/tutorials/tabdelimited/names_3.php

One slight problem

You might notice from the file above we have an extra row. This is because there's an extra record in the text file. There are two solutions:

- Remove the extra row from the text file
- Check the size of the array. In this method, we ensure our result has only four columns. We nest this check inside our second loop

The second solution is better because as you add additional records, you will encounter the same problem. Add the following check in our script:

```
<?php
//create a variable, and assign it the data stream from names.txt
using the fopen function
$theFile=fopen("names.txt","r");
//run an error check on the variable to make sure we received a data
stream
if(!$theFile){
    print "<p>Couldn't open the data file. Try again later.</p>";
}
else{
    while(!feof($theFile)){
        $data=explode("\t", fgets($theFile));
        if (sizeof($data)==4) {
        print "<tr>\n";
        for($i=0; $i<4; $i++){
            print "<td>" . $data[$i] . "</td>\n";
        }
        print "</tr>\n";
        }
    }
}
//done with the file, close it
fclose($theFile);
}
?>
```

Let's examine the code in greater detail:

- First, we use a condition after the explode function and pass the following parameters:
 - Using the sizeof function, we check the size of the array:
 - If it's equal to four columns, we output the data to the browser

Make sure you properly nest the conditional check outside the loop. Save your file and preview the results.

http://midwestwebdesign.net/tutorials/tabdelimited/names_4.php

Summary

In this article, you learned the following:

- Different file formats used on the web
- How to create a tab-delimited text file using Excel
- Different data sources, such as databases and text files, specifically, tab-delimited
- How to read a tab-delimited text file using PHP
- Built-in functions in PHP which allow us to read and work with text files:
 - open
 - fgets
 - fclose
- How to open a connection to the text file
- How to iterate and read through records using a "while" loop
- How to display each record from our text file using another loop, in this case, "for"
- How to check for an extra record in a text file using a simple conditional check which determines the size of an array

If you have questions, please follow the link below.

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