
# Creating a Mailing List Manager with PHP

## By Duncan Lamb

# Table of Contents

# Introduction

There are many, many uses for mailing lists, from informing users of news about your site to alerting interested people about an update to a project you are working on. Coupled with some management tools, disseminating information through a mailing list can be a snap, and new users can add themselves and be welcomed to the to the list without your involvement.

There are programs available for download and for purchase which promise to do the dirty work for you, but with a little effort, PHP offers all the tools you need to create a full−featured management application which you can access from anywhere on the net. Best of all, doing it yourself will give you the skills you need to customize the application to precisely fit your needs, which is a true mark of a scripting professional.

This tutorial will guide you through building a complete mail list administration solution, capable of handling multiple lists. Suitable for small−medium sized lists (less than 500 names), flat files are used to store information for later retrieval. Many file access commands will be used and explained, and we'll cover how to send mail from a PHP script.

# Creating the front page

The structure of the application will be straightforward. All the scripts will be in the main directory, and all data files in a data/ directory. Each list will have its addresses stored in a separate file in that directory. There will also be a log file, saved as a simple text file.

The lists we are going to use (only one at the moment) are stored in the **lists.txt** file in the **data** directory. The format for the file has one list per line, with the "familiar" list name and the filename for the list separated by a pipe ("|"). This file will be used to build the select box used to specify which list to perform a particular action on. By grabbing the filename ($Filename) from a form, it is easy to grab the addresses out of the specified list.

I'm a big believer in putting the most often used features up front for ease of use. Once the application is up and running, the most often−used feature would be sending an email to the list. With that in mind, we'll build a "Send to" form on the index page, along with links to all the administrative functions:

index.php3

```
<html><head><title>Mailing List
Administration</title></head><body>
<br>
<center><H1>Mailing List Administration</H1></center>
Send an email to a mailing list:
<form method=post action="sendemail.php3">
<table><tr><td>
<b>From Address:</b>
<input type=text name="From" size="40" value="">
<br>
<b>Subject:</b><input type=text name="Subject" size="40">
</td><td><table cellspacing=15><tr><td valign=top>
<b>List:</b>
</td><td>
<select name="List" size=4>
<?
$groups = file("data/lists.txt");
for ($index=0; $index < count($groups); $index++)
{
$grouplist = split("\|", chop($groups[$index]));
?>
<option value="<? echo $grouplist[1] ?>"
<? if ($index==0)
{echo "selected";} ?>
<? echo $grouplist[0] ?><br>
<?
}
?>
```

```
</select></td>
<td valign=top><b><a href="newlist.php3">Make a new
list.</a></b>
<br><a href="addnames.php3">Add names to a list</a>.
<br><a href="picklist.php3">Edit/Delete names</a>.
<br><a href="data/log.txt">View Send Log</a>.
<br><a href="autoresponder.php3">View/Edit Autoresponder</a>.
</td></tr></table>
</td></tr></table>

Type or paste your message below:
<br><textarea cols=50 rows=10 name="Body"></textarea>
<br><br>

<input type="submit" name="Submit" value="Send Mail">
</form>
<br>
</body></html>
```

You'll notice the entire script is normal HTML, except for lines 14−26. These lines read in the lists.txt file, spilt each line on the "|", and format them into the select list. (The **file** function is explained in more detail below.) This produces a tidy list, which will display the common name of the list to the user, and store the filename for the list as the value of the picked option.

Note: To make things even easier, set a default "value" for the "From Address" in the form. By doing so, that blank will automatically be filled in for you, but still allow you to change it if needed.

By grabbing the filename ($Filename) from this form on the index page, it is easy to grab the addresses out of the specified list.

When submitted, an entire list is mailed the message. The data goes directly to this script:

sendemail.php3

```
<html><head><title>Updating file....</title></head><body>
<?
$addresses = file("data/$List");

for ($index=0; $index < count($addresses); $index++)
{
mail("$addresses[$index]","$Subject",
"$Body","From: $From\nReply-To: $From");
}

$myfile = fopen("data/log.txt","a");
fputs($myfile,$Subject."\t".date("dS of F Y h:i:s
```

```
A")."\t".$List."\n");
fclose($myfile);
?>

Your message was sent!
<br><br>
<a href="index.php3">Home</a>.
</body></html>
```

This script makes use of some very useful functions. First is the **mail** function (line 7), one of the most high−impact, yet easy to use functions that exist in PHP. By default, it accesses sendmail on a UNIX system, but this can be changed to another server by editing the php.ini file. The format for using it is as follows:

mail(string to, string subject, string message, string [additional_headers]);

For additional headers, you can include everything observed by looking at the headers of your mail messages, including X−Mailer, Reply−To, CC:, Bcc, Mime−Version, you can even make up your own headers if you are so inclined.

To retrieve email addresses (lines 3−5), the **file** function is used. This function reads the entire file into an array, with each line in the file represented by an element in the array. Since the names are stored in the file one to a line, each element in the array will contain an email address. Once this is done, looping through the addresses, and calling mail for each one will send your message out individually to each list member.

Note: An important caveat to using the file function is that the newline is also stored in each array element. This makes it necessary to chop() or trim() each element in many situations.

The final step in the script is to make a log entry so we can track the messages we send (lines 11−13). Here a log is kept in its simplest form, just a text file. The "\t" characters used translate to tabs, much like "\n" translates to a newline character.

Another interesting function is used here – the **date** function. It has some rather involved formatting rules, which are adequately explained in the PHP documentation. Be sure to read it, and try to use it in sensible places, as it does allow quite a few options.

http://www.devshed.com/Server_Side/PHP/Manual/manfiles/function.date.html

# Adding names to the list

Here, we'll automate the process of adding a name to the list:

addnames.php3

```
<html><head><title>Add an email to the
list</title></head><body>
<br><br>
<form method=post action="saveemail.php3">
<table><tr><td valign=top>
Which List?:
</td><td>
<select name="List" size=4>
<?
$fileloc = "lists.txt";
$groups = file("data/lists.txt");
for ($index=0; $index < count($groups); $index++)
{
$grouplist = split("\|", chop($groups[$index]));
?>
<option value="<? echo $grouplist[1] ?>"
<? if ($index==0)
{echo "selected";} ?>>
<? echo $grouplist[0] ?><br>
<?
}
?>
</select></td></tr></table>

Email Address:<input type=text name="Email" size="40">
<br><br>

<input type="submit" Value="Add This Email"></FORM>
<BR><BR><a href="./">Home</a>.
<br><br><a href="picklist.php3">Edit/Delete names</a>.
<br><br><a href="data/log.txt">View Send Log</a>.
<br>
</body></html>
```

First, a list is chosen in lines 7−21 (this is the same function used on the index page), and then it and the email address is passed to the next script:

saveemail.php3

```
<html><head><title>Updating file....</title></head><body>
<br><br>
<?
if (file_exists("data/$List"))
{
$myfile = file("data/$List");
$fh = fopen("data/$List","w");
for ($index=0; $index < count($myfile); $index++)
{
if ($Email != chop($myfile[$index]))
{fputs($fh,$myfile[$index]);}
}
fputs($fh,$Email."\n");
fclose($myfile);
}
else
{
$myfile = fopen("data/$List","w");
fputs($myfile,$Email."\n");
fclose($myfile);
}
?>

<br>
<? echo $Email ?> written to <? echo $List ?>
<br><br>
<a href="index.php3">Home</a>.
</body></html>
```

After making sure the file exists, the first thing this script does is read the current list into an array (lines 4−6), then checks for each name to see if it already exists in the list (no one likes being on the same mailing list TWICE!). It does this in lines 8−12 by writing each name back into the file, and just skipping the name if it already exists. Once every name is compared, the new name is written at the end of the file.

You'll notice that **fopen** uses "w" here, for writing, while it used "r" on the index page (for reading) and "a" is used on the Send Email page(for appending to an existing file). The format for the command is:

fopen (filename, mode)

With mode being either "r", "w", or "a". Writing an existing file erases the current contents, while appending starts writing at the end of the file. Each of these modes can allow both reading and writing by placing a "+" after the mode. But the initial state (whether the current contents are deleted, and where the initial pointer is) will still occur.

*Note: If you are going to open a file for reading and writing simultaneously, nasty things can happen that can destroy data in other files if a problem occurs. Good programming practice dictates you should modify a file by reading it into an array, closing the file connection (which file does automatically), making the*

*modifications, and writing the changed contents back to the file.*

Adding lists uses the same technique, Just joining the List and Filenames together with a "|". Here's the scripts used for to add a list to your collection:

newlist.php3

```
<html><head><title>Mailing List
Administration</title></head><body>
<br>
<b>Both blanks must be filled in!</b><br>
<form method=post action="makenewlist.php3">

<b>Name of the list:</b><input type=text name="Listname"
size="40">
<br><br>
<b>One word description of the list:</b>
<input type=text name="Filename" size="40">
</td><td><table><tr><td valign=top>


<br><br>

<input type="submit" name="Submit" value="Make list">
</form>
<br><br><a href="addnames.php3">Add names to an existing
list</a>.
<br><a href="picklist.php3">Edit/Delete names</a>.
<br><br><a href="data/log.txt">View Send Log</a>.
<br>
</body></html>
```

```
<html><head><title>Updating file....</title></head><body>

<?
$Filename = $Filename.".lst";
$myfile = fopen("data/lists.txt","a");
fputs($myfile,$Listname."|".$Filename."\n");
fclose($myfile);

?>
Created new list <? echo $Listname ?>.<br>
<br><br><a href="index.php3">Home</a>.
<br><br><a href="addnames.php3">Add names to the list</a>.
```

```
<br><br><a href="picklist.php3">Edit/Delete names</a>.
<br><br><a href="data/log.txt">View Send Log</a>.
<br>
</body></html>
```

---

# Editing the names

To edit the names, we'll use a crude but effective method: direct editing! Incorporating the editing into a form will make it less daunting, however. Observe the following code:

First, the user should pick the list to edit:

picklist.php3

```
<html><head><title>Pick the list</title></head><body>
<center>
<br><br>
Please pick the list you would like to edit:
<br>
<form method=post action="editnames.php3">

<select name="List" size=4>
<?
$fileloc = "lists.txt";
$groups = file("data/lists.txt");
for ($index=0; $index < count($groups); $index++)
{
$grouplist = split("\|", chop($groups[$index]));
?>
<option value="<? echo $grouplist[1] ?>"
<? if ($index==0)
{echo "selected";} ?>>
<? echo $grouplist[0] ?><br>
<?
}
?>
</select>
<br><br>
<input type="submit" value="Edit list"></form></center>
</body></html>
```

Then, a form is created to allow editing and deleting of the names:

editnames.php3

```
<html><head><title>Edit Maillist
addresses</title></head><body>
<form method=post action="writenamefile.php3">
<br>
```

```
Editing <? echo $List ?>.
<br><br>
Fix an address by editing in place, or delete an address
by deleting the WHOLE line. <b>No blank lines or spaces
allowed! </b>
<br><br><textarea cols=50 rows=20 name="Body">
<?
if (file_exists("data/$List"))
{readfile("data/$List");}
?>
</textarea>
<br><br>
<input type="hidden" name="List" value="<? echo $List ?>">
<input type="submit" name="submit" value="Save This
List"></FORM>
<br><br><a href="addnames.php3">Add names to the list</a>.
<br><br><a href="data/log.txt">View Send Log</a>.
<br>
</body></html>
```

The **readfile** function (line 8) reads in a file and directly sends in to the browser. This places all the addresses for the specified list into a textbox, where they can be deleted or fixed directly. Also, note the warnings about one address to a line, something to remind the administrator about what's right and wrong. The next script saves all the changes:

writenamefile.php3

```
<html><head><title>Updating file....</title></head><body>
<br><br>

Changes saved to <? echo $List ?>.<br>
<?
$myfile = fopen("data/$List","w");
fputs($myfile,$Body);
fclose($myfile);
?>
<br>
<a href="index.php3">Home.</a>
</body></html>
```

This was just as easy as reading the file in! Since the whole list was in the textbox, it is all included in the variable $Body, newlines and all. Writing $Body to the file saves the entire thing exactly as it looked in the previous form.

# Autoresponders

Autoresponders are mechanisms that automatically email a message to a user when a certain event is completed. Usually, this would be when a new user signs into the list by using a form. To edit the autoresponder, a technique similar to what we used to edit addresses will be used:

autoresponder.php3

```
<html><head><title>Edit Maillist
addresses</title></head><body>
<form method=post action="writeautoresponder.php3">
<br>
This is the automatic message sent to people who submit
their name to the Test List mailing list. See Page 5 in the
tutorial to see an example of using this feature.
<br><br>
<textarea cols=70 rows=20 name="Body">
<? readfile("data/autoresponder.txt"); ?>
</textarea>
<br><br>
<input type="submit" name="submit" value="Save This
List"></FORM>
<br><br><a href="addnames.php3">Add names to a list</a>.

<br><br><a href="data/log.txt">View Send Log</a>.
<br>
</body></html>
```

And then the form saves our changes:

writeautoreponder.php3

```
<html><head><title>Updating file....</title></head><body>
<br><br>
<b>The following autoresponder message has been saved:</b><br>
<?
$myfile = fopen("data/autoresponder.txt","w");
fputs($myfile,$Body);
fclose($myfile);
?>
<br>
<pre><? echo $Body ?> </pre><br>
<br>
<a href="index.php3">Home.</a>
```

```
</body></html>
```

To activate it, create a code snipet which will put a small form on any page you choose. Something similar to this would work anywhere:

```
<b>Get notified of updates by email:</b><br>
<form method="post" action="thanks.php3">
<input type="text" name="Email" size="20">
<input type="submit" Value="Subscribe"></form>
```

All that's left is a processing and Thank You page, which borrows code from the other pages made earlier:

thanks.php3

```
<html><head><title>Thanks!</title></head><body>
<?

$Body = readfile("data/autoresponder.txt");
mail("$Email","Welcome to my mailing list!","$Body","From:
Me\nReply-To: me@myaddress.com");

$myfile = file("data/mylist.lst");
$fh = fopen("data/mylist.lst","w");
for ($index=0; $index < count($myfile); $index++)
{
if ($Email != chop($myfile[$index]))
{fputs($fh,$myfile[$index]);}
}
fputs($fh,$Email."\n");
fclose($myfile);

?>

Thank you!
<br><br>
<a href="index.php3">Home</a>.
</body></html>
```

Note line 4 – it's a lightly documented trick that allows you to read an entire file into a variable, which is exactly what we need to do here. Of course, if you use this script yourself, it will probably be in another directory, so watch your paths. Also note this is just for one list – what would you need to do to make multiple autoresponders?

And that’s it! Now your new members will immediately receive a welcome message, or be sent important info, receive a joke of the day, or whatever else you can dream up!

# Conclusion

By using PHP creatively and intelligently, creating small, powerful application such as a mailing list manager can ease your administration tasks, and automate much of your busy work. Using an all file−based system such as this means you can put it to immediate use without access to a database. As mentioned before, the one major drawback of using files for storage is scalability – with lists over 500 names, modifying these scripts to use mySQL tables would be a wise move.

These ideas can also be extended to other applications which can really be a boon to your users, and keep them coming back. One of the best simple applications these scripts could be easily modified to perform is a "Question of the Day" sort of setup, such as that used at http://www.cramsession.com/lists/cramsubscribe.asp

Putting a number of questions in a mySQL database, and using cron−activated scripts to grab a question a day from the database would be a near−autopilot solution which looks very sophisticated, but requires almost no administration. Think about innovative uses of these techniques and how you can use them to encourage visitors to register with you and return for more.

Resources:

PHP function references:

Mail:
http://www.devshed.com/Server_Side/PHP/Manual/manfiles/function.mail.html

File:
http://www.devshed.com/Server_Side/PHP/Manual/manfiles/function.file.html

readfile:
http://www.devshed.com/Server_Side/PHP/Manual/manfiles/function.readfile.html

fopen:
http://www.devshed.com/Server_Side/PHP/Manual/manfiles/function.fopen.html

fputs:
http://www.devshed.com/Server_Side/PHP/Manual/manfiles/function.fputs.html