Dreamweaver Developer's Instant Troubleshooter

Rachel Andrew

Gareth Downes-Powell

Nancy Gill

Kevin Marshall

Drew McLellan

Dreamweaver Developer's Instant Troubleshooter

ISBN (pbk): 1-59059-233-6

Printed and bound in the United States of America 12345678910

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY, 10010 and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany.
In the United States: phone 1-800-SPRINGER, email orders@springer-ny.com, or visit *http://www.springer-ny.com*. Outside the United States: fax +49 6221 345229, email orders@springer.de, or visit *http://www.springer.de*.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, email *info@apress.com*, or visit *http://www.apress.com*.

The source code for this book is available to readers at *http://www.apress.com* in the Downloads section.

Introduction

Dreamweaver MX is a complex tool. With its new features for building web sites with server-side scripting, standards-compliant code, advanced template features, and Cascading Style Sheets, it can hold many traps for the unwary. Sometimes you'll end up with problems you just can't find answers for.

This book is here to help you find those answers. Rather than focusing on the basics of the interface, this book deals with the problems encountered with the more complex parts of the program. So whether you're having difficulty setting up IIS to test your PHP pages or you have a flexible layout quandary that you just can't get your head around, this is the book for you.

Each chapter focuses on a particular area of Dreamweaver MX development, looking at the common problems and providing solutions. It's written by people who are members of Team Macromedia or use Dreamweaver every day, so they know which problems come up regularly and, more important, how to solve them.

Who Is This Book For?

This book is for the experienced Dreamweaver user who has come up against problems when experimenting with the new features available in MX. If you know everything about client-side development and HTML, but not much about PHP, ASP, and ASP.NET, then this book will help you set up a development environment. If you're mystified by templates, CSS, and web standards, then this book will enlighten you.

What's Inside?

Each chapter focuses on a particular problem area:

- **Chapter 1** looks at site definitions and the important steps to follow when creating them.
- **Chapter 2** covers databases and the problems you may encounter with them in ASP, ASP.NET, and ColdFusion.
- **Chapter 3** teaches you how to set up Microsoft Internet Information Services (IIS) on your own PC, so that you can test ASP or PHP pages on your own machine.
- **Chapter 4** looks at installing Apache, a free open source web server, on Windows and configuring PHP to work with it.
- **Chapter 5** details how to install Apache on the Linux OS as a test machine for PHP.
- **Chapter 6** shows you how to install MySQL and covers typical problems you may have when connecting to the database with Dreamweaver MX.
- **Chapter 7** introduces the CSS tools in Dreamweaver and suggests solutions to problems like Netscape 4 compatibility and styling forms.
- **Chapter 8** details the new template features in Dreamweaver MX.
- **Chapter 9** is about web standards, why you should use them, and how to do so in Dreamweaver MX.
- **Chapter 10** shows you how to build flexible layouts with CSS—pages that stretch to fill a browser window.
- **Chapter 11** focuses on the most common PHP development problems, from sending e-mail to uploading files.
- **Chapter 12** helps you with some common ASP problems, including DSN-less connections and cookies.
- **Chapter 13** is full of advice for the ASP.NET user, including server controls, code-behind, and the page life cycle.

What Do You Need to Begin?

A copy of Dreamweaver MX is essential for all chapters. Some chapters may require you to download installation files from the Web, so a fast Internet connection would be helpful.

Support/Feedback

If you spot an error, please let us know about it by submitting it on this book's page at *http://www.apress.com*.

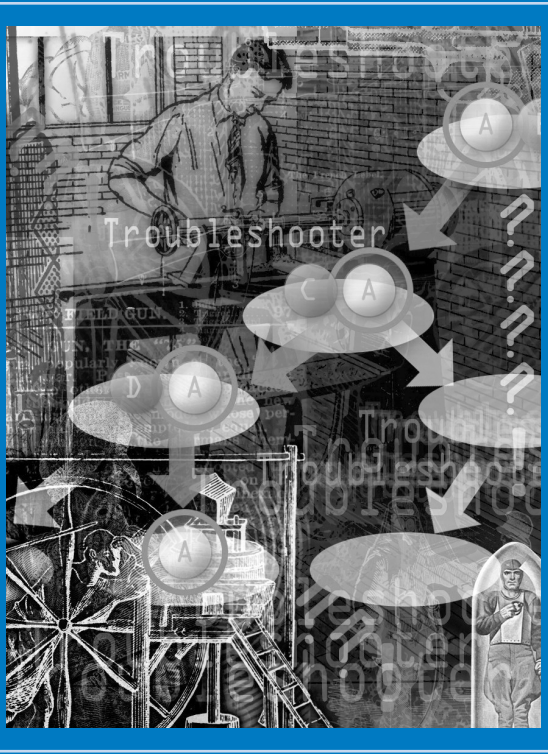
Web Support

You'll want to go and visit our web site, at *http://www.apress.com*. It features a freely downloadable compressed version of the full code for this book, in both .sit and .zip formats, for Mac and Windows users. You can also find details of all our other published books, author interviews, and more.

11

In this Chapter

- Uploading files using PHP
- HTML e-mail in PHP
- Date and time problems
- Common PHP problems



Author: Gareth Downes-Powell

Top PHP Questions

In this chapter, we show working solutions for many of the PHP questions that are often asked. We look at dealing with session variables, file uploading, e-mail, date and time, common PHP errors, and other common queries.

For each question we show some basic working code that you can easily expand on for your own projects. For information on installing PHP and MySQL, see Chapters 5, 6, 7, and 8.

Why Don't Session Variables Work?

One of the main problems faced by users of PHP is that once they have installed it, they can't get session variables to work. The main cause of session variables not working on a new server installation is the default settings in the `php.ini` file. Information held in session variables is actually stored in files in a temporary directory on the server. PHP must be able to write to this directory successfully to use session variables. The setting that causes the problem is the default `session.save_path` in the `php.ini` file, which on a new installation is usually set to the following:

```
session.save_path = /tmp
```

On Windows systems, this directory doesn't exist, and it isn't created during the PHP installation, so PHP has nowhere to store session data. To fix this problem, create a directory somewhere on your server, such as `C:/temp`, and then edit the `php.ini` file and change the `session.save_path` setting to point to this new directory. For security reasons, make sure this directory is outside of the web folder, for example:

```
session.save_path = "C:/temp";
```

You'll then need to restart the web server so that the new setting takes effect, unless you're running PHP as a CGI module. On a Linux server running Apache, this situation doesn't occur so often, as most Linux servers do have a `/tmp` directory. However, it's a good idea not to use the `/tmp` directory and instead create a new directory exclusively for PHP temporary files. For example, you can create a directory at `/home/phptemp` using the command

```
mkdir /home/phptemp
```

Next, open the Apache configuration file, `httpd.conf`, and find which user the server runs under (usually nobody). Next, give PHP write permissions by using the command

```
chown nobody /home/phptemp
```

where the user that Apache runs under is `nobody`. Restart Apache, and session variables should now be working.

How Do I Check That Session Variables Are Working?

To check that session variables are working, you need to create two simple pages, one called `set_session.php` and one called `read_session.php`. The first page will set a session variable, and the second page will read it back.

`set_session.php` contains the following:

```
<?php
// Code to place a value in a Session Variable
session_start();
session_register("testSession");
$HTTP_SESSION_VARS['testSession'] = "Sessions are working!"
?>
<html>
<head>
<title>Set Session</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<a href="read_session.php">Read Session</a>
</body>
</html>
```

This page starts session support by using the `session_start` function. Next, you register a session variable with the name `testSession` using `session_register`. Finally, you assign it a value.

read_session.php includes the following code:

```
<?php
session_start();
?>
<html>
<head>
<title>Read Session</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<?php
echo $HTTP_SESSION_VARS['testSession'];
?>
</body>
</html>
```

This page uses session_start again to enable sessions, and then you print the value held in the testSession session variable to the screen.

Open a browser and request set_session.php. You can then click the *Read Session* link to go to the page read_session.php, and you should see the message *Sessions are working!*, which was stored in the session variable.

If the preceding code doesn't work, it could be because your browser is set not to allow cookies. PHP relies on cookies to store the ID of the users' sessions as they move between pages. If your browser does not allow cookies, then the session ID is lost. To check if this is the case, change the link in the set_session.php page from

```
<a href="read_session.php">Read Session</a>
```

to

```
<a href="read_session.php?<?php echo SID;?>">Read Session</a>
```

This uses the URL to send the session ID, and when you click the link to go to the read_session.php page, you'll see the URL is similar to the following:

http://linux/session_read.php?PHPSESSID=d00ac744017c1e782e10db9eb59a512e

The read_session.php page should now show the message *Sessions are working!*, as PHP can now identify the session.

You can set PHP to include the session ID in the URL automatically by setting the option `session.use_trans_sid` to 1 in your `php.ini` file. Be aware, though, that including the session ID in the URL is not entirely secure and is not recommended, because the session ID could be bookmarked or sent to another person.

When Do I Have to Use session_start?

The PHP `session_start` function needs to be on every single page that uses session variables, and it should be placed in the header of the page—in other words, above the `<html>` tag. Because `session_start` needs to be on every page that uses session variables, it's a good idea to add the following line of code to the top of your sites' template:

```
<?php session_start(); ?>
```

How Do I Upload a File Through the Browser?

The first job is to create the HTML form through which the user selects the file to upload. This consists of the following HTML code:

```
<form action="<?php echo $PHP_SELF; ?>" method="post"
enctype="multipart/form-data" name="form1">
<input name="filename" type="file" id="filename"><br>
<input name="Upload" type="submit" id="Upload" value="Upload">
<input name="MAX_FILE_SIZE" type="hidden" id="MAX_FILE_SIZE"
value="50000">
</form>
```

To create the form, insert a normal form onto your page (*Insert > Form*), and change the form `action` and `enctype` to the preceding values to allow file uploading. To create the file field, select *Insert > Form Objects > File Field*. This inserts a field for the filename and a button that the users can click to browse for a file on their computer. Set the name of the file field to `filename`. Next, add a submit button so the user can submit the form (*Insert > Form Objects > Button*), and change its label and value to `Upload`.

Finally, add a hidden field, and set its name to `MAX_FILE_SIZE` and its value to `50000`. This is the maximum size in bytes of the file that the user can upload for that particular form. This value cannot be higher than the `upload_max_filesize` setting in the server `php.ini` file.

The next step is to open the page in Code View and adding the following PHP code block to the top of the page, before the `<html>` tag:

```
<?
if($_POST['Upload'] == "Upload"){
$originalFileName = $_FILES['filename']['name'];
$tempFileName = $_FILES['filename']['tmp_name'];
$siteRoot = "c:/webserver/";
$newPath = $siteRoot . "/data/" . $originalFileName;
umask(0);
move_uploaded_file($tempFileName, $newPath);
system("chmod 755 $newPath"); //Linux only
}
?>
```

First, you check to see if the submit button named and labeled *Upload* was used to submit the form. If it was, then you run the block of code in the `if` statement. Next, you get the original name of the file and the temporary uploaded filename from the PHP `$_FILES` array. This array holds the following information:

- `$_FILES['filename_field']['name']` – The original filename of the file.
- `$_FILES['filename_field']['type']` – The MIME type of the file. For example, `image/jpg`.
- `$_FILES['filename_field']['size']` – The size in bytes of the uploaded file.
- `$_FILES['filename_field']['tmp_name']` – The temporary name of the uploaded file.
- `$_FILES['filename_field']['error']` – An error code if there was an error (PHP 4.2.0 >).

Next, you create a new path for the file, telling PHP where on the server the file is to be stored (in this case, in a directory called `data`, which is in the root directory of the web site). Finally, you use the PHP `move_uploaded_file` function, passing the temporary filename allocated when the file was uploaded, with the new name and path on the server. If there is already a file with the same name, it will be overwritten by the new file. If you want to avoid this, you can check to see if there is an existing file with the same name by using the PHP `file_exists` command, passing it the filename to check. If the `file_exists` command returns `true`, then you can change the name of the file before you move it with the `move_uploaded_file` command, so the original file is not overwritten.

Finally, you change permissions on the image, so everyone can read it. The PHP `umask(0)` command is used to temporarily remove any permissions mask that may exist, so the exact permissions you specify are set. Note that `$siteRoot` should be set to the path to your web site root directory, in terms of a server path rather than a web path. For example, if your server is IIS running locally at *http://localhost/*, then the path might be `c:/inetpub/wwwroot/`.

This completes all the code needed to actually upload the file. Note that it's important that PHP has write permissions on the directory in which the images will be stored. On Linux, you need to use the following command to set the correct permissions:

```
chmod 777 directoryname
```

Displaying an Uploaded Image

If the file upload is for images, then you can add the following code to your page, underneath the form, and this will display the image once it has been uploaded. Note that the dimensions of the image are read automatically using the PHP `getimagesize` function. We look at this function in more detail in the "Miscellaneous Questions" section at the end of this chapter.

```
<?php
if($_POST['Upload'] == "Upload"){
$size = getimagesize($newPath);
echo "<img src=\"data/" . $originalFileName. "\" width=\"" . $size[0] .
"\" height=\""
$size[1]. "\">";
}
?>
```

The preceding code again only runs if the form has been submitted with the *Upload* button. Once it's run, the code prints out an HTML image tag to display the image.

How Do I Get a List of Files in a Directory?

To get a list of files in a certain directory, you use the PHP `readdir` function, as shown in the following example code, which you can add to the body of a PHP page:

```
<?php
$mainDir = "c:/webserver/data/";
if (file_exists($mainDir)) {
$handle = opendir($mainDir);
while (false !== ($file = readdir($handle))) {
if($file !== "." && $file !== ".."){
echo $file . "<br>";
}
}
closedir($handle);
}
?>
```

First, you create the path to the directory you want to list in terms of a server path to the directory (rather than the web path). Before you carry on, you use the PHP `file_exists` function to check that the directory in the path does actually exist. If it does, you create a handle to the directory with the PHP `opendir` function. You then use a `while` loop to read in the files in the directory using the PHP `readdir` function, and display the filenames. When `readdir` has read to the end of the directory, it will return `false`, and the code stops. To check for this, you use the `while` loop, set to run when the output from `readdir` is not equal to `false`, which is done with the operator `!==` (which means "not equal to").

The filenames returned by the `readdir` command include the directory names "." and "..". You screen these out by checking that the filename isn't one of these two names before printing it to the screen.

> *This code comes in handy on web-based file manager pages, allowing users to manage their uploaded files.*

This code comes in handy on web-based file manager pages, allowing users to manage their uploaded files.

E-mail

In this section, we are going to look at some of the frequently asked questions regarding e-mail and PHP. We will begin by looking at a more fundamental issue: how to actually send an e-mail as HTML.

How Do I Send an E-mail As HTML?

As the PHP `mail` function defaults to sending plain text e-mails unless otherwise specified, a frequent question is how to send HTML e-mails using the `mail` function. The format for the `mail` function is as follows:

```
mail($to, $subject, $message, $headers);
```

where `$to` is the e-mail address to send to, `$subject` is the subject line for the e-mail, `$message` contains the e-mail message, and `$headers` contains any optional headers you may wish to add.

To send an e-mail that's recognized and treated as HTML, you need to use two special headers:

```
MIME-Version: 1.0
Content-type: text/html; charset=iso-8859-1
```

These should be included in the `$headers` variable as shown in the following complete code:

```
<?php
$to = "gareth@myemail.com";
$subject = "This is an HTML E-mail";
$message = " <span style=\"font-family: Arial, Helvetica, sans-serif;
font-size: 20px; font-weight: bold; color: #009900\">";
$message .= "This is an HTML e-mail message";
$message .= "</span>";
$headers  = "MIME-Version: 1.0\n";
$headers .= "Content-type: text/html; charset=iso-8859-1\n";
$headers .= "From: gareth<gareth@myemail.com>";
mail($to, $subject, $message, $headers);
?>
```

It's important to note that to run the code, you need to have an e-mail server set up, so it's easiest to test this code on your web host's server.

As the e-mail is being sent as HTML, you can include HTML tags in the message, as shown in the preceding code. When the e-mail is received, the e-mail will be rendered as an HTML page (assuming that the user's e-mail program has that facility).

How Do I Send a Newsletter with PHP?

If you wish to send a newsletter out to a number of users at once, hiding the e-mail addresses so the recipient can't see who else you sent the mail to, you can use the `bcc` header, which stands for "blind carbon copy." Any addresses in the `bcc` header will be sent a copy of the e-mail, but they can't see who else it was sent to. The following code shows a working example, which sends the e-mail to each address in the array provided:

```
<?php
$addresses = array("fred@cemetry.com","george@another.com");
?>
<?
$to = "gareth@myemail.co.uk";
$subject = "PHP Newsletter";
$message = "This e-mail shows how to use a BCC Header to send a
newsletter";
$headers .= "From: gareth<gareth@myemail.co.uk>\n";
$headers .= "bcc: ";
$count = 0;
foreach($addresses as $address){
if($count == 0){
```

```
$headers .= $address;
}else{
$headers .= ", " . $address;
}$count ++;
}
$headers .= "\n";
mail($to, $subject, $message, $headers);
?>
```

In the preceding code, all the e-mail addresses are specified in the `$addresses` array. This could be changed to use a field from a recordset to get the e-mail addresses from a database table.

If you are sending the same e-mail to a large number of users, the previous method is the best way. This is because PHP contacts the mail server once, and then the mail server has the job of sending the e-mail to all the e-mail addresses specified, which means that the PHP script finishes quicker and uses fewer resources.

If, however, you're sending a personalized e-mail, each e-mail will have to be sent individually by PHP to add the individualized data.

How Do I Stop a Script from Timing Out When Sending Many E-mails?

If you're sending a personalized e-mail to a large number of users, you have to send each e-mail individually, and it may take a while for the script to send out all the e-mails. This creates a problem sometimes, as the script may time out before all the e-mails have been sent.

To avoid this, you need to increase the amount of time that the script can run for. The default setting is usually **30** seconds. You can increase the time limit for a script by adding the following code to the top of the page:

```
<?php set_time_limit(5*60); ?>
```

This will allow the script to run for up to 5 minutes (60 * 5 = 300 seconds) .

Date and Time Problems

In this section, we look at commonly asked questions regarding the `date` function in PHP.

How Do I Read the Date or Time from the Server?

To read the date or time from the server, you need to use the PHP `date` function. You pass the `date` function a string, with special tokens in place of the date parameters you require. When the code is run, the tokens will be replaced with the date or time section that they represent. For example:

```
<?php echo date("d/m/Y"); ?> outputs 11/02/2003
<?php echo date("dS F, Y");?> outputs 11th February, 2003
<?php echo date("g:i a"); ?> outputs 9:10 pm
<?php echo date("H:i:s"); ?> outputs 21:10:48
<?php echo date("l, dS F, Y"); ?> outputs Monday, 21st January, 2003
```

Notice that you can put your own characters in the string that you pass to `date`, which can act as separators for the data, such as : or /. Using the tokens, you build up a date or time in any format you choose. It's worthwhile reading the manual page for the `date` function at the PHP online manual at *http://www.php.net/manual/en/function.date.php.*

How Do I Convert a Date to and from MySQL Format?

Another common question is how to work with dates and MySQL, as MySQL uses its own date format: yyyy-mm-dd. This means that if you want to display the date, you need to change the format to a more normal one, such as the U.S. date format: mm/dd/yyyy.

The easiest way to convert the date is to use the PHP `explode` function. The `explode` function takes a string and splits it by a common separator into an array. This is very useful for transforming dates, as dates have a common separator and so are easy to split into separate parts, which can then be rearranged into your desired format.

Note that you can also format the date in your SQL query using the MySQL `DATE_FORMAT` command, although this does make your SQL queries longer. Full details are available on the following page of the online MySQL manual: *http://www.mysql.com/doc/en/Date_and_time_functions.html.*

How Do I Convert a Date from U.S. Format to MySQL?

To convert a date from U.S. format to MySQL format, you can use the following function:

```
<?php
function date2mysql($date){
$splitArray = explode("/",$date);
$newDate = $splitArray[2] . "-" . $splitArray[0] . "-" . $splitArray[1];
return $newDate;
}
?>
```

This function takes a date in U.S. format and converts it to the MySQL format. To call this function, you just pass it the date in U.S. format, as follows:

```
<?php echo date2mysql("02/20/2003"); ?>
```

This will output the following:

```
2003-02-20
```

This can then be entered into a MySQL `Date` field. Note that to cater for other date formats, you can just rearrange the date parts into a different order.

How Do I Convert a Date from MySQL Format to U.S. Format?

To convert a date from MySQL format to U.S. date format, you can use the following function:

```
<?php
function mysql2date($date){
$splitArray = explode("-",$date);
$newDate = $splitArray[1] . "/" . $splitArray[2] . "/" . $splitArray[0];
return $newDate;
}
?>
```

To use this function, you call it by passing the date in MySQL format as follows:

```
<?php echo mysql2date("2003-02-20"); ?>
```

This will output the following:

```
02/20/2003
```

To use the function with a value from a recordset, simply use the field from the recordset as the parameter for the function, for example:

```
<?php echo mysql2date($row_Recordset1['date']); ?>
```

This would convert a database field named `date` in the recordset named `Recordset1` from MySQL format to the U.S. date format.

How Do I Find the Length of Time Between Two Dates?

In order to find the length of time between two dates, you need to convert both dates to a common format, and then you can take one date away from the other. The common format you use is the number of seconds since the Linux epoch date, which is **1/1/1970**. To convert a date to the common format, you use the PHP `mktime` function, which takes the following parameters and returns the number of seconds since 1/1/1970:

```
mktime(hour, minute, second, month, day, year);
```

You use the `mktime` function to convert the start date and end date into seconds, take the start date away from the end date, and then divide by the number of seconds in a day. You can do this with the following code:

```
<?php
$startDate = "02/20/2003"; $endDate = "02/28/2003";
$splitStart = explode("/",$startDate);
$splitEnd = explode("/", $endDate);
$start = mktime(0,0,0,$splitStart[0],$splitStart[1],$splitStart[2]);
$end = mktime(0,0,0,$splitEnd[0],$splitEnd[1], $splitEnd[2]);
$secondsInDay = 60 * 60 * 24;
$days = abs($end-$start)/$secondsInDay;
echo $days;
?>
```

When the preceding code is run, it will display 8, which is the number of days between `$startDate` and `$endDate`.

How Do I Find a Date, x Days/Months/Years in the Past or Future?

To find dates in the past or future, you can use a combination of the `mktime` and `date` functions, which allows you to easily add or subtract days, months, or years to/from the current date, as follows:

```
<?php
$tomorrow  = mktime (0,0,0,date("m")  ,date("d")+1,date("Y"));
$lastmonth = mktime (0,0,0,date("m")-1,date("d"),  date("Y"));
$nextyear  = mktime (0,0,0,date("m"),  date("d"),  date("Y")+1);
echo date("m/d/Y", $tomorrow) . "<br>";
echo date("m/d/Y", $lastmonth) . "<br>";
echo date("m/d/Y", $nextyear) . "<br>";
?>
```

Any date can be found by simply adding or subtracting the desired period to/from one of the date parts, as shown in the preceding example code.

Common PHP Errors

In this section, we are going to look at some of the common PHP errors that occur and how to solve them.

Parse Errors

A parse error occurs when the format of your PHP code is incorrect. For example, the following code:

```
<?php
for($i=1;$i<10;$i++){
$output = "Current Iteration: " . $i . "<br>"
echo $output;
}
?>
```

will return an error similar to the following:

Parse error: parse error, unexpected T_ECHO in c:\webserver\test2.php on line 4

This error message is shown because there is a missing semicolon at the end of the third line of the preceding example. The message tells you that PHP wasn't expecting the `echo` command on line 4. This is because it expected there to be a semicolon at the end of line 3. Most of the time when you get this error, the reason will be a problem with the previous line, such as the missing semicolon as in the previous example.

> *Most of the time when you get this error, the reason will be a problem with the previous line.*

This error also occurs with braces, {}, for example with the following code:

```
<?php
for($i=1;$i<10;$i++){
$output = "Current Iteration: " . $i . "<br>";
if($output==5){
echo "This is the fifth iteration";
}
}
}
?>
```

which will return the error message

Parse error: parse error, unexpected '}' in c:\webserver\test2.php on line 8

This is caused by an extra closing brace, }, on line 8. Although it's simple to see the problem in the preceding short block of code, it can be much harder with complicated code containing many nested loops or `if` statements, and it's hard to match the opening braces to the closing braces.

To quickly find the problem, you can use the Dreamweaver MX *Balance Braces* command, which is in the *Edit* menu. You can place the cursor at a line of code, select the *Balance Braces* command, and it will highlight all the lines of code for that block from the opening brace to the closing brace. By checking the layers of your code with the *Balance Braces* command, you can quickly find where there is a missing or extra brace.

Undefined Index or Variable

This section explains what to do if you receive a message on your web page like one of the following:

Warning: Undefined index: action in \home\www\login.php on line 25
Warning: Undefined variable: message in home\www\login.php on line 52

These messages frequently confuse people, as they look like error messages and can occur even with code that works perfectly. The messages aren't actually errors; instead, they are classed as "notices," which alert you to situations where there isn't actually an error that would stop the code from working. What the message tells you is that the variable mentioned hasn't been specifically defined using a PHP `var` statement.

As an example, look at the following two blocks of code:

```
<?php
var $username;
$username = $HTTP_SESSION_VARS['username'];
?>
```

and:

```
<?php
$username = $HTTP_SESSION_VARS['username'];
?>
```

Both blocks of code have identical results, but the second block of code will bring up a message similar to the ones shown previously, because we haven't specifically defined the variable `$username`. PHP works by defining your variables first, which means it can alert you if you misspell a variable in your code, as the variable will be new and not previously defined.

There are two solutions to this problem. The first is to go back through your code and make sure that every variable is specifically defined. The second option is to stop notices from being displayed, as they're often more trouble than they're worth. You can do this by adding the following code to the top of each page that has the error:

```
<?php error_reporting (E_ALL &~ E_NOTICE); ?>
```

This tells PHP to show all error messages and warnings but not to show the notices. To do this permanently, change the `error_reporting` setting in your `php.ini` file to the preceding setting.

Headers Already Sent Errors

This is another error message that occurs frequently, especially when using cookies or sessions, and it is similar to the following message:

Cannot modify header information - headers already sent by (output started at /home/web/newtest.php:3) in /home/web/test.php on line 5

The following is a screen shot of the code that caused the error:

```
1 <?php session_start; ?>
2
3 <?php
4 if($HTTP_SESSION_VARS['username'] == "fred"){
5   header("Location: test.php");
6 }
7 ?>
```

The problem is caused by the blank spaces before the `header` function. The PHP `header` function is used to redirect the user to another page, which it does by sending commands in the page header to the browser, telling it to go to the new page. However, once a header has been sent to the browser, you can no longer use functions that access the header, such as the `header` function. If anything is sent to the browser that isn't a special header function, the header is closed and everything else is assumed to be HTML for the web page. So why is this happening with the preceding code?

If anything is sent to the browser that isn't a special header function, the header is closed and everything else is assumed to be HTML for the web page.

The problem occurs because there is a blank line (line 2) that is sent to the browser. This causes the header to be closed, so the `header` function fails, as it can no longer write to the header. However, even if you remove the blank line at line 2, you will get the same error message. This is because there are two spaces at the end of the `?>` tag on line 1. These must also be deleted, as again they will be sent to the browser and will close the header. These spaces at the end of lines can be hard to spot because they're invisible!

There is a method you can use, though, to make finding them easier. If you place your cursor at the end of each line and left-click and move the mouse right, any spaces at the end of the line will be highlighted, as shown in the previous screen shot. The spaces can then be removed, leaving the code looking like the following screen shot:

```
1 <?php session_start; ?>
2 <?php
3 if($HTTP_SESSION_VARS['username'] == "fred"){
4   header("Location: test.php");
5 }
6 ?>
```

If the code is now tested again, it should work correctly.

This error can also occur if there is output not enclosed in PHP tags in any include files for the page. So if your main page code looks fine and you're still getting the error, the next step will be to check through any include files the page uses. Often, the error is caused by a new line after the final PHP tag in the include file, which must be removed if present.

How Do I Force a File to Download Rather Than Open?

When a user clicks a link to download a file, and the browser recognizes the file type, it may open the file instead of downloading it. For example, if the user has Adobe Acrobat Reader installed, when the user clicks on a link to download a PDF file, the file will open in Adobe Acrobat instead of being downloaded and saved to a specified location.

Although this behavior is quite useful, there are times when you want to allow the user to download the file instead of opening it. To do this, create a new PHP page and remove all the default HTML so the page is blank. You can then add the following code:

```
<?php
$filepath = "/home/www/webroot/downloads/manual.pdf";
$filename = "manual.pdf";
$size=filesize($filepath);
header("Content-Type: application/octet-stream");
header("Content-Length: " . $size);
header("Content-Disposition: attachment; filename=" . $filename);
header ("Content-Transfer-Encoding: binary");
$fh = fopen($filepath, "r");
fpassthru($fh);
?>
```

The preceding code is used to tell the user's browser to accept the file for downloading instead of opening. The user can now link to this page with a normal link, and once it's clicked, the file specified will be downloaded and the user will stay on the same page of your site.

How Do I Find the Dimensions of an Image?

When you're working with images whose paths are stored in a database, it's useful to be able to read the width and height straight from an image itself, rather than storing its dimensions in the database after asking the user for them.

To read the dimensions directly from an image (in any of the common web formats such as GIF, JPEG, PNG, and so on), you can use the PHP `getimagesize` function, to which you pass the path to your image. This function can read from all the common graphic file formats, and it returns an array where the first element (0) is the image's width and the second element (1) is the image's height. The following code shows an example of how to use the `getimagesize` function:

```
<?php
$image = "images/image.gif";
$imagepath = $_SERVER['DOCUMENT_ROOT'] . "/" . $image;
$size = getimagesize($imagepath);
echo "Width: " . $size[0] . "<br>";
echo "Height: " . $size[1] . "<br>";
echo "<img src='" . $image . "' width='" . $size[0] . "' height='" .
$size[1] . "'>";
?>
```

This code prints the width and height of the image, and it also inserts the width and height into an HTML `<img>` tag to display the image correctly.

How Do I Create a Random Password?

In some applications, it can be useful to generate a random password, such as setting the user's initial password and then letting the user change it if he or she wishes. The function shown in the following code generates a password with a number of random characters, and you can set the length of the password when you call the function:

```
<?php
function randomPassword($length) {
    $possibleCharacters =
"abcdefghijklmnopqrstuvwxyz1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ";
$characterLength = strlen($possibleCharacters);
$seed = (double) microtime() * 1000000;
srand($seed);
$password = "";
for($i=1;$i<=$length;$i++){
$character = rand(1,$characterLength);
$character = substr($possibleCharacters,$character, 1);
$password .= $character;
}
return $password;
}
?>
```

To call the function, you can use the following code:

```
<?php
$randomPassword = randomPassword(8);
echo $randomPassword;
?>
```

This would create a random eight-character password—for example, `VcF1TG65` or `BJuNj7ao`.

To create the password, first you set up a string containing characters that can be used for the password. You then find the number of characters in the string with the PHP `strlen` function.

Next, you seed the PHP random number generator and use it to create a random number between 1 and the number of characters that can be used. The seed is first created by using the PHP `microtime` function to return the number of seconds since the Linux epoch date, and you then multiply this by 1 million so that you have a whole number, which will be different each time the code is run. Next, you use this number as a seed for the random number generator using the PHP `srand` command, so that it generates truly random numbers. Note that with PHP version 4.2.0 and upward, you don't have to seed the random number generator as it's done automatically.

This random number is then used to extract a single character from the string of allowed characters by reading the character at the position in the string equal to the random number. This character is then added to the new password, until the number of characters in the password matches the number specified when the function was called.

Summary

In this chapter, we looked at numerous frequently asked PHP questions covering a range of subjects. For each question, we presented a solution that included working code to use as an example and as a base for your own PHP projects. Some of the main topics we covered included session variables, uploading files, date and time problems, and common PHP errors.