File Uploading Using Forms 28

Introduction

Forms allow users to enter data via form fields such as the text and password fields. We introduced forms in Chapter 18. One thing that we omitted from that chapter was a field type to allow a user to submit an entire file along with the form. The form input type "file" enables users to browse your local computer directory and select a file to upload to the Web server.

Once uploaded, a PHP script can be written to access this file, move it, store it and, as we shall see, use it for the user's benefit.

Form Input Type File

To create a form which allows us to upload a file we must insert the attribute "enctype" into the form element with the value "multipart/form-data". This informs the server that the form may contain an uploaded file:

```
<form enctype='multipart/form-data' action='form.php'
method='post'>
```

The form input element can be assigned the attribute type="file" to create a form input field in which we can browse for and select a file on the local machine to upload to the server. The format of this element is as follows:

```
<input name='myFileName' type='file'>
```

In addition, the maximum file size in bytes can be limited using a hidden field in which the system variable MAX_FILE_SIZE is set to a certain value. An example of this is:

```
<input type='hidden' name='MAX_FILE_SIZE' value='100000'>
```

An example of a form with a file upload field is:

```
<?php

// File Upload - Example 28-1
//---------------------

?>

<form enctype='multipart/form-data' action='<?php echo
$_SERVER["PHP_SELF"];
?>' method='post'>

<input type='hidden' name='MAX_FILE_SIZE' value='100000'>
Document File: <input name='myfile' type='file'
value='$file'><br><br>
<input type='submit' value='Submit the Document' name='send'>
</form>
```

Figure 28.1 illustrates what this form looks like when viewed in a browser.

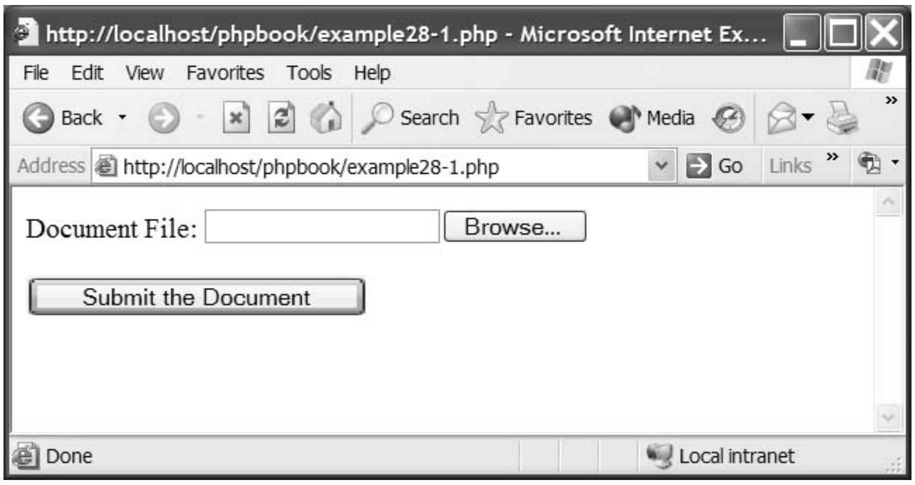

http://localhost/phpbook/example28-1.php - Microsoft Internet Ex...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example28-1.php Go Links
Document File:
Browse...
Submit the Document
Done
Local intranet

Figure 28.1 Simple file upload form.

You can enter the name and location of the file to upload in the form field or you can click the Browse button to open a "choose a file window" and locate and select the file to upload. This window is shown in Figure 28.2.

Unfortunately, while we have created our form, we have not yet written any PHP code with which to access it.

Accessing an Uploaded File with PHP

There are a number of steps, which need to be followed in writing a PHP script to access an uploaded file. We need to be careful to do this correctly as there are many security risks

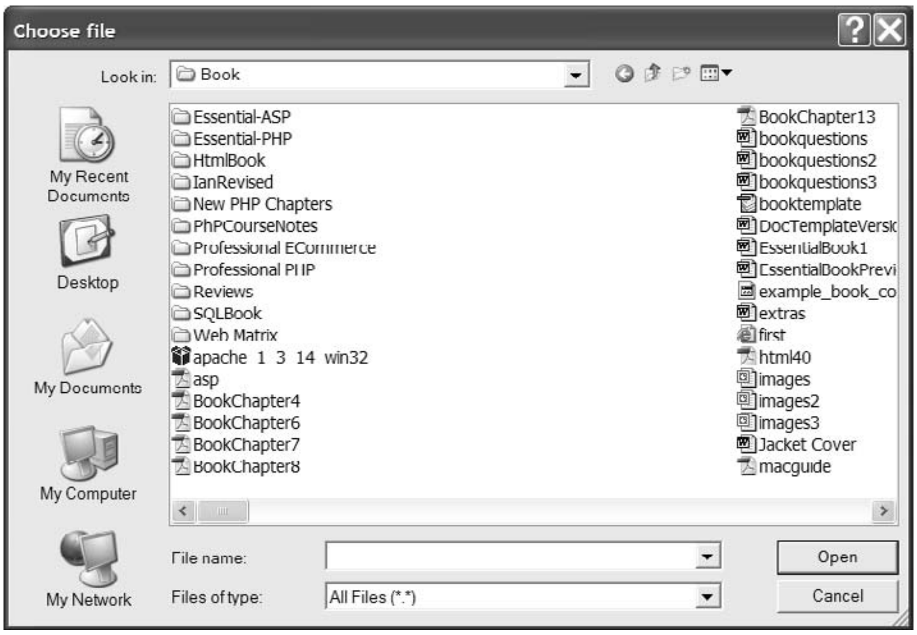


Figure 28.2 Choose a file window.

if you do not. Before we introduce these steps let's examine how file upload in PHP works. This is illustrated in Figure 28.3.

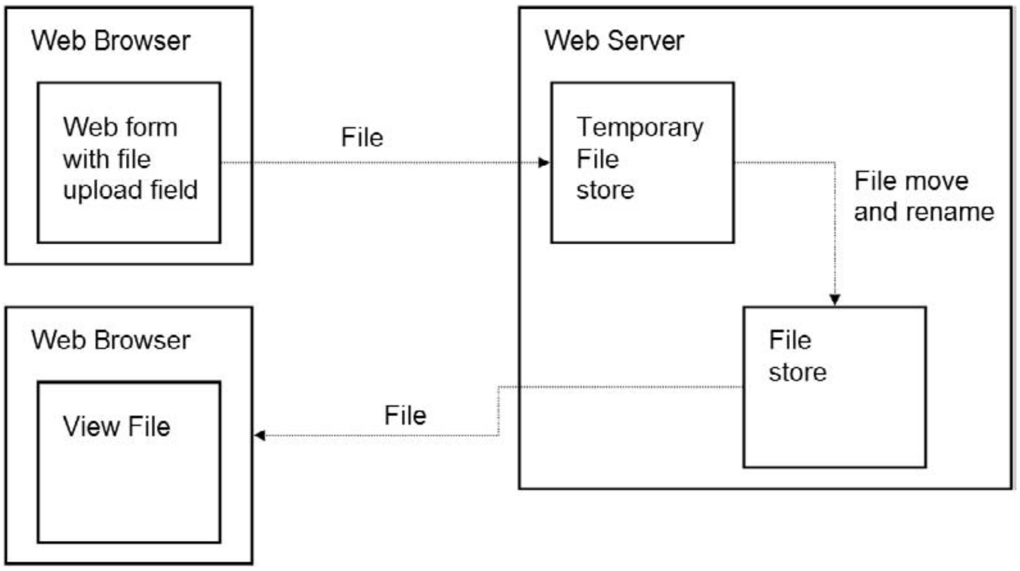


Figure 28.3 How file upload in PHP works.

When a file is uploaded via a form the file is stored in a temporary directory as shown in Figure 28.3. This is specified by the developer and will depend very much on your computer's operating system. The file is stored using a temporary name created by PHP. A couple of environment variables are created which enables the temporary name of the file and the real name of the file to be stored. PHP contains some functions which enable you to determine whether the temporary file was indeed uploaded via a PHP form and, if so, will allow you to move the file to its final resting place with its original name. These functions provide a degree of security from would-be invaders.

Let's examine the different PHP instructions which allow us to access uploaded files. First, the temporary directory into which uploaded files are first placed is set with the $TMPDIR environment variable, for example:

```
$TMPDIR = "c:\\Temp\\uploads";
```

This example sets the temporary directory to a c:\temp\uploads. Note the use of the \\ escape sequence to allow us to specify a \ character. When the file is uploaded, the name assigned to the form field (in the previous example it was myfile) is used to store the temporary and real names of the files in an environment variable array called $HTTP_POST_FILES. The temporary file name and real file name can be accessed like this:

```
$filename = $HTTP_POST_FILES['myfile']['tmp_name'];
$realname =$HTTP_POST_FILES['myfile']['name'];
```

It is important to access this information as we need to use it to check whether this is a genuine file, which was uploaded correctly. To do this we use the is_uploaded_file() function:

Function prototype:

```
bool is_uploaded_file(string filename);
```

Function arguments and return details:

Name	Type	Description
filename	String	The name of the file to check has been uploaded
is_uploaded_file() returns	Bool	A value of "TRUE" if the file is a valid uploaded file, otherwise it returns "FALSE"

Function example:

```
$res = Is-Iploaded_file($filename);
```

If the file is genuine, then it can be moved and renamed using the move_uploaded_file() function:

Function prototype:

```
bool move_uploaded_file(string filename, string destination);
```

Function arguments and return details:

Name	Type	Description
filename	String	The temporary name of the uploaded file
destination	String	The name and location to which to move the file
move_uploaded_file() returns	Bool	A value of "TRUE" if the file is moved successfully, otherwise it returns "FALSE"

Function example:

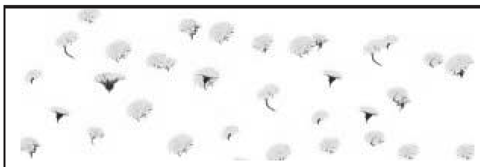
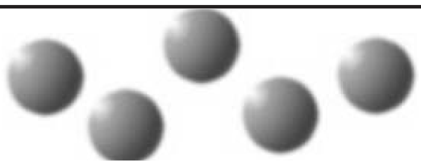
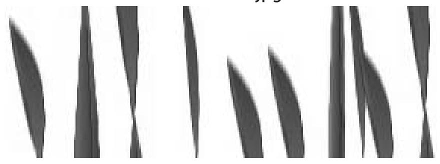
```
move_uploaded_file($filename,("c:\\Program
Files\\httpd\\HtDocs\\bigbook\\uploads\\".$newname));
```

We now have enough information to create a simple file upload program.

A Simple File Upload

For this example we began by creating four images which we stored in c:\temp\pics. These pictures were saved as rope.jpg, bluedots.jpg, dandelions.jpg and grass.jpg and are shown in Table 28.1.

Table 28.1 Our file images and names.

 Dandelions.jpg	 Bluedots.jpg
 Grass.jpg	Rope.jpg

Next, a simple file upload form was created allowing two files to be uploaded. These were given the names picture1 and picture2; the script is:

```
<?php

// File Upload - Example 28-2
//---------------------

?>
```

```
<form enctype='multipart/form-data' action='example28-3.php'
method='post'>
<input type='hidden' name='MAX_FILE_SIZE' value='1000000'>
First Picture: <input name='picture1' type='file'><br>
Second Picture: <input name='picture2' type='file'><br><br>
<input type='submit' value='Submit the Pictures' name='send'>
</form>
```

This form is shown in Figure 28.4. The form was designed to upload two of the four images created previously, although no error checking is made to ensure that the uploaded files are, in fact, images.

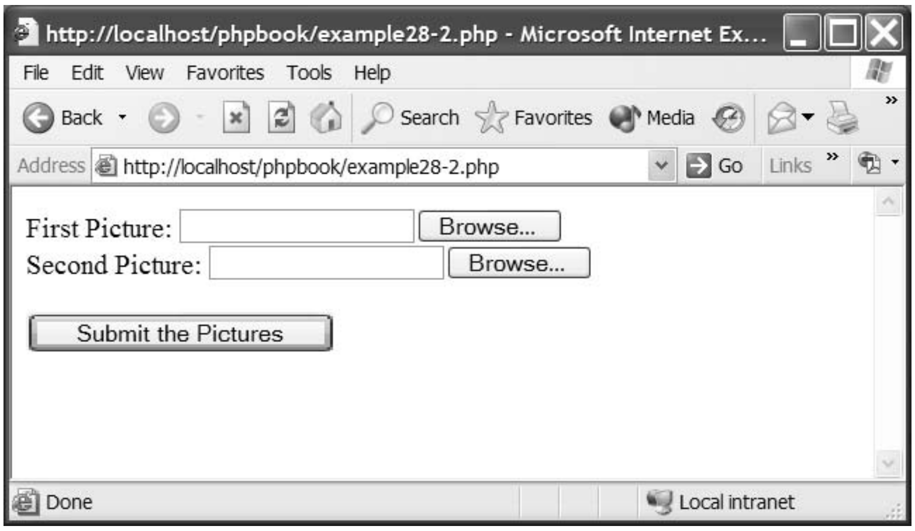

http://localhost/phpbook/example28-2.php - Microsoft Internet Ex...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example28-2.php Go Links
First Picture: Browse...
Second Picture: Browse...
Submit the Pictures
Done
Local intranet

Figure 28.4 Picture upload form.

Next, the PHP script to handle the uploaded files needs to be created:

```
<?php

// File Upload - Example 28-3
//---------------------

$TMPDIR = "c:\\Temp\\uploads";
$filename = $HTTP_POST_FILES['picture1']['tmp_name'];
if (is_uploaded_file($filename))
      move_uploaded_file($filename,
      "c:\\Inetpub\\wwwroot\\phpbook\\graphics\\picture1.jpg");
$filename = $HTTP_POST_FILES['picture2']['tmp_name'];
if (is_uploaded_file($filename))
      move_uploaded_file($filename,
      "c:\\Inetpub\\wwwroot\\phpbook\\graphics\\picture2.jpg");
```

```
?>
<img src="graphics\picture1.jpg">
<br>
<img src="graphics\picture2.jpg">
<br>
<br>
<a href="example28-2.php">Back</a> to form upload.
```

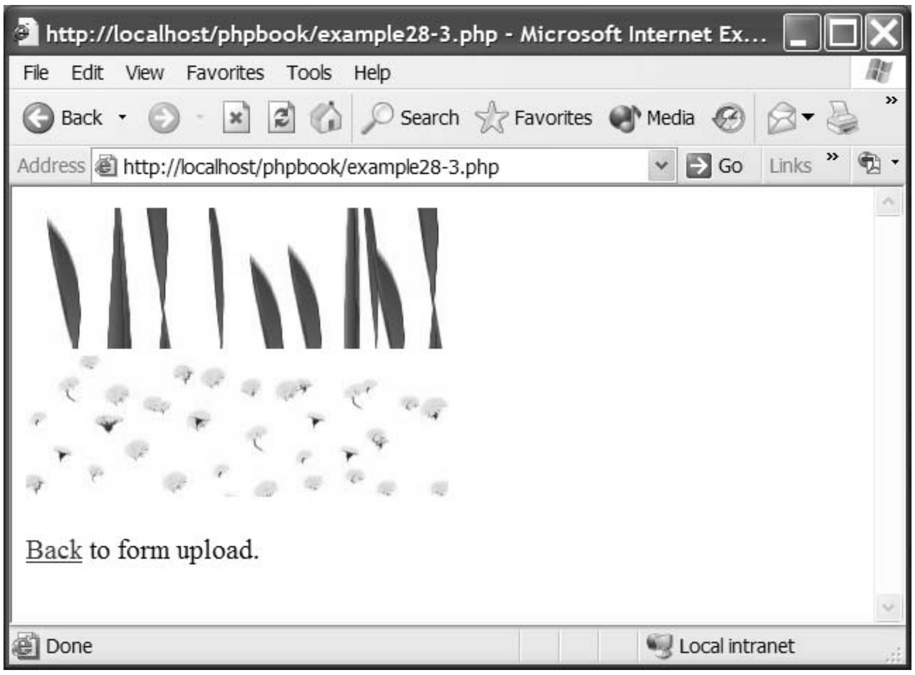
This script checks that the first file (picture1) is a valid file and moves this to

```
"c:\\Inetpub\\wwwroot\\phpbook\\graphics\\picture1.jpg");
```

Then the second file (picture2) is checked and moved to

```
"c:\\Inetpub\\wwwroot\\phpbook\\graphics\\picture2.jpg");
```

Finally, the script displays the two images using the <IMG SRC> element and provides a hyperlink back to the file upload form. An example output from the script is shown in Figure 28.5. Note that the images displayed will depend on which images you choose to upload.


http://localhost/phpbook/example28-3.php - Microsoft Internet Ex...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example28-3.php Go Links
Done
Local intranet

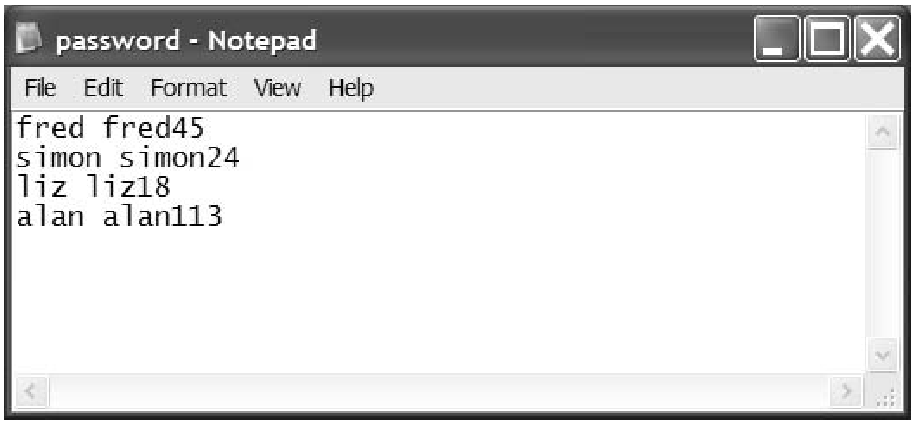
Back to form upload.

Figure 28.5 Picture upload display.

Securing File Uploads

While our previous file upload example illustrates how the concept of file uploading and manipulation works, it also highlights a major issue of file uploading – that of security. In our example no check is made on who is uploading images and thus anyone who knows the address of the upload form page can upload an image and replace those previously installed. While in this example this would be nothing more than an annoyance, if you were uploading documents that you only wanted certain people to be able to upload then a more secure environment is required. Let's begin to investigate how we would do this.

The first thing we need to consider is that we need to create a mechanism which will allow us to identify who is trying to upload a document and determine if you wish them to be allowed to do so. A simple username and password system would suffice. To create a system we first create a simple username and password file where each username and password is listed on separate lines of the file using the Notepad text editor. Each username and password is separated with a space character. An example of such a file is illustrated in Figure 28.6 and can be saved as a file called password.txt. Make sure you don't save this in your Web-server directory or potential hackers will be able to access this and view your list of passwords and usernames.



```
fred fred45
simon simon24
liz liz18
alan alan113
```

Figure 28.6 Username and password file.

We are now ready to create the file submit application and here it is:

```
<?php

// File Upload - Example 28-4
//---------------------

if (isset($_POST['username']))
  $username = $_POST['username'];
else
  $username = "";
if (isset($_POST['password']))
  $password = $_POST['password'];
```

```
else
  $password = "";
if (isset($_POST['file']))
  $file = $_POST['file'];
else
  $file = "";

$passwordOkay = 0;

if (isset($_POST['send'])) {
  $okay =
checkUsernamePassword("c:\\Temp\\password.txt",$username,$password);
  if($okay) {
    $passwordOkay = 1;
    $TMPDIR = "c:\\Temp\\uploads";
    $filename = $HTTP_POST_FILES['myfile']['tmp_name'];
    $realname = $HTTP_POST_FILES['myfile']['name'];
    if (is_uploaded_file($filename)) {
      $date = fDate();
      $time = fTime();
      $newname = ($realname . "-" . $date . "-" . $time);

  move_uploaded_file($filename,
  ("c:\\Inetpub\\wwwroot\\phpbook\\uploads\\".
$newname));
      appendToFile("uploads\\uploads.txt", $date . " "
. $time . " " . $username . " " . $realname);
      echo("Thank you document submitted. Click <a
href='example28-5.php'>here</a> to see a list of submitted
documents.");
    }
  }
  else
    echo("Incorrect Password and/or Username.");
}

$phpself = $_SERVER['PHP_SELF'];

if(!isset($_POST['send']) || $passwordOkay == 0) {
  echo("<h2>Welcome to the Documents Submission Page</h2>");
  echo("Please complete the form and press the send file
button.<P>");
  echo("<form enctype='multipart/form-data' action='$phpself'
method='post'>");
  echo("<input type='hidden' name='MAX_FILE_SIZE' value='100000'>");
  echo("Username: <input type='text' name='username'
value='$username'><br>");
  echo("Password: <input type='password' name='password'
value='$password'><br>");
```

```
  echo("Document File: <input name='myfile' type='file'
value='$file'><br><br>");
  echo("<input type='submit' value='Submit the Document'
name='send'>");
  echo("</form>");
}

function fDate() {
  $date = getdate();
  $monthText = $date["month"];
  $year = $date["year"];
  $mday = $date["mday"];
  return $mday . "-" . $monthText . "-" . $year;
}

function fTime() {
  $time = localtime();
  return $time[2] . "-" . $time[1] . "-" . $time[0];
}

function appendToFile($file,$data) {
  $out = fopen($file,"a");
  fputs($out,$data."\n");
  fclose($out);
}

function checkUsernamePassword($file,$username,$password) {
  $found=0;
  $in = fopen($file,"r");
  $line = fgets($in,4096);
  while(!feof($in) && !$found) {
    $splitLine = explode (" ", $line);
    $splitLine[1] = substr($splitLine[1],0,strlen($splitLine[1])-2);
    if($splitLine[0] == $username && $splitLine[1] == $password)
      $found=1;
    $line = fgets($in,4096);
  }
  fclose($in);
return $found;
}
?>
```

It is a little longer than our first example – but don't worry we shall examine this a little at a time explaining what is going on. The first part of the script obtains the form variables:

```
if (isset($_POST['username']))
      $username = $_POST['username'];
```

```
else
      $username = "";
if (isset($_POST['password']))
      $password = $_POST['password'];
else
      $password = "";
if (isset($_POST['file']))
      $file = $_POST['file'];
else
      $file = "";

$passwordOkay = 0;
```

The next part of the script checks that the form data have been sent and, if so, invokes the function checkUsernamePassword(), passing it the parameters password text file location, username and password:

```
if (isset($_POST['send'])) {
      $okay =
checkUsernamePassword("c:\\Temp\\password.txt",$username,$password);
```

The checkUsernamePassword() function returns a "0" if the username/password does not exist or "1" if all is okay. We shall return to this function later when we encounter its definition further down the listing. If the password and username are okay, then the next part of the script sets the temporary directory and accesses the temporary and real names of the uploaded file. It then checks that the file is a valid file upload by calling the is_uploaded_file() function:

```
      if($okay) {
            $passwordOkay = 1;
            $TMPDIR = "c:\\Temp\\uploads";
            $filename = $HTTP_POST_FILES['myfile']['tmp_name'];
            $realname = $HTTP_POST_FILES['myfile']['name'];
            if (is_uploaded_file($filename)) {
```

If the file is a valid upload, then the date and time are obtained by invoking the functions fDate() and fTime(). The date and time returned from these functions are used to create a new file name which encorporates the real file name along with the date and time:

```
              $date = fDate();
              $time = fTime();
              $newname = ($realname . "-" . $date . "-" . $time);
```

The file is then moved to a directory that we have created called uploads. You can change the location of this directory if you wish.

```
    move_uploaded_file($filename,
("c:$\\Inetpub\\wwwroot\\phpbook\\uploads\\".
$newname));
```

The function appendToFile() is invoked, passing the parameters of the file name to write to and the date, time, username and realname of the file as a single string. Finally, a message confirming the submission of the document and asking the user if they wish to view the submitted documents is displayed:

```
appendToFile("uploads\\uploads.txt", $date . " " . $time . " " .
$username . " " . $realname);
                echo("Thank you document submitted. Click <a
href='example28-5.php'>here</a> to see a list of
submitted documents.");
    }
```

If the username or password is incorrect, a message is displayed to that effect:

```
   else
         echo("Incorrect Password and/or Username.");
```

If the form was not submitted or the password/username was incorrect, then the form is displayed. This consists of fields which allow the user to enter the username, password and the document to upload:

```
$phpself = $_SERVER['PHP_SELF'];

if(!isset($_POST['send']) || $passwordOkay == 0) {
   echo("<h2>Welcome to the Documents Submission Page</h2>");
   echo("Please complete the form and press the send
file button.<P>");
   echo("<form enctype='multipart/form-data' action='$phpself'
method='post'>");
   echo("<input type='hidden' name='MAX_FILE_SIZE' value='100000'>");
   echo("Username: <input type='text' name='username'
value='$username'><br>");
   echo("Password: <input type='password' name='password'
value='$password'><br>");
   echo("Document File: <input name='myfile' type='file'
value='$file'><br><br>");
   echo("<input type='submit' value='Submit the Document'
name='send'>");
   echo("</form>");
}
```

The remainder of the program consists of the user-defined functions, which we have invoked previously. The first two of these are the fDate() and fTime() functions which return the date and time as strings:

```
function fDate() {
      $date = getdate();
      $monthText = $date["month"];
      $year = $date["year"];
      $mday = $date["mday"];
      return $mday . "-" . $monthText . "-" . $year;
}

function fTime() {
      $time = localtime();
      return $time[2] . "-" . $time[1] . "-" . $time[0];
}
```

The appendToFile() function writes a record of the document, which was just uploaded to the uploads.txt file. This text file is used by our second PHP program to display the documents which have been uploaded:

```
function appendToFile($file,$data) {
      $out = fopen($file,"a");
      fputs($out,$data."\n");
      fclose($out);
}
```

The final part of the program is the checkUsernamePassword() function. This function opens the passwords.txt file and reads each line of the file:

```
function checkUsernamePassword($file,$username,$password) {
    $found=0;
    $in = fopen($file,"r");
    $line = fgets($in,4096);
    while(!feof($in) && !$found) {
```

Each line of the file is split apart using the explode function to obtain the stored username and password:

```
    $splitLine = explode (" ", $line);
```

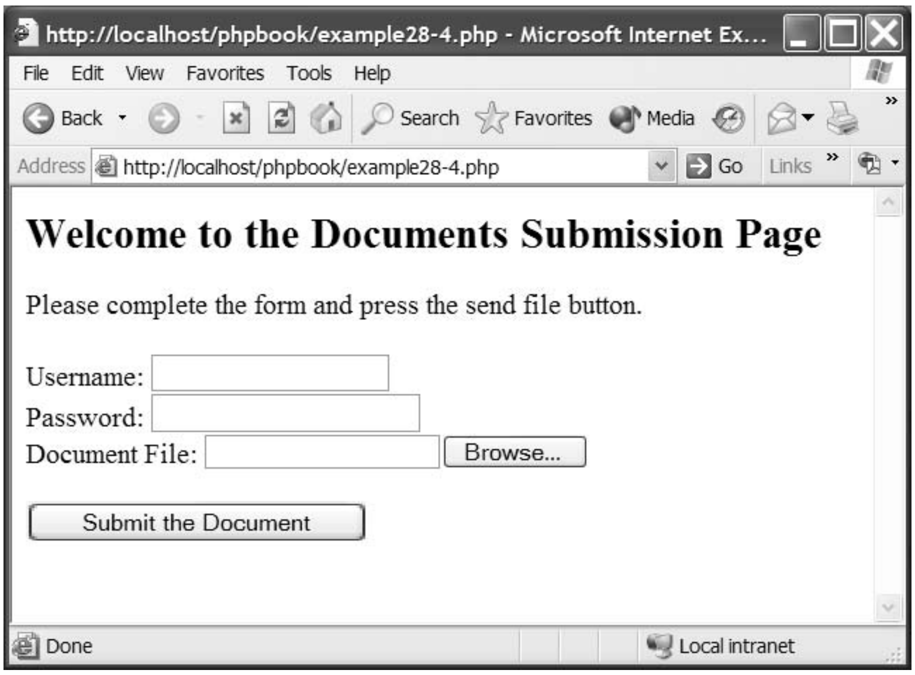
As a text file often has some line-feed characters inserted at the end of each line, these are removed using the substr() function which deletes the last two hidden characters from the string:

```
        $splitLine[1] = substr($splitLine[1],0,strlen($splitLine[1])-2);
```

The username and passwords passed to the function are compared with those read from the file and, if they match, the variable $found is set to "1" and this is returned by the function:

```
            if($splitLine[0] == $username && $splitLine[1] == $password)
                $found=1;
            $line = fgets($in,4096);
        }
        fclose($in);
    return $found;
    }
```

The output produced by this program is shown in Figure 28.7.


http://localhost/phpbook/example28-4.php - Microsoft Internet Ex...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example28-4.php Go Links
Done Local intranet

Welcome to the Documents Submission Page

Please complete the form and press the send file button.

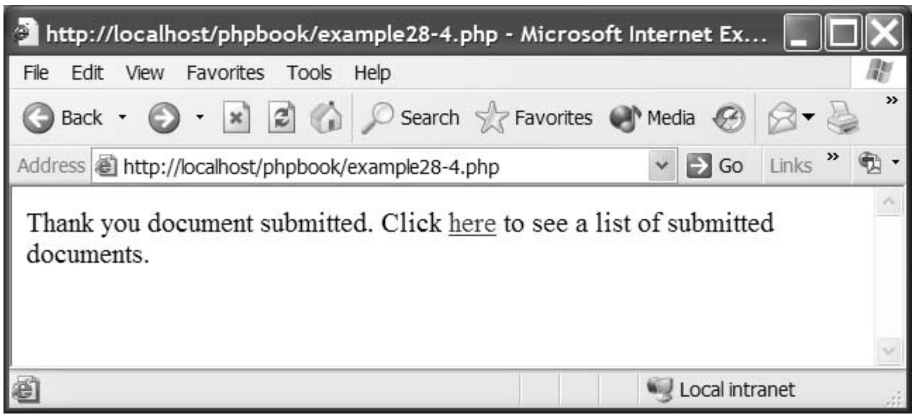
Username:
Password:
Document File: Browse...

Submit the Document

Figure 28.7 Secure file upload screen.

If a correct username and password are entered and a file successfully uploaded then the message illustrated in Figure 28.8 is shown.

The next stage is to create the program, which will display the documents uploaded when the user selects the here hyperlink. The program simply opens the uploads.txt file and reads each line of the file. The details of each uploaded file is displayed within a table:


http://localhost/phpbook/example28-4.php - Microsoft Internet Ex...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example28-4.php Go Links
Local intranet

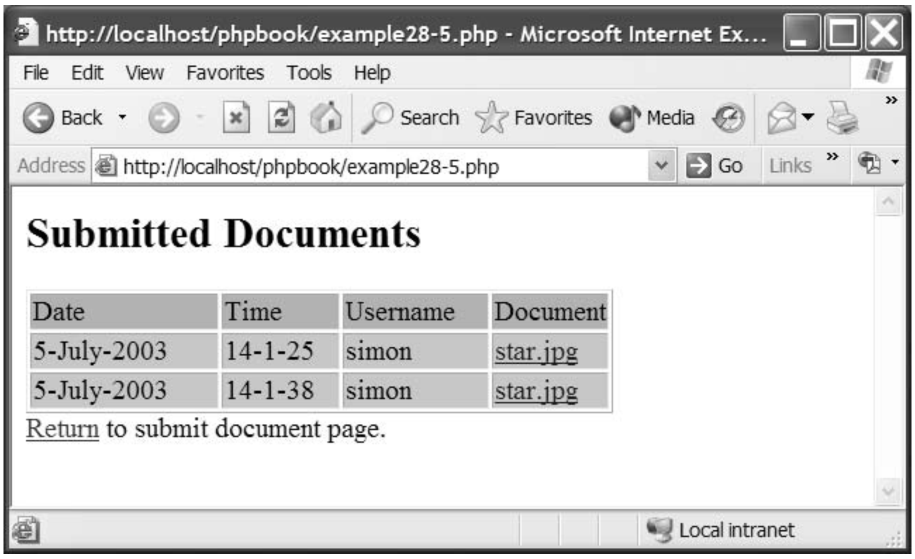
Thank you document submitted. Click here to see a list of submitted documents.

Figure 28.8 Successful upload.

```
<h2>Submitted Documents</h2>
<table border=1>
<tr bgcolor='cyan'><td width=130>Date</td><td width=80>Time</td><td
width=100>Username</td><td>Document</td></tr>

<?php

// File Upload - Example 28-5
//---------------------

$in = fopen("uploads//uploads.txt","r");
$line = fgets($in,4096);
while(!feof($in)) {
   $splitLine = explode (" ", $line);
   echo("<tr bgcolor='lightgreen'><td>" . $splitLine[0] .
    "</td><td>" .
$splitLine[1] . "</td><td>" . $splitLine[2] .
   "</td><td><a href='uploads\\" . $splitLine[3] . "-" .
$splitLine[0] . "-" . $splitLine[1] . "'>" . $splitLine[3] .
"</a></td></tr>");
   $line = fgets($in,4096);
}
fclose($in);
?>
</table>
<a href='example28-4.php'>Return</a> to submit document page.
```

An example output from this program is shown in Figure 28.9.

The user can access the uploaded files by clicking on the name of the document. Depending on the type of document this will either, as in the case of an image file, be


http://localhost/phpbook/example28-5.php - Microsoft Internet Ex...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/phpbook/example28-5.php Go Links
Local intranet

Submitted Documents

Date	Time	Username	Document
5-July-2003	14-1-25	simon	star.jpg
5-July-2003	14-1-38	simon	star.jpg

Return to submit document page.

Figure 28.9 Uploaded documents.

displayed or, if a document, the user may be prompted as to whether they wish to download a copy of the document to their local computer.

Summary

This chapter has introduced the concept of file upload via a form. We have shown how this can be achieved and the use to which such a facility can be put. We have explained that there are security issues in allowing file upload and have discussed how PHP ensures that the uploaded files are genuine, in addition to what you, the programmer, must do to restrict user access. In the next chapter we shall examine what PHP can accomplish by manipulating simple images.