**MORE**

**PHP & MySQL**

**Common programming techniques**

**Sending values to a script**

There are 2 different ways, other than user-submitted data, to pass variables and values to a script:

1. Using HTMLs hidden input type: *<input type=”hidden” name=”do” value=”this”/>.* As long as this code is anywhere between the form tags, the variable $\_POST[‘do’] will have a value of ‘this’ in the handling PHP script, assuming that the form uses the POST method. $\_GET[‘do’] will have the value if the method of the form is GET.
2. You could also skip the creation of a form and just directly append a name=value pair to the url: [*www.example.com/page,php?do=this*](http://www.example.com/page,php?do=this)

Example (full example on p300-303):

‘<a href=”edit\_user.php?id-‘” .$row[‘user\_id’] .’”>Edit</a>

* Assuming you was connected to MySQL and ran a SELECT query, this would send the user\_id value to the edit\_user.php script

To append multiple variables to a URL, use: *page.php?name1=value1&name2=value2&name3=value3 …*

**Using hidden form inputs**

<input type=”hidden” name=”id” value=”’ .$id. ‘”/>

* Assuming the id value came from the ‘sending values to a script’ example. This id would now be accessible in the handling script using a hidden form input – if the handling php script was another form

\*See full example on p304-308\*

Make sure to still use conditionals to ensure the correct value has been sent:

If(isset($\_POST(‘id’)) && is\_numeric($\_POST[‘id’])) {

$id = $\_POST[‘id’];

}else{

Echo “error”;

}

**Editing form inputs**

See example on p309-315

**Paginating query results**

Similar to search engines, results are displayed over numerous pages, not just as one long list. Paginating query results makes extensive use of the LIMIT SQL clause. To paginate the returned results of a query, each iteration of the page will run the same query using different LIMIT parameters. To make this work, 2 values must be passed from page to page in the URL.

\*SIDE NOTE – ternary operator: *(condition) ? valueTrue: valueFalse*\*

\*See full example on p316-322 and make sure this can be done\*

**Making sortable displays**

Sorting data makes use of switch statements to change the ORDER BY clause to display results in different orders (selected by users)

\*See full example on p323-327 and make sure this can be done\*

**Web application development**

**Sending Email**

Mail(to, subject, body, [headers]);

To = an email address or a series of addresses separated by commas

Subject = subject line

Body = body of message

Assign these as variables to make the mail() function more legible.

You can create an email message that goes over multiple lines using (\n) within double quotation marks (this can’t be used for the subject). It is advised not to allow any line to exceed 70 characters, this is easily achieved using the wordwrap() function:

Wordwrap($body, 70);

The 4th mail() argument allows for additional headers, e.g. from, reply-to, Cc, Bcc… To use multiple headers of different types in your email, separate each with: \r\n

It is advisable to always include a From value

$headers = “From: [rezaa91@hotmail.co.uk\r\n](mailto:rezaa91@hotmail.co.uk\r\n)”;

$headers ,= “Cc: [Jane@example.com](mailto:Jane@example.com), [joe@example.com\r\n](mailto:joe@example.com\r\n)”;

**Handling file uploads**

The process of uploading a file has 2 dimensions:

1. The HTML form must be displayed, with the proper code to allow for file uploads. Upon submission of the form, the server will first store the uploaded file in a temporary directory
2. The PHP script must then copy the uploaded file to its final destination

For this process to work, several things must be in place:

* PHP must run with the correct settings
* A temporary storage directory must exist with the correct permissions
* The final storage directory must exist with the correct permissions

You will need to edit the php.ini file if any of these conditions apply:

* File\_uploads is disabled
* PHP has no directory to use
* You will be uploading very large files (larger than 2mb)

If you are using a hosted site (where you don’t have access to php.ini), this should hopefully already be configured.

\*It is more secure to place the uploads folder outside the web directory. It would be easier to place the folder within the web directory, however, for PHP to be able to place files in the uploads folder, it needs to have write permissions on that directory. On most servers, PHP is running as the same user as the Web server itself. On a hosted server, this means that all X number of sites being hosted are running as the same user. Creating a folder that PHP can write to means creating a folder that everyone can write to. If you keep the uploads folder publicly accessible, you could limit access to the uploads folder using an .htaccess file (you would state only image files in the folder would be publicly viewable for example. See Appendix A for info on how to use .htaccess files). Basically store outside the web directory for more security – even though less convenient\*

1. Run phpinfo(), one of, if not the most important function in PHP, & search for the following settings:

|  |  |  |  |
| --- | --- | --- | --- |
| **Value:** | **Setting:** | **Value type:** | **Importance:** |
| On | File\_uploads | Boolean | Enables php for file uploads |
|  | Max\_input\_time | Integer | Indicates how long (s), a php script is allowed to run |
| Default = 8M | Post\_max\_size | Integer | Size, in bytes, of the total allowed POST data |
| Default = 2M | Upload\_max\_filesize | Integer | Size, in bytes, of the largest possible file upload allowed |
|  | Upload\_tmp\_dir | String | Indicates where uploaded files should be temporarily stored |

Ensure that the post\_max\_size and upload\_max\_filesize are large enough.

The upload\_tmp\_dir may need to be indicated in the php.ini file – e.g. set to: C:\tmp

1. If necessary, make changes in the php.ini file – save & restart the server
2. Confirm changes by rerunning the phpinfo() script
3. You may need to change permissions of the temp folder – right click > properties > security
4. Create a new directory (called uploads) in a directory outside the web directory (htdocs) – and set permissions so that the web server can write to it.

Uploading files with PHP

Once the server is set up, you can create the script that does the actual file handling. The required syntax for a form to handle a file upload has 3 parts:

1. <form enctype=”multipart/form-data” action=”script.php” method=”post”>
2. <input type=”hidden” name=”Max\_FILE\_SIZE” value=”30000”/>
3. <input type=”file” name=”upload”/>

‘enctype’ indicates that the form should be able to handle multiple types of data, including files.

Upon form submission, the uploaded file can be accessed using the $\_FILES superglobal. The variable will be an array of values:

|  |  |
| --- | --- |
| **Index** | **Meaning** |
| Name | The original name of the file (as it was on the users computer) |
| Type | The MIME type of the file, as provided by the browser |
| Size | The size (in bytes) |
| Tmp\_name | The temporary filename of the uploaded file as it was stored on the server |
| Error | The error code associated with any problem |

Once the file has been received by the PHP script, the move\_uploaded\_file() function can transfer it from the temporary directory to its permanent location:

Move\_uploaded\_file(temp\_filename, /path/to/destination/filename);

Example (full example p343-347):

If($\_SERVER[‘REQUEST\_METHOD’] == ‘POST’) {

If(isset($\_FILES[‘upload’])) {

//validate the type. Should be JPEG or PNG

$allowed = array(‘image/pjeg’, ‘image/jpeg’, ‘image/JPG’, ‘image/X-PNG’, ‘image/PNG’, ‘image/png’, ‘image/x-png’);

If(in\_array($\_FILES[‘upload’][‘type’], $allowed)) {

//move the file over

If(move\_uploaded\_file($\_FILES[‘upload’][‘tmp\_name’], “.../uploads/{$\_FILES[‘upload’][‘name’]}”)) {

Echo “The file has been uploaded”;

} // end of move IF

}else{ //invalid type

Echo “Please upload a JPEG or PNG image”;

}

} // end of isset($\_FILES[‘upload’]) IF

//check for an error

If($\_FILES[‘upload’][‘error’] > 0) {

Echo “There was an error”;

//see full example for switch statement showing actual errors

}

//delete the file if it still exists in the temp folder

If(file\_exists($\_FILES[‘upload’][‘tmp\_name’]) && is\_file($\_FILES[‘upload’][‘tmp\_name’]) ) {

Unlink($\_FILES[‘upload’][‘tmp\_name’]);

}

} // end of submitted conditional

//create HTML form

**Understanding HTTP headers**

A proxy script is able to provide access to content on the server that would otherwise be unavailable (e.g. outside web directory).

HTTP (Hypertext Transfer Protocol) is the technology at the heart off the World Wide Web and defines the way clients & servers communicate. When a browser requests a web page, it receives a series of HTTP headers in return – this happens behind the scenes.

PHP’s built in header() function can be used to take advantage of this protocol: *header(header string);* the list of possible header strings is quite long, as headers are used for everything from redirecting the web browser, to sending files, to creating cookies, to controlling page caching & more…

To redirect web browser:

Header(‘location: <http://www.example.com/page.php>’);

Note that if a script uses multiple header calls, each should be terminated by a newline (\n)

\*The header() function must be called before anything is sent to the web browser! E.g. html, echo functions…\*

Example (p358-361):

$name = FALSE; //flag variable

//check for an image name in the URL:

If(isset($\_GET[‘image’])) {

//make sure it has image’s extension

$ext = strtolower(substr($\_GET[‘image’], -4));

If(($ext == ‘.jpg’) OR ($ext == ‘jpeg’) OR ($ext == ‘.png’)) {

//full image path

$image = “../uploads/{$\_GET[‘image’]}”;

//check that the image exists and is a file:

If(file\_exists($image) && (is\_file($image))) {

$name = $\_GET[‘image’];

}

}//end of $ext IF

}//end of isset($\_GET[‘image’]) IF.

//if there was a problem, use the default image

If(!$name) {

$image = ‘images/unaviable.png’;

$name = ‘unavailable.png’;

}  
 //get the image information

$info = getimagesize($image);

$fs = filesize($image);

//send the content information

Header(“Content-type: {$info[‘mime’]}\n”);

Header(“Content-Disposition: inline; filename=\”$name\”\n”);

Header(“Content-Length: $fs\n”);

//send the file

Readfile($image);

\*This example works with the javascript with PHP example on p352-354\*

**See p362-365 for php date & time functions**

**TEST**

**Part 1**

1. What are the 2 ways to pass variables and values to a script?
2. Give an example using both
3. How do you attach multiple name and value pairs to a script?
4. Create a table of all users in a database and create scripts so the administrator is able to delete or edit user information
5. What does paginating query results refer to? Give an example
6. Allow the previously created table to be ordered by which ever column the user clicks on

**Part 2 (including HTTP)**

1. How do you send an email through PHP?
2. What does the wordwrap() function do?
3. How do you use multiple headers of different types in your email?
4. Create a form for a user to submit their information (including email) – to then receive an email (include a from and Cc header)
5. What is HTTP?
6. Give an example of the header() function to redirect the web browser
7. Follow the example under the HTTP heading above

**Part 3**

1. What function do you use to check configuration settings in PHP?
2. When handling file uploads, which of these settings do you need to check?
3. How do you edit these settings if they need adjusting?
4. What superglobal variable is used to access the uploaded file?
5. This variable is an array with 5 values – what are they?
6. How do you transfer the file from the temporary directory to its final destination?
7. Give example – upload a file and store it the final destination folder – make sure to only allow image files