

Chapter 23

How to work with files, uploads, and images



Murach's PHP and MySQL (3rd Ed.)

C23, Slide 1

Display a directory listing

```
$path = getcwd();
$item = scandir($path);
echo "<p>Contents of $path</p>";
echo '<ul>';
foreach ($item as $item) {
    echo '<li>' . $item . '</li>';
}
echo '</ul>';
```



Murach's PHP and MySQL (3rd Ed.)

C23, Slide 4

Objectives

Applied

1. Get data from and save data in files, including CSV files.
2. Upload and save files from users, including image files.
3. Create new images, resize images, and work with image transparency.

Knowledge

1. In general terms, describe the PHP functions for working with directories and files, including CSV files.
2. Describe the HTML for uploading a file and the PHP function for saving an uploaded file in a permanent directory.
3. In general terms, describe the PHP functions for working with images, resizing images, and working with image transparency.



Murach's PHP and MySQL (3rd Ed.)

C23, Slide 2

Display the files from a directory listing

```
$path = getcwd();
$item = scandir($path);
$files = array();
foreach ($item as $item) {
    $item_path = $path . DIRECTORY_SEPARATOR . $item;
    if (is_file($item_path)) {
        $files[] = $item;
    }
}
echo "<p>Files in $path</p>";
echo '<ul>';
foreach ($files as $file) {
    echo '<li>' . $file . '</li>';
}
echo '</ul>';
```



Murach's PHP and MySQL (3rd Ed.)

C23, Slide 5

Three functions to test if a file or directory exists

```
is_file($path)
is_dir($path)
file_exists($path)
```

A function to get the current directory

```
getcwd()
```

A constant that contains the path separator

```
DIRECTORY_SEPARATOR
```

A function to get a directory listing

```
scandir($path)
```



Murach's PHP and MySQL (3rd Ed.)

C23, Slide 3

Three functions to read an entire file

Function	Description
<code>file(\$name)</code>	Returns an array with each element containing one line from the file.
<code>file_get_contents(\$name)</code>	Returns the contents of the file as a string.
<code>readfile(\$name)</code>	Reads a file and echoes it to the web page.

A function to write an entire file

Function	Description
<code>file_put_contents(\$name, \$data)</code>	Writes the specified data string to the specified filename.



Murach's PHP and MySQL (3rd Ed.)

C23, Slide 6

How to read and write text

Read text from a file

```
$text = file_get_contents('message.txt');
$text = htmlspecialchars($text);
echo '<div>' . $text . '</div>';
```

Write text to a file

```
$text = "This is line 1.\nThis is line 2.\n";
file_put_contents('message.txt', $text);
```

Functions that open and close a file

Function	Description
<code>fopen(\$path, \$mode)</code>	Opens the specified file with the specified mode and returns a file handle.
<code>feof(\$file)</code>	Returns TRUE when the end of the specified file is reached.
<code>fclose(\$file)</code>	Closes the specified file.

Functions that read from and write to a file

Function	Description
<code>fread(\$file, \$length)</code>	Reads up to the specified number of bytes from the specified file.
<code>fgets(\$file)</code>	Reads a line from the specified file.
<code>fwrite(\$file, \$data)</code>	Writes the specified string data to the specified file.

How to read and write arrays

Read a file into an array

```
$names = file('usernames.txt');
foreach ($names as $name) {
    echo '<div>' . $name . '</div>';
}
```

Write an array to a file

```
$names = array('joelmurach', 'rayharris', 'mikemurach');
$names = implode("\n", $names);
file_put_contents('usernames.txt', $names);
```

Read from a file

```
$file = fopen('usernames.txt', 'rb');
$names = '';
while (!feof($file)) {
    $name = fgets($file);
    if ($name == false) continue;
    $name = trim($name);
    if (strlen($name) == 0 || substr($name, 0, 1) == '#')
        continue;
    $names .= "<div>" . $name . "</div>\n";
}
fclose($file);
echo $names;
```

Modes for opening a file with the fopen() function

Mode	Description
'rb'	Opens the file for reading. If the file doesn't exist, fopen() returns FALSE.
'wb'	Opens the file for writing. If the file exists, the existing data is deleted. If the file doesn't exist, it is created.
'ab'	Opens the file for writing. If the file exists, the new data is appended. If the file doesn't exist, it is created.
'xb'	Creates a new file for writing. If the file exists, fopen() returns FALSE.

Write to a file

```
$path = getcwd();
$item = scandir($path);
$file = fopen('listing.txt', 'wb');
foreach ($item as $item) {
    $item_path = $path . DIRECTORY_SEPARATOR . $item;
    if (is_dir($item_path)) {
        fwrite($file, $item . "\n");
    }
}
fclose($file);
```

Functions that read and write CSV files

Function	Description
<code>fgetcsv(\$file)</code>	Reads in a line of comma-separated values and returns them in an array.
<code>fputcsv(\$file, \$array)</code>	Writes the specified array to the specified file as a line of comma-separated values.

A simple CSV file

```
MMS-1234,Trumpet,199.95
MMS-8521,Flute,149.95
```



Rename a file

```
$name2 = 'message_2.txt';
$name3 = 'message_copy.txt';
if (file_exists($name2)) {
    $success = rename($name2, $name3);
    if ($success) {
        echo '<div>File was renamed.</div>';
    }
}
```

Delete a file

```
$name3 = 'message_copy.txt';
if (file_exists($name3)) {
    $success = unlink($name3);
    if ($success) {
        echo '<div>File was deleted.</div>';
    }
}
```



Write tabular data to a CSV file

```
$products = array(array('MMS-1234', 'Trumpet', 199.95),
    array('MMS-8521', 'Flute', 149.95));
$file = fopen('products.csv', 'wb');
foreach ($products as $product) {
    fputcsv($file, $product);
}
fclose($file);
```

Read tabular data from a CSV file

```
$file = fopen('products.csv', 'rb');
$products = array();
while (!feof($file)) {
    $product = fgetcsv($file);
    if ($product === false) continue;
    $products[] = $product;
    echo "<div>$product[0] | $product[1] |
$product[2]</div>";
}
```



An HTML form for uploading a file

```
<form action="upload.php" method="post"
    enctype="multipart/form-data">
    <input type="file" name="file1"><br>
    <input type="submit" value="Upload">
</form>
```

The browser display of the HTML



Functions to copy, rename, and delete files

```
copy($oldname, $newname)
rename($oldname, $newname)
unlink($name)
```

Copy a file

```
$name1 = 'message.txt';
$name2 = 'message_2.txt';
if (file_exists($name1)) {
    $success = copy($name1, $name2);
    if ($success) {
        echo '<div>File was copied.</div>';
    }
}
```



Elements in the \$_FILES array

- 'name'
- 'size'
- 'tmp_name'
- 'type'
- 'error'

A function to save an uploaded file

```
move_uploaded_file($tmp, $new)
```



PHP for working with an uploaded file

```

$tmp_name = $_FILES['file1']['tmp_name'];
$path = getcwd() . DIRECTORY_SEPARATOR . 'images';
$name = $path . DIRECTORY_SEPARATOR .
    $_FILES['file1']['name'];
$success = move_uploaded_file($tmp_name, $name);
if ($success) {
    $upload_message = $name . ' has been uploaded.';
}

```

**Code that gets image information (continued)**

```

// Display the image type
switch ($image_type) {
    case IMAGETYPE_JPEG:
        echo 'This is a JPEG image.<br>';
        break;
    case IMAGETYPE_GIF:
        echo 'This is a GIF image.<br>';
        break;
    case IMAGETYPE_PNG:
        echo 'This is a PNG image.<br>';
        break;
    default:
        echo 'File must be a JPEG, GIF, or PNG.<br>';
        exit;
}

```

**A function that gets information about an image**

```
getimagesize($path)
```

Common IMAGETYPE constants

```

IMAGETYPE_JPEG
IMAGETYPE_GIF
IMAGETYPE_PNG

```

**Functions that work with images**

Name	Description
<code>imagecreatefromxxx(\$path)</code>	Creates an image of the xxx type from the specified file path.
<code>imagesx(\$image)</code>	Returns the width of the specified image.
<code>imagesy(\$image)</code>	Returns the height of the specified image.
<code>imagexxx(\$image, \$path)</code>	Writes the specified image of the xxx type to the specified file path.
<code>imagedestroy(\$image)</code>	Frees any memory that's used for the specified image.

**Code that gets information about an image**

```

// Set the path to the image
$image_path = getcwd() . DIRECTORY_SEPARATOR .
    'gibson_sg.png';

// Get the image width, height, and type
$image_info = getimagesize($image_path);
$image_width = $image_info[0];
$image_height = $image_info[1];
$image_type = $image_info[2];

```

**Code that reads and writes an image**

```

// Set the paths for the images
$image_path = getcwd() . DIRECTORY_SEPARATOR .
    'gibson_sg.png';
$image_path_2 = getcwd() . DIRECTORY_SEPARATOR .
    'gibson_sg_2.png';

// Get the image width, height, and type
$image_info = getimagesize($image_path);
$image_type = $image_info[2];

```



Code that reads and writes an image (continued)

```
// Set up the function names for the image type
switch($image_type) {
  case IMAGETYPE_JPEG:
    $image_from_file = 'imagecreatefromjpeg';
    $image_to_file = 'imagejpeg';
    break;
  case IMAGETYPE_GIF:
    $image_from_file = 'imagecreatefromgif';
    $image_to_file = 'imagegif';
    break;
  case IMAGETYPE_PNG:
    $image_from_file = 'imagecreatefrompng';
    $image_to_file = 'imagepng';
    break;
  default:
    echo 'File must be a JPEG, GIF, or PNG image.';
    exit;
}
```

Resizing an image to 100 by 100 pixels maximum

```
// Set some variables
$old_path = getcwd() . DIRECTORY_SEPARATOR .
'gibson_sg.png';
$new_path = getcwd() . DIRECTORY_SEPARATOR .
'gibson_sg_100.png';
$image_type = IMAGETYPE_PNG;

// Get the old image and its height and width
$old_image = imagecreatefrompng($old_path);
$old_width = imagesx($old_image);
$old_height = imagesy($old_image);

// Calculate height and width ratios for 100x100 maximum
$width_ratio = $old_width / 100;
$height_ratio = $old_height / 100;
```

Code that reads and writes an image (continued)

```
// Create a new image from the file
$image = $image_from_file($image_path);

// Check the image's width and height
$image_width = imagesx($image);
$image_height = imagesy($image);

// Write the image to a file
$image_to_file($image, $image_path_2);

// Free any memory associated with the image
imagedestroy($image);
```

Resizing an image (continued)

```
// If image larger than ratio, create the new image
if ($width_ratio > 1 || $height_ratio > 1) {

  // Calculate height and width for the new image
  $ratio = max($width_ratio, $height_ratio);
  $new_height = round($old_height / $ratio);
  $new_width = round($old_width / $ratio);

  // Create the new image
  $new_image =
    imagecreatetruecolor($new_width, $new_height);

  // Copy old image to new image to resize the file
  $new_x = 0; // Start new image in upper left corner
  $new_y = 0;
  $old_x = 0; // Copy old image from upper left corner
  $old_y = 0;
  imagecopyresampled($new_image, $old_image,
    $new_x, $new_y, $old_x, $old_y,
    $new_width, $new_height,
    $old_width, $old_height);
}
```

Functions that can resize an image

Name	Description
<code>imagecreatetruecolor(\$w, \$h)</code>	Returns an all black truecolor image of the specified size.
<code>imagecopyresampled(\$di, \$si, \$dx, \$dy, \$sx, \$sy, \$dw, \$dh, \$sw, \$sh)</code>	Copies a rectangular portion of the source image (s) to the destination image (d), resizing the image if necessary.

Resizing an image (continued)

```
// Write the new image to a file
imagepng($new_image, $new_path);

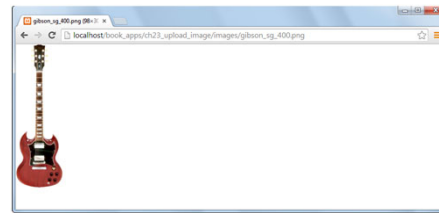
// Free any memory associated with the new image
imagedestroy($new_image);
}

// Free any memory associated with the old image
imagedestroy($old_image);
```

Functions that work with image transparency

Name	Description
<code>imagecolorallocatealpha(\$i, \$r, \$g, \$b, \$a)</code>	Returns an identifier for the transparent (alpha) color of the specified image. The RGB values specify the color. The alpha value specifies the amount of transparency.
<code>imagecolortransparent(\$i, \$a)</code>	Sets the transparent color in the specified image.
<code>imagealphablending(\$i, \$f)</code>	To turn alpha blending mode off, set the second parameter to FALSE.
<code>imagesavealpha(\$i, \$f)</code>	To attempt to save full alpha channel information, set the second parameter to TRUE (alpha blending mode must be turned off).

An image displayed in the browser



Code that works with image transparency

```
// calculate the width and height for the new image
// and set the image type for the new image
$new_image =
    imagecreatetruecolor($new_width, $new_height);

// Set transparency according to image type
if ($image_type == IMAGETYPE_GIF) {
    $alpha =
        imagecolorallocatealpha($new_image, 0, 0, 0, 127);
    imagecolortransparent($new_image, $alpha);
}
if ($image_type == IMAGETYPE_PNG)
    || $image_type == IMAGETYPE_GIF) {
    imagealphablending($new_image, false);
    imagesavealpha($new_image, true);
}

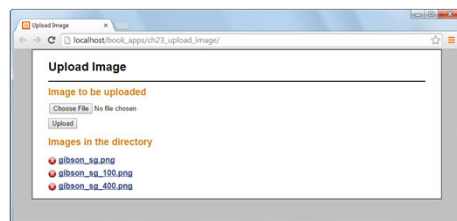
// code that writes the new image to a file
```

The file_util.php file

```
function get_file_list($path) {
    $files = array();
    if (!is_dir($path)) return $files;

    $items = scandir($path);
    foreach ($items as $item) {
        $item_path = $path . DIRECTORY_SEPARATOR . $item;
        if (is_file($item_path)) {
            $files[] = $item;
        }
    }
    return $files;
}
```

The user interface



The image_util.php file

```
<?php
function process_image($dir, $filename) {
    // Set up the variables
    $dir = $dir . DIRECTORY_SEPARATOR;
    $i = strpos($filename, '.');
    $image_name = substr($filename, 0, $i);
    $ext = substr($filename, $i);

    // Set up the read path
    $image_path = $dir . DIRECTORY_SEPARATOR . $filename;

    // Set up the write paths
    $image_path_400 = $dir . $image_name . '400' . $ext;
    $image_path_100 = $dir . $image_name . '_100' . $ext;

    // Create an image that's a maximum of 400x300 pixels
    resize_image($image_path, $image_path_400, 400, 300);

    // Create a thumbnail image that's 100x100 pixels max
    resize_image($image_path, $image_path_100, 100, 100);
}
```

The image_util.php file (continued)

```
function resize_image($old_image_path, $new_image_path,
    $max_width, $max_height) {

    // Get image type
    $image_info = getimagesize($old_image_path);
    $image_type = $image_info[2];
```

**The image_util.php file (continued)**

```
// If image larger than ratio, create the new image
if ($width_ratio > 1 || $height_ratio > 1) {

    // Calculate height and width for the new image
    $ratio = max($width_ratio, $height_ratio);
    $new_height = round($old_height / $ratio);
    $new_width = round($old_width / $ratio);

    // Create the new image
    $new_image =
        imagecreatetruecolor($new_width, $new_height);

    // Set transparency according to image type
    if ($image_type == IMAGETYPE_GIF) {
        $alpha = imagecolorallocatealpha(
            $new_image, 0, 0, 0, 127);
        imagecolortransparent($new_image, $alpha); }
    if ($image_type == IMAGETYPE_PNG ||
        $image_type == IMAGETYPE_GIF) {
        imagealphablending($new_image, false);
        imagesavealpha($new_image, true); }
```

**The image_util.php file (continued)**

```
// Set up the function names
switch($image_type) {
    case IMAGETYPE_JPEG:
        $image_from_file = 'imagecreatefromjpeg';
        $image_to_file = 'imagejpeg';
        break;
    case IMAGETYPE_GIF:
        $image_from_file = 'imagecreatefromgif';
        $image_to_file = 'imagegif';
        break;
    case IMAGETYPE_PNG:
        $image_from_file = 'imagecreatefrompng';
        $image_to_file = 'imagepng';
        break;
    default:
        echo 'File must be a JPEG, GIF, or PNG image.';
        exit;
}
```

**The image_util.php file (continued)**

```
// Copy old image to new image and resize
$new_x = 0;
$new_y = 0;
$old_x = 0;
$old_y = 0;
imagecopyresampled($new_image, $old_image,
    $new_x, $new_y, $old_x, $old_y,
    $new_width, $new_height, $old_width, $old_height);

// Write the new image to a new file
$image_to_file($new_image, $new_image_path);

// Free any memory associated with the new image
imagedestroy($new_image);

} else {
    // Write the old image to a new file
    $image_to_file($old_image, $new_image_path); }

// Free any memory associated with the old image
imagedestroy($old_image); } ?>
```

**The image_util.php file (continued)**

```
// Get the old image and its height and width
$old_image = $image_from_file($old_image_path);
$old_width = imagesx($old_image);
$old_height = imagesy($old_image);

// Calculate height and width ratios
$width_ratio = $old_width / $max_width;
$height_ratio = $old_height / $max_height;
```

**The controller (index.php)**

```
<?php
require_once 'file_util.php'; // the get_file_list function
require_once 'image_util.php'; // the process_image function

$image_dir = 'images';
$image_dir_path = getcwd() . DIRECTORY_SEPARATOR .
    $image_dir;

$action = filter_input(INPUT_POST, 'action');
if ($action == NULL) {
    $action = filter_input(INPUT_GET, 'action');
    if ($action == NULL) {
        $action = '';
    }
}
```



The controller (continued)

```

switch ($action) {
    case 'upload':
        if (isset($_FILES['file1'])) {
            $filename = $_FILES['file1']['name'];
            if (empty($filename)) {
                break;
            }
            $source = $_FILES['file1']['tmp_name'];
            $target = $image_dir_path . DIRECTORY_SEPARATOR
                . $filename;
            move_uploaded_file($source, $target);

            // create the '400' and '100' images
            process_image($image_dir_path, $filename);
        }
        break;

```

**The view (uploadform.php) (continued)**

```

<h2>Images in the directory</h2>
<?php if (count($files) == 0) : ?>
<p>No images uploaded.</p>
<?php else : ?>
<ul>
<?php foreach($files as $filename) :
    $file_url = $image_dir . '/' .
        $filename;
    $delete_url =
        '?action=delete&filename=' .
            urlencode($filename);
?>
<li>
    <a href="<?php echo $delete_url;?>">
        </a>
    <a href="<?php echo $file_url; ?>">
        <?php echo $filename; ?></a>
</li>
<?php endforeach; ?>
</ul>

```

**The controller (continued)**

```

case 'delete':
    $filename = filter_input(INPUT_GET, 'filename',
        FILTER_SANITIZE_STRING);
    $target = $image_dir_path . DIRECTORY_SEPARATOR .
        $filename;
    if (file_exists($target)) {
        unlink($target);
    }
    break;
}

$files = get_file_list($image_dir_path);
include('uploadform.php');
?>

```

**The view (uploadform.php) (continued)**

```

<?php endif; ?>
</main>
</body>
</html>

```

**The view (uploadform.php)**

```

<!DOCTYPE html>
<html>
<head>
    <title>Upload Image</title>
    <link rel="stylesheet" type="text/css" href="main.css"/>
</head>
<body>
    <header>
        <h1>Upload Image</h1>
    </header>
    <main>
        <h2>Image to be uploaded</h2>
        <form id="upload_form"
            action="." method="POST"
            enctype="multipart/form-data">
            <input type="hidden" name="action"
                value="upload">
            <input type="file" name="file1"><br>
            <input id="upload_button" type="submit"
                value="Upload">
        </form>
    </main>

```

