COMP30680
Web Application Development

PHP – Error handling

David Coyle
d.coyle@ucd.ie

PHP Error Handling

Error handling is an important part of creating scripts and web applications.

If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

We’re going to cover some of the most common error checking methods in PHP.

- Simple "die()" statements
- Custom errors and error triggers
- Error reporting

We’ll also look at Exceptions

See http://www.w3schools.com/php/php_error.asp

The default error handling in PHP is very simple.
An error message with filename, line number and a message describing the error is sent to the browser.

The die() function

Consider the following code:

```
<?php
$file=fopen("welcome.txt","r");
?>
```

If “welcome.txt” does not exist, then by default you will get a message like the following in your browser:

```
Warning: fopen(welcome.txt) [function.fopen]: failed to open stream:
No such file or directory in C:\webfolder\test.php on line 2
```

A better approach is to check if the file exists and create your own message:

```
<?php
if(!file_exists("welcome.txt")) {
  die("File not found");
} else {
  $file=fopen("welcome.txt","r");
}
?>
```

This is better than the earlier code, because it uses a simple error handling mechanism to stop the script after the error.

Custom Error Handlers

Even better, create a custom error handler: a special function that can be called when an error occurs in PHP.

```
error_function(error_level,error_message,
error_file,error_line,error_context)
```

This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context):

Parameter	Description
error_level	Required. Specifies the error report level for the user-defined error. Must be a value number. See table below for possible error report levels
error_message	Required. Specifies the error message for the user-defined error
error_file	Optional. Specifies the filename in which the error occurred
error_line	Optional. Specifies the line number in which the error occurred
error_context	Optional. Specifies an array containing every variable, and their values, in use when the error occurred

Error Report levels

These error report levels are the different types of error the error handler can be used for.

Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
8191	E_ALL	All errors and warnings (E_STRICT became a part of E_ALL in PHP 5.4)

Create an Error Handler

```
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Ending Script";
  die();
}
```

This code creates a simple error handling function.

When it is triggered, it gets the error level and an error message. It then outputs the error level and message and terminates the script.

The next step is to decide when it should be triggered.

Set the Error Handler

```
set_error_handler("customError");
```

The default error handler for PHP is the built in error handler. You can change the default error handler for the duration of the script.

```
<?php
//error handler function
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr";
}

//set error handler
set_error_handler("customError");

//trigger error
echo($test);
?>
```

The output of this code should be something like this:

```
Error: [8] Undefined variable: test
```

It is also possible to change the error handler to apply for only some errors, that way the script can handle different errors in different ways.

Explicitly trigger an error

In a script where users can input data it is useful to trigger errors when an illegal input occurs. In PHP, this is done by the trigger_error() function.

```php
<?php
$test=2;
if ($test>1) {
  trigger_error("Value must be 1 or below");
}
?>
```

The output of the code above should be something like this:

```
Notice: Value must be 1 or below
in C:\webfolder\test.php on line 6
```

Specify the error level

An error can be triggered anywhere you wish in a script, and by adding a second parameter, you can specify what error level is triggered.

Possible error types:

- E_USER_ERROR - Fatal user-generated run-time error. Errors that can not be recovered from. Execution of the script is halted

- E_USER_WARNING - Non-fatal user-generated run-time warning. Execution of the script is not halted

- E_USER_NOTICE - Default. User-generated run-time notice. The script found something that might be an error, but could also happen when running a script normally

Specify the error level

```php
<?php
//error handler function
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Ending Script";
  die();
}

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
if ($test>1) {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
}
?>
```

E_USER_WARNING occurs if the "test" variable is bigger than "1".

If an E_USER_WARNING occurs we will use our custom error handler and end the script

```
Error: [512] Value must be 1 or below
Ending Script
```

Error logging

By default, PHP sends an error log to the server's logging system or a file, depending on how the error_log configuration is set in the php.ini file. By using the error_log() function you can send error logs to a specified file or a remote destination, or even send an email.

```php
<?php
//error handler function
function customError($errno, $errstr) {
  echo "<b>Error:</b> [$errno] $errstr<br>";
  echo "Webmaster has been notified";
  error_log("Error: [$errno] $errstr",1,
  "someone@example.com","From: webmaster@example.com");
}

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
if ($test>1) {
  trigger_error("Value must be 1 or below",E_USER_WARNING);
}
?>
```

The mail received from the code above looks like this:

```
Error: [512] Value must be 1 or below
```

PHP Exception Handling

Exception handling is used to change the normal flow of the code execution if a specified error (exceptional) condition occurs. This condition is called an exception.

This is what normally happens when an exception is triggered:

- The current code state is saved
- The code execution will switch to a predefined (custom) exception handler function
- Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code

W3schools describe different error handling methods:

- Basic use of Exceptions
- Creating a custom exception handler
- Multiple exceptions
- Re-throwing an exception
- Setting a top level exception handler

See http://www.w3schools.com/php/php_exception.asp

PHP Exception Handling

Exception handling is used to change the normal flow of the code execution if a specified error (exceptional) condition occurs. This condition is called an exception.

This is what normally happens when an exception is triggered:

- The current code state is saved
- The code execution will switch to a predefined (custom) exception handler function
- Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code

W3schools describe different error handling methods:

- **Basic use of Exceptions**
- Creating a custom exception handler
- Multiple exceptions
- Re-throwing an exception
- Setting a top level exception handler

See http://www.w3schools.com/php/php_exception.asp

Basic Use of Exceptions

When an exception is **thrown**, the code following it will not be executed, and PHP will try to find the matching "**catch**" block.

If an exception is not caught, a fatal error will be issued with an "Uncaught Exception" message.

```php
<?php
//create function with an exception
function checkNum($number) {
  if($number>1) {
    throw new Exception("Value must be 1 or below");
  }
  return true;
}

//trigger exception
checkNum(2);
?>
```

```
Fatal error: Uncaught exception 'Exception'
with message 'Value must be 1 or below' in C:\webfolder\test.php:6
Stack trace: #0 C:\webfolder\test.php(12):
checkNum(28) #1 {main} thrown in C:\webfolder\test.php on line 6
```

See **throw_no_catch.php**

Try, throw and catch

To avoid the previous error, we need to create the proper code to handle an exception.

Proper exception code should include:

- **Try** - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown"
- **Throw** - This is how you trigger an exception. Each "throw" must have at least one "catch"
- **Catch** - A "catch" block retrieves an exception and creates an object containing the exception information

```
<?php
//create function with an exception
function checkNum($number) {
  if($number>1) {
    throw new Exception("Value must be 1 or below");
  }
  return true;
}

//trigger exception in a "try" block
try {
  checkNum(2);
  //If the exception is thrown, this text will not be show
  echo 'If you see this, the number is 1 or below';
}

//catch exception
catch(Exception $e) {
  echo 'Message: ' .$e->getMessage();
}
?>
```

Try, throw and catch

This code throws an exception and catches it:

1. The checkNum() function is created. It checks if a number is greater than 1. If it is, an exception is thrown
2. The checkNum() function is called in a "try" block
3. The exception within the checkNum() function is thrown
4. The "catch" block retrieves the exception and creates an object ($e) containing the exception information
5. The error message from the exception is echoed by calling $e->getMessage() from the exception object

```
<?php
//create function with an exception
function checkNum($number) {
  if($number>1) {
    throw new Exception("Value must be 1 or below");
  }
  return true;
}

//trigger exception in a "try" block
try {
  checkNum(2);
  //If the exception is thrown, this text will not be show
  echo 'If you see this, the number is 1 or below';
}

//catch exception
catch(Exception $e) {
  echo 'Message: ' .$e->getMessage();
}
?>
```

Questions, Suggestions?

Materials and further reading:

w3Schools advanced PHP tutorials:
http://www.w3schools.com/php/php_includes.asp

Next:

Forms and filters