26. PHP Error Handling

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.

26.1. PHP Error Handling

When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

This tutorial contains some of the most common error checking methods in PHP.

We will show different error handling methods:

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

26.2. Basic Error Handling: Using the die() function

The first example shows a simple script that opens a text file:

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

If the file does not exist you might get an error like this:

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

To prevent the user from getting an error message like the one above, we test whether the file exist before we try to access it:

```
<?php
if(!file_exists("welcome.txt"))
  {
  die("File not found");
  }
```

```
     else
       {
       $file=fopen("welcome.txt","r");
       }
     ?>
```

Now if the file does not exist you get an error like this:

```
     File not found
```

The code above is more efficient than the earlier code, because it uses a simple error handling mechanism to stop the script after the error.

However, simply stopping the script is not always the right way to go. Let's take a look at alternative PHP functions for handling errors.

26.3. Creating a Custom Error Handler

Creating a custom error handler is quite simple. We simply create a special function that can be called when an error occurs in PHP.

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):

Syntax

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

26.4. Error Report levels

These error report levels are the different types of error the user-defined 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)

Now lets create a function to handle errors:

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

The code above is 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.

Now that we have created an error handling function we need to decide when it should be triggered.

26.5. Set Error Handler

The default error handler for PHP is the built in error handler. We are going to make the function above the default error handler for the duration of the script.

It is possible to change the error handler to apply for only some errors, that way the script can handle different errors in different ways. However, in this example we are going to use our custom error handler for all errors:

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

Since we want our custom function to handle all errors, the set_error_handler() only needed one parameter, a second parameter could be added to specify an error level.

Example

Testing the error handler by trying to output variable that does not exist:

```
<?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 the code above should be something like this:

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

26.6. 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.

Example

In this example an error occurs if the "test" variable is bigger than "1":

```
<?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
```

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

Example

In this example an 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:

```
<?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);
       }
     ?>
```

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

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

Now that we have learned to create our own errors and how to trigger them, lets take a look at error logging.

26.7. 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.

Sending error messages to yourself by e-mail can be a good way of getting notified of specific errors.

26.8. Send an Error Message by E-Mail

In the example below we will send an e-mail with an error message and end the script, if a specific error occurs:

```
     <?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 output of the code above should be something like this:

```
Error: [512] Value must be 1 or below
Webmaster has been notified
```

And the mail received from the code above looks like this:

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

This should not be used with all errors. Regular errors should be logged on the server using the default PHP logging system.