Lab 06: Error Handling

In this lab exercise, you will learn how to deal with error handling and logging.

By default, when an error occurs, error message with filename, line number and a message describing the error is sent to the browser.

Alternatively, you can also define your own error handling rules, as well as modify the way the errors can be logged. This allows you to change and enhance error reporting to suit your needs. In additions, you can send messages directly to other machines, to an email (or email to pager gateway!), to system logs, etc., so you can selectively log and monitor the most important parts of your applications and websites.

1.1 Default PHP Errors

When PHP encounters error, it will throw an error message to the web browser, dependent on the settings in the phpinfo.ini file. You can change these settings in PHP Manager.

By default, it is set to OFF. Hence, only a blank HTML page is returned when an error occurs.

When it is set to ON (which we did, in Lab 01), PHP will return the error message, location of php file the error occurs and line number.

Sample default error message is shown in the Figure 1 below.

Figure 1: Default PHP error message, showing the error message, file location and line number.

This is not desirable as it could pose risks of the message displaying sensitive information pertaining to the page execution, potentially revelling variable values and/or internal execution process. These risks can be overcome using custom error handler.

1.2 Creating a Custom Error Handler

A custom error handler is created by using 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).

Creating a custom error handler involves 3 parts, i.e. custom function, handler registration, and error trigger.

1.2.1 Custom Function

The following is the syntax for custom function:

<error_function> (<error_level>, <error_message> [, <error_file>, <error_line>, <error_context>])

Level Constant

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)

1.2.2 Registering Error Handler

The following is the syntax for registering error handler:

set_error_handler (<error_function>, [<error_level>]);

1.2.3 Triggering an Error

Error can be triggered to generate a user-level error/warning/notice message. This function is useful when you need to generate a particular response to an exception at runtime.

The following is the syntax to trigger an error:

trigger_error (<error_msg> [, <error_level>])

Refer to Figure 2 below for an example of how to create and use custom error handler.

```php
<?php

function customErrMsg ($ErrLevel, $ErrMsg) {
    switch ($ErrLevel) {
        case E_USER_ERROR:
            echo "<b>Fatal Error:</b> [$ErrLevel] $ErrMsg<br/>\n";
            break;
        case E_USER_WARNING:
            echo "<b>Warning:</b> [$ErrLevel] $ErrMsg<br/>\n";
            break;
        case E_USER_NOTICE:
            echo "<b>Important Notice:</b> [$ErrLevel] $ErrMsg<br/>\n";
            break;
        default:
            echo "<b>Unknown error</b>: [$ErrLevel] $ErrMsg<br/>\n";
            break;
    }
    return true;
}

set_error_handler (customErrMsg);

trigger_error ("Test custom error", E_USER_ERROR);
trigger_error ("Display custom error by code number", 512);

?>
```

Figure 2: Sample PHP code using custom error handler.

You tasks:

1. Create a page that contains custom error handler.
2. The page is used to receive inputs from the user to calculate a nett total, as follows:
   INPUT: Customer's Name, Product Code, Gross Total, Tax Rate, Date
   OUTPUT Nett Total, all entered inputs
3. All inputs are required. Give a custom error for any incomplete or invalid input.
4. The custom error message should be displayed next to (or around) where the error occurs on the user's page. All entered input are to be re-populated back into input textboxes.

-- END --