Exception handling and error (Error & Exception Handling)

Error handling is the process of catching errors created by your program and then taking the appropriate actions. If you handle errors incorrectly, it can lead to unexpected results.

Error handling is the process of detecting errors created when running your code and then taking the appropriate actions. If you handle errors incorrectly, it can lead to unexpected results. In **PHP** , handling a error is quite simple.

Use the die () function in PHP

During PHP programming, you should check all possible error conditions before continuing and taking appropriate actions when necessary.

Try the following example without the **/tmp/test.xt** file and with this file.

```
   php // Ph?n ki?m tra ?i?u ki?n l?i.
  if (! file_exists ( "/tmp/test.txt" )) { die ( "Không tìm th?
y file!" ); } else {  $file  =  fopen ( "/tmp/test.txt" , "r" ); print "M?
 file thành công" ; } // Ph?n code chính c?a ch??ng trình. ?>
```

In this way, you can write an effective code. Using the die () function, you can stop the program whenever an error occurs and display a more friendly and meaningful message to the user.

Manually define functions to handle errors in PHP

You can write your own function to handle errors arising in your program. PHP provides you with a framework for defining error handling functions.

This function must be able to manipulate at least two parameters (error level and error message), but up to 5 parameters (optional: file, line-number, and error context):

Syntax

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

The following table details the above parameters:

Parameter Descriptionerror_level Required - Defines the error level for the self-defined error (user-defined). Must be a numeric value. error_message Required - Define error messages for self-defined errors. error_file

Optional - Define the file name where the error occurred. error_line Optional - Defines the number of lines where the error occurred. error_context Options - Define an array containing all their variables and values, use when an error occurs.

Possible error levels in PHP

These error levels are different types of errors. These values ??can be used in combination by using the operator.

ValueNumberDescription1 E_ERROR Error at critical run time (Fatal run-time error). Critical errors and script execution are stopped. 2 E_WARNING Error at runtime but not serious (Non-fatal run-time error). The errors are not serious and the script execution is not stopped. 4 E_PARSE Error parsing while compiling (Compile-time parse error). These syntax errors should only be created by the parser. 8 E_NOTICE Run-time notice. The script finds something that might be an error, but it can also happen when running a script normally. 16 E_CORE_ERROR Critical errors occurred during the initial launch of PHP. 32 E_CORE_WARNING Critical errors occur during the initial launch of PHP. 256 E_USER_ERROR A serious user-generated error. It is the same as the E_ERROR error set by the programmer using the trigger_error () function in PHP. 512 E_USER_WARNING A non-serious user-generated error. It is the same as the E_WARNING error set by the programmer using PHP's trigger_error () function. 1024 E_USER_NOTICE Notice created by the user. It is like an E_NOTICE set by the programmer with the function trigger_error () in PHP. 2048 E_STRICT Notice while running. Allows HPH to propose changes to the code, to ensure the best interoperability and compatibility for the code. 4096 E_RECOVERABLE_ERROR Critical error can be captured. Same as an E_ERROR but can be captured by the self-defined error handler (refer to set_error_handler ()). 8191 E_ALL All errors and warnings, except E_STRICT (E_STRICT will be part of E_ALL since PHP 6.0).
All of the above error levels can be set using the following function library available in PHP, with the level can be any value defined in the table above.

```
error_reporting int ([$ int level])
```

Here's how you can create an error handling function in PHP:

```
";
echo "Stop PHP Script";
die ();
}
?>
```

Once the error handler has been defined, you will use the **set_error_handler** function available in PHP to set it up. Now check the above example by calling a function that does not exist.

```
   php error_reporting (  E_ERROR  ); function  handleError ( $errno ,  $errstr ,
X?y ra l?i: [$errno] $errstr - $error_file:$error_line" ;  echo  "
" ;  echo  "D?ng PHP Script" ; die (); } //thi?t l?p trình x? lý l?
i  set_error_handler ( "handleError" ); //code ch?y khi l?i x?
y ra (trigger error)  myFunction (); ?>
```

Exception handling (Exception Handling) in PHP

PHP 5 has an Exception Model (Exception Model) similar to other programming languages. Exceptions are important and provide better control over error handling

Below explains some keywords related to exception in PHP:

1. **Try** : A function that uses an exception must be in the try block. If the exception is not enabled, the code will continue as usual. However, if the exception is enabled, an exception is "thrown".
2. **Throw** : This is how you trigger an exception. Each "throw" must have at least one "catch".
3. **Catch** : Each block "catch" receives an exception and creates an object that contains the exception information.

When an exception is "thrown", the code following that command will not be executed, and PHP will try to find the first matching block. If an exception is not "catch", a Fatal Error in PHP will be notified with the "Uncaught Exception ." introduction.

1. An exception can be "throw", and "catch" inside PHP. Code can be encapsulated in a try block.
2. Each try block must have at least one corresponding catch block. Many catch blocks can be used to catch different exception classes.
3. Exceptions can be "thrown" (or "re-throw") inside a catch block.

For example

Below is a code, you copy and paste this code into a file and check the results:

```
   php  try {  $error  = 'Luôn qu?ng ra l?
i này' ; throw new Exception ( $error ); // Ph?n code theo sau m?t ngo?i l?
 s? không ???c th?c thi.  echo  'Ph?n code không bao gi? ???c th?
c thi' ; } catch ( Exception  $e ) {  echo  'B?t m?
t exception: ' ,  $e -> getMessage (), "n" ; } // Ti?p t?c th?
c thi  echo  'Xin chào!' ; ?>
```

In the above example, the function $ e-> getMessage is used to get the error message. Here are some functions that can be used from the **Exception** class in PHP.

1. **getMessage ()** - notice of exception
2. **getCode ()** - code of exception
3. **getFile ()** - source file name
4. **getLine ()** - source line
5. **getTrace ()** - n array of backtrace ()
6. **getTraceAsString ()** - string is formatted by trace

Create Custom Exception Handler in PHP

You can define the Exception Handler yourself. Use the following functions to set up a self-defined exception handler function.

```
  string set_exception_handler ($ exception_handler callback)
```

Here, exception_handler is the function name called when an uncaught exception appears. This function must be defined before calling the set_exception_handler () function.

For example

Follow tutorialspoint

Previous lesson: Regular Expression in PHP

Next lesson: Bug and Debug in PHP

You finished reading the article "**Exception handling and error (Error & Exception Handling)**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.

© 2019 TipsMake.com