[**PHP Error 和 Logging 函数** 1](#_Toc278906056)

[debug\_backtrace() 函数 3](#_Toc278906057)

[debug\_print\_backtrace() 函数 5](#_Toc278906058)

[error\_get\_last() 函数 6](#_Toc278906059)

[error\_log() 函数 8](#_Toc278906060)

[error\_reporting() 函数 9](#_Toc278906061)

[restore\_error\_handler() 函数 11](#_Toc278906062)

[restore\_exception\_handler() 函数 13](#_Toc278906063)

[set\_error\_handler() 函数 14](#_Toc278906064)

[set\_exception\_handler() 函数 15](#_Toc278906065)

[set\_error\_handler() 函数 17](#_Toc278906066)

[set\_exception\_handler() 函数 18](#_Toc278906067)

[trigger\_error() 函数 19](#_Toc278906068)

**PHP Error 和 Logging 函数**

* [Previous Page](http://www.w3school.com.cn/php/php_ref_directory.asp)
* [Next Page](http://www.w3school.com.cn/php/php_ref_filesystem.asp)

**PHP Error 和 Logging 简介**

error 和 logging 函数允许你对错误进行处理和记录。

error 函数允许用户定义错误处理规则，并修改记录错误的方式。

logging 函数允许用户对应用程序进行日志记录，并把日志消息发送到电子邮件、系统日志或其他的机器。

**安装**

error 和 logging 函数是 PHP 核心的组成部分。无需安装即可使用这些函数。

**PHP Error 和 Logging 函数**

PHP：指示支持该函数的最早的 PHP 版本。

|  |  |  |
| --- | --- | --- |
| **函数** | **描述** | **PHP** |
| [debug\_backtrace()](http://www.w3school.com.cn/php/func_error_debug_backtrace.asp) | 生成 backtrace。 | 4 |
| [debug\_print\_backtrace()](http://www.w3school.com.cn/php/func_error_debug_print_backtrace.asp) | 输出 backtrace。 | 5 |
| [error\_get\_last()](http://www.w3school.com.cn/php/func_error_get_last.asp) | 获得最后发生的错误。 | 5 |
| [error\_log()](http://www.w3school.com.cn/php/func_error_log.asp) | 向服务器错误记录、文件或远程目标发送一个错误。 | 4 |
| [error\_reporting()](http://www.w3school.com.cn/php/func_error_reporting.asp) | 规定报告哪个错误。 | 4 |
| [restore\_error\_handler()](http://www.w3school.com.cn/php/func_error_restore_error_handler.asp) | 恢复之前的错误处理程序。 | 4 |
| [restore\_exception\_handler()](http://www.w3school.com.cn/php/func_error_restore_exception_handler.asp) | 恢复之前的异常处理程序。 | 5 |
| [set\_error\_handler()](http://www.w3school.com.cn/php/func_error_set_error_handler.asp) | 设置用户自定义的错误处理函数。 | 4 |
| [set\_exception\_handler()](http://www.w3school.com.cn/php/func_error_set_exception_handler.asp) | 设置用户自定义的异常处理函数。 | 5 |
| [trigger\_error()](http://www.w3school.com.cn/php/func_error_trigger_error.asp) | 创建用户自定义的错误消息。 | 4 |
| user\_error() | trigger\_error() 的别名。 | 4 |

**PHP Error 和 Logging 常量**

PHP：指示支持该常量的最早的 PHP 版本。

|  |  |  |  |
| --- | --- | --- | --- |
| **值** | **常量** | **描述** | **PHP** |
| 1 | E\_ERROR | 致命的运行时错误。错误无法恢复。脚本的执行被中断。 |  |
| 2 | E\_WARNING | 非致命的运行时错误。脚本的执行不会中断。 |  |
| 4 | E\_PARSE | 编译时语法解析错误。解析错误只应该由解析器生成。 |  |
| 8 | E\_NOTICE | 运行时提示。可能是错误，也可能在正常运行脚本时发生。 |  |
| 16 | E\_CORE\_ERROR | 由 PHP 内部生成的错误。 | 4 |
| 32 | E\_CORE\_WARNING | 由 PHP 内部生成的警告。 | 4 |
| 64 | E\_COMPILE\_ERROR | 由 Zend 脚本引擎内部生成的错误。 | 4 |
| 128 | E\_COMPILE\_WARNING | 由 Zend 脚本引擎内部生成的警告。 | 4 |
| 256 | E\_USER\_ERROR | 由于调用 trigger\_error() 函数生成的运行时错误。 | 4 |
| 512 | E\_USER\_WARNING | 由于调用 trigger\_error() 函数生成的运行时警告。 | 4 |
| 1024 | E\_USER\_NOTICE | 由于调用 trigger\_error() 函数生成的运行时提示。 | 4 |
| 2048 | E\_STRICT | 运行时提示。对增强代码的互用性和兼容性有益。 | 5 |
| 4096 | E\_RECOVERABLE\_ERROR | 可捕获的致命错误。（参阅 set\_error\_handler()） | 5 |
| 8191 | E\_ALL | 所有的错误和警告，除了 E\_STRICT。 | 5 |

# debug\_backtrace() 函数

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

## 定义和用法

PHP debug\_backtrace() 函数生成一个 backtrace。

该函数返回一个关联数组。下面是可能返回的元素：

|  |  |  |
| --- | --- | --- |
| **名称** | **类型** | **描述** |
| function | 字符串 | 当前的函数名。 |
| line | 整数 | 当前的行号。 |
| file | 字符串 | 当前的文件名。 |
| class | 字符串 | 当前的类名 |
| object | 对象 | 当前对象。 |
| type | 字符串 | 当前的调用类型，可能的调用：   * 返回: "->"  - 方法调用 * 返回: "::"  - 静态方法调用 * 返回 nothing - 函数调用 |
| args | 数组 | 如果在函数中，列出函数参数。如果在被引用的文件中，列出被引用的文件名。 |

### 语法

debug\_backtrace()

## 例子

<?php

function one($str1, $str2)

{

two("Glenn", "Quagmire");

}

function two($str1, $str2)

{

three("Cleveland", "Brown");

}

function three($str1, $str2)

{

print\_r(debug\_backtrace());

}

one("Peter", "Griffin");

?>

输出：

Array

(

[0] => Array

(

[file] => C:\webfolder\test.php

[line] => 7

[function] => three

[args] => Array

(

[0] => Cleveland

[1] => Brown

)

)

[1] => Array

(

[file] => C:\webfolder\test.php

[line] => 3

[function] => two

[args] => Array

(

[0] => Glenn

[1] => Quagmire

)

)

[2] => Array

(

[file] => C:\webfolder\test.php

[line] => 14

[function] => one

[args] => Array

(

[0] => Peter

[1] => Griffin

)

)

)

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

# debug\_print\_backtrace() 函数

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

## 定义和用法

debug\_print\_backtrace() 函数输出 backtrace。

### 语法

debug\_print\_backtrace()

## 例子

<?php

function one($str1, $str2)

{

two("Glenn", "Quagmire");

}

function two($str1, $str2)

{

three("Cleveland", "Brown");

}

function three($str1, $str2)

{

debug\_print\_backtrace();

}

one("Peter", "Griffin");

?>

输出：

#0 three(Cleveland, Brown) called at [C:\webfolder\test.php:8]

#1 two(Glenn, Quagmire) called at [C:\webfolder\test.php:4]

#2 one(Peter, Griffin) called at [C:\webfolder\test.php:15]

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

# error\_get\_last() 函数

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

## 定义和用法

error\_get\_last() 函数获取最后发生的错误。

该函数以数组的形式返回最后发生的错误。

返回的数组包含 4 个键和值：

* [type] - 错误类型
* [message] - 错误消息
* [file] - 发生错误所在的文件
* [line] - 发生错误所在的行

### 语法

error\_get\_last()

## 例子

<?php

echo $test;

print\_r(error\_get\_last());

?>

输出：

Array

(

[type] => 8

[message] => Undefined variable: test

[file] => C:\webfolder\test.php

[line] => 2

)

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

# error\_log() 函数

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

## 定义和用法

error\_log() 函数向服务器错误记录、文件或远程目标发送一个错误。

若成功，返回 true，否则返回 false。

### 语法

error\_log(error,type,destination,headers)

|  |  |
| --- | --- |
| **参数** | **描述** |
| error | 必需。要记录的错误消息。 |
| type | 可选。规定错误记录的类型。  可能的记录类型：   * 0 - 默认。根据在 php.ini 文件中的 error\_log 配置，错误被发送到服务器日志系统或文件。 * 1 - 错误被发送到 destination 参数中的地址。只有该类型使用 headers 参数。 * 2 - 通过 PHP debugging 连接来发送错误。该选项只在 PHP 3 中可用。 * 3 - 错误发送到文件目标字符串。 |
| destination | 可选。规定向何处发送错误消息。该参数的值依赖于 "type" 参数的值。 |
| headers | 可选。只在 "type" 为 1 时使用。  规定附加的头部，比如 From, Cc 以及 Bcc。由 CRLF (\r\n) 分隔。  注释：在发送电子邮件时，必须包含 From 头部。可以在 php.ini 文件中或者通过此参数设置。 |

## 例子

本例发送一封带有自定义错误的电子邮件：

<?php

$test=2;

if ($test>1)

{

error\_log("A custom error has been triggered",

1,"someone@example.com","From: webmaster@example.com");

}

?>

输出：

A custom error has been triggered

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

# error\_reporting() 函数

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

## 定义和用法

error\_reporting() 设置 PHP 的报错级别并返回当前级别。

### 语法

error\_reporting(report\_level)

如果参数 level 未指定，当前报错级别将被返回。下面几项是 level 可能的值：

|  |  |  |
| --- | --- | --- |
| **值** | **常量** | **描述** |
| 1 | E\_ERROR | Fatal run-time errors. Errors that can not be recovered from. Execution of the script is halted |
| 2 | E\_WARNING | Non-fatal run-time errors. Execution of the script is not halted |
| 4 | E\_PARSE | Compile-time parse errors. Parse errors should only be generated by the parser |
| 8 | E\_NOTICE | Run-time notices. The script found something that might be an error, but could also happen when running a script normally |
| 16 | E\_CORE\_ERROR | Fatal errors at PHP startup. This is like an E\_ERROR in the PHP core |
| 32 | E\_CORE\_WARNING | Non-fatal errors at PHP startup. This is like an E\_WARNING in the PHP core |
| 64 | E\_COMPILE\_ERROR | Fatal compile-time errors. This is like an E\_ERROR generated by the Zend Scripting Engine |
| 128 | E\_COMPILE\_WARNING | Non-fatal compile-time errors. This is like an E\_WARNING generated by the Zend Scripting Engine |
| 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() |
| 2048 | E\_STRICT | Run-time notices. PHP suggest changes to your code to help interoperability and compatibility of the code |
| 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, except level E\_STRICT (E\_STRICT will be part of E\_ALL as of PHP 6.0) |

## 例子

任意数目的以上选项都可以用“或”来连接（用 OR 或 |），这样可以报告所有需要的各级别错误。例如，下面的代码关闭了用户自定义的错误和警告，执行了某些操作，然后恢复到原始的报错级别：

<?php

//禁用错误报告

error\_reporting(0);

//报告运行时错误

error\_reporting(E\_ERROR | E\_WARNING | E\_PARSE);

//报告所有错误

error\_reporting(E\_ALL);

?>

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

# restore\_error\_handler() 函数

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

## 定义和用法

restore\_error\_handler() 函数恢复之前的错误处理程序，该程序是由 set\_error\_handler() 函数改变的。

该函数永远返回 true。

### 语法

restore\_error\_handler()

## 提示和注释

提示：之前的错误处理程序可能是在错误处理程序或用户自定义函数中构建的。

## 例子

<?php

//custom error handler function

function customError($errno, $errstr, $errfile, $errline)

{

echo "<b>Custom error:</b> [$errno] $errstr<br />";

echo " Error on line $errline in $errfile<br />";

}

//set user-defined error handler

set\_error\_handler("customError");

$test=2;

//trigger error

if ($test>1)

{

trigger\_error("A custom error has been triggered");

}

//restore built-in error handler

restore\_error\_handler

//trigger error again

if ($test>1)

{

trigger\_error("A custom error has been triggered");

}

?>

输出：

**Custom error:** [1024] A custom error has been triggered

Error on line 14 in C:\webfolder\test.php

**Notice**: A custom error has been triggered in

**C:\webfolder\test.php** on line **21**

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

# restore\_exception\_handler() 函数

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

## 定义和用法

restore\_exception\_handler() 函数恢复之前的异常处理程序，该程序是由 set\_exception\_handler() 函数改变的。

该函数永远返回 true。

### 语法

restore\_exception\_handler()

## 提示和注释

提示：之前的异常处理程序可能是在异常处理程序或用户自定义函数中构建的。

## 例子

<?php

restore\_exception\_handler();

throw new Exception('Uncaught Exception occured');

?>

输出：

**Fatal error**: Uncaught exception 'Exception' with message

'Uncaught Exception occured' in C:\webfolder\test.php:4

Stack trace: #0 {main} thrown in **C:\webfolder\test.php** on line **4**

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

# set\_error\_handler() 函数

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

## 定义和用法

set\_error\_handler() 函数设置用户自定义的错误处理函数。

该函数用于创建运行时期间的用户自己的错误处理方法。

该函数会返回旧的错误处理程序，若失败，则返回 null。

### 语法

set\_error\_handler(error\_function,error\_types)

|  |  |
| --- | --- |
| **参数** | **描述** |
| error\_function | 必需。规定发生错误时运行的函数。 |
| error\_types | 可选。规定在哪个错误报告级别会显示用户定义的错误。默认是 "E\_ALL"。 |

## 提示和注释

提示：如果使用了该函数，会完全绕过标准的 PHP 错误处理函数，如果必要，用户定义的错误处理程序必须终止 (die() ) 脚本。

注释：如果在脚本执行前发生错误，由于在那时自定义程序还没有注册，因此就不会用到这个自定义错误处理程序。

## 例子

<?php

//error handler function

function customError($errno, $errstr, $errfile, $errline)

{

echo "<b>Custom error:</b> [$errno] $errstr<br />";

echo " Error on line $errline in $errfile<br />";

echo "Ending Script";

die();

}

//set error handler

set\_error\_handler("customError");

$test=2;

//trigger error

if ($test>1)

{

trigger\_error("A custom error has been triggered");

}

?>

输出：

**Custom error:** [1024] A custom error has been triggered

Error on line 19 in C:\webfolder\test.php

Ending Script

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

# set\_exception\_handler() 函数

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

## 定义和用法

set\_exception\_handler() 函数设置用户自定义的异常处理函数。

该函数用于创建运行时期间的用户自己的异常处理方法。

该函数会返回旧的异常处理程序，若失败，则返回 null。

### 语法

set\_exception\_handler(exception\_function)

|  |  |
| --- | --- |
| **参数** | **描述** |
| error\_function | 必需。规定未捕获的异常发生时调用的函数。  该函数必须在调用 set\_exception\_handler() 函数之前定义。  这个异常处理函数需要需要一个参数，即抛出的 exception 对象。 |

## 提示和注释

提示：在这个异常处理程序被调用后，脚本会停止执行。

## 例子

<?php

function myException($exception)

{

echo "<b>Exception:</b> " , $exception->getMessage();

}

set\_exception\_handler('myException');

throw new Exception('Uncaught Exception occurred');

?>

输出：

**Exception:** Uncaught Exception occurred

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

# set\_error\_handler() 函数

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

## 定义和用法

set\_error\_handler() 函数设置用户自定义的错误处理函数。

该函数用于创建运行时期间的用户自己的错误处理方法。

该函数会返回旧的错误处理程序，若失败，则返回 null。

### 语法

set\_error\_handler(error\_function,error\_types)

|  |  |
| --- | --- |
| **参数** | **描述** |
| error\_function | 必需。规定发生错误时运行的函数。 |
| error\_types | 可选。规定在哪个错误报告级别会显示用户定义的错误。默认是 "E\_ALL"。 |

## 提示和注释

提示：如果使用了该函数，会完全绕过标准的 PHP 错误处理函数，如果必要，用户定义的错误处理程序必须终止 (die() ) 脚本。

注释：如果在脚本执行前发生错误，由于在那时自定义程序还没有注册，因此就不会用到这个自定义错误处理程序。

## 例子

<?php

//error handler function

function customError($errno, $errstr, $errfile, $errline)

{

echo "<b>Custom error:</b> [$errno] $errstr<br />";

echo " Error on line $errline in $errfile<br />";

echo "Ending Script";

die();

}

//set error handler

set\_error\_handler("customError");

$test=2;

//trigger error

if ($test>1)

{

trigger\_error("A custom error has been triggered");

}

?>

输出：

**Custom error:** [1024] A custom error has been triggered

Error on line 19 in C:\webfolder\test.php

Ending Script

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

# set\_exception\_handler() 函数

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

## 定义和用法

set\_exception\_handler() 函数设置用户自定义的异常处理函数。

该函数用于创建运行时期间的用户自己的异常处理方法。

该函数会返回旧的异常处理程序，若失败，则返回 null。

### 语法

set\_exception\_handler(exception\_function)

|  |  |
| --- | --- |
| **参数** | **描述** |
| error\_function | 必需。规定未捕获的异常发生时调用的函数。  该函数必须在调用 set\_exception\_handler() 函数之前定义。  这个异常处理函数需要需要一个参数，即抛出的 exception 对象。 |

## 提示和注释

提示：在这个异常处理程序被调用后，脚本会停止执行。

## 例子

<?php

function myException($exception)

{

echo "<b>Exception:</b> " , $exception->getMessage();

}

set\_exception\_handler('myException');

throw new Exception('Uncaught Exception occurred');

?>

输出：

**Exception:** Uncaught Exception occurred

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

# trigger\_error() 函数

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)

## 定义和用法

trigger\_error() 函数创建用户定义的错误消息。

trigger\_error() 用于在用户指定的条件下触发一个错误消息。它与内建的错误处理器一同使用，也可以与由 set\_error\_handler() 函数创建的用户自定义函数使用。

如果指定了一个不合法的错误类型，该函数返回 false，否则返回 true。

### 语法

trigger\_error(error\_message,error\_types)

|  |  |
| --- | --- |
| **参数** | **描述** |
| error\_message | 必需。规定错误消息。长度限制为 1024 个字符。 |
| error\_types | 可选。规定错误消息的错误类型。 可能的值：   * E\_USER\_ERROR * E\_USER\_WARNING * E\_USER\_NOTICE |

## 例子

<?php

$test=2;

if ($test>1)

{

trigger\_error("A custom error has been triggered");

}

?>

输出：

**Notice**: A custom error has been triggered

in **C:\webfolder\test.php** on line **6**

[PHP Error 和 Logging 函数](http://www.w3school.com.cn/php/php_ref_error.asp)