# PHP的错误级别

首先需要了解php有哪些错误。截至到php5.5，一共有16个错误级别

http://php.net/manual/zh/errorfunc.constants.php

修改错误报告级别的方法

1. 在程序里直接直接错误的报告级别

```
error_reporting(E_ALL ^ E_NOTICE);
```

2. 修改php.ini文件中的 error_reporting

```
error_reporting =  E_ALL & ~E_NOTICE & ~E_WARNING
```

## E_ERROR

这种错误是致命错误，会在页面显示Fatal Error， 当出现这种错误的时候，程序就无法继续执行下去了

错误示例：

```
// Fatal error: Call to undefined function hpinfo() in /tmp/php/index.php on line 5
hpinfo();  //E_ERROR
```

注意，如果有未被捕获的异常，也是会触发这个级别的。

```
// Fatal error: Uncaught exception 'Exception' with message 'test exception' in /tmp/php/index.php:5 Stac
k trace: #0 {main} thrown in /tmp/php/index.php on line 5
throw new \Exception("test exception");
```

## E_WARNING

这种错误只是警告，不会终止脚本，程序还会继续进行，显示的错误信息是Warning。比如include一个不存在的文件。

```
//Warning: include(a.php): failed to open stream: No such file or directory in /tmp/php/index.php on line
 7
//Warning: include(): Failed opening 'a.php' for inclusion (include_path='.:/usr/share/pear:/usr/share/ph
p') in /tmp/php/index.php on line 7
include("a.php"); //E_WARNING
```

## E_NOTICE （ 可以忽略 ）

这种错误程度更为轻微一些，提示你这个地方不应该这么写。这个也是运行时错误，这个错误的代码可能在其他地方没有问题，只是在当前上下文情况下出现了问题。

比如$b变量不存在，我们把它赋值给另外一个变量

```
//Notice: Undefined variable: b in /tmp/php/index.php on line 9
$a = $b; //E_NOTICE
```

## E_PARSE

这个错误是编译时候发生的，在编译期发现语法错误，不能进行语法分析。

比如下面的z没有设置为变量。

```
// Parse error: syntax error, unexpected '=' in /tmp/php/index.php on line 20
z=1; // E_PARSE
```

## E_STRICT

这个错误是PHP5之后引入的，你的代码可以运行，但是不是PHP建议的写法。

比如在函数形参传递++符号

```
// Strict Standards: Only variables should be passed by reference in /tmp/php/index.php on line 17
function change (&$var) {
  $var += 10;
}

$var = 1;
change(++$var);
// E_STRICT
```

## E_RECOVERABLE_ERROR

这个级别其实是ERROR级别的，但是它是期望被捕获的，如果没有被错误处理捕获，表现和E_ERROR是一样的。

经常出现在形参定义了类型，但调用的时候传入了错误类型。它的错误提醒也比E_ERROR的fatal error前面多了一个Catachable的字样。

```
//Catchable fatal error: Argument 1 passed to testCall() must be an instance of A, instance of B given, c
alled in /tmp/php/index.php on line 37 and defined in /tmp/php/index.php on line 33
class A {
}

class B {
}

function testCall(A $a) {
}

$b = new B();
testCall($b);
```

## E_DEPRECATED

这个错误表示你用了一个旧版本的函数，而这个函数后期版本可能被禁用或者不维护了。

比如curl的CURLOPT_POSTFIELDS使用\@FILENAME来上传文件的方法

```
// Deprecated: curl_setopt(): The usage of the @filename API for file uploading is deprecated. Please use
 the CURLFile class instead in /tmp/php/index.php on line 42
$ch = curl_init("http://www.remotesite.com/upload.php");
curl_setopt($ch, CURLOPT_POSTFIELDS, array('fileupload' => '@'. "test"));
```

## E_CORE_ERROR, E_CORE_WARNING

这两个错误是由PHP的引擎产生的，在PHP初始化过程中发生。

## E_COMPILE_ERROR, E_COMPILE_WARNING

这两个错误是由PHP引擎产生的，在编译过程中发生。

## E_USER_ERROR, E_USER_WARNING, E_USER_NOTICE, E_USER_DEPRECATED,

这些错误都是用户制造的，使用trigger_error，这里就相当于一个口子给用户触发出各种错误类型。这个是一个很好逃避try catch异常的方式。

```
trigger_error("Cannot divide by zero", E_USER_ERROR);
// E_USER_ERROR
// E_USER_WARING
// E_USER_NOTICE
// E_USER_DEPRECATED
```

## E_ALL

E_STRICT出外的所有错误和警告信息。

## 错误控制

php中有很多配置和参数是可以控制错误，以及错误的日志显示的。第一步，我们需要了解的是php中的有关错误的配置有哪些？

我们按照php+php-fpm的模型来说，会影响php错误显示的其实是有两个配置文件，一个是php本身的配置文件php.ini，另外一个是php-fpm的配置文件，php-fpm.conf。

## php.ini中的配置

```
error_reporting = E_ALL  // 报告错误级别，什么级别的
```

```
error_log = /tmp/php_errors.log // php中的错误显示的日志位置
display_errors = On // 是否把错误展示在输出上，这个输出可能是页面，也可能是stdout
display_startup_errors = On // 是否把启动过程的错误信息显示在页面上，记得上面说的有几个Core类型的错误是启动时候发生的，这个就是控制这些错误是否显示页面的。
log_errors = On // 是否要记录错误日志
log_errors_max_len = 1024 // 错误日志的最大长度
ignore_repeated_errors = Off // 是否忽略重复的错误
track_errors = Off // 是否使用全局变量$php_errormsg来记录最后一个错误
xmlrpc_errors = 0 //是否使用XML-RPC的错误信息格式记录错误
xmlrpc_error_number = 0 // 用作 XML-RPC faultCode 元素的值。
html_errors = On  // 是否把输出中的函数等信息变为HTML链接
docref_root = http://manual/en/ // 如果html_errors开启了，这个链接的根路径是什么
fastcgi.logging = 0 // 是否把php错误抛出到fastcgi中
```

我们经常会被问到，error_reporting和display_errors有什么区别呢？这两个函数是完全不一样的。
PHP默认是会在日志和标准输出（如果是fpm模式标准输出就是页面）
error_reporting的参数是错误级别。表示什么样子的级别才应该触发错误。如果我们告诉PHP，所有错误级别都不需要触发错误，那么，不管是日志，还是页面，都不会显示这个错误，就相当于什么都没有发生。
display_errors是控制是否要在标准输出展示错误信息
log_errors则是控制是否要在日志中记录错误信息。

error_log是显示错误日志的位置，这个在php-fpm中往往会被重写，于是往往会发现的是cli和fpm的错误日志竟然不是在同一个文件中。

ignore_repeated_errors这个标记控制的是如果有重复的日志，那么就只会记录一条，比如下面的程序：

```
error_reporting(E_ALL);
ini_set('ignore_repeated_errors', 1);
ini_set('ignore_repeated_source', 1);

$a = $c; $a = $c; //E_NOTICE
//Notice: Undefined variable: c in /tmp/php/index.php on line 20
```

本来会出现两次NOTICE的，但是现在，只会出现一次了...

track_errors开启会把最后一个错误信息存储到变量里面去，这个可能在对记日志的时候会有一些用处吧。不过我觉得真是没啥用...

html_errors 和 docref_root 两个是个挺有人性化的配置，配置了这两个参数以后，我们返回的错误信息中如果有一些在文档中有的信息，就会变成链接形式。

```
error_reporting(E_ALL);
ini_set('html_errors', 1);
ini_set('docref_root', "https://secure.php.net/manual/zh/");

include("a2.php"); //E_WARNING
```

页面显示：

Warning: include(a2.php) [function.include]: failed to open stream: No such file or directory in /tmp/php/index.php on line 18

Warning: include() [function.include]: Failed opening 'a2.php' for inclusion (include_path='.:/usr/share/pear:/usr/share/php') in /tmp/php/index.php on line 18