Atitit 增强代码健壮性 出错继续执行恢复模式，就像vbs那样

我以为我可以使用Try/Catch，但是我找不到异常后是否可以继续执行代码，并且找不到如何在最后显示错误消息。

目录

[PHP错误级别 1](#_Toc19906)

[PHP错误相关函数 2](#_Toc31871)

[PHP异常 3](#_Toc26793)

[PHP异常与错误的抛出 3](#_Toc9085)

[PHP错误处理 4](#_Toc31675)

[PHP异常处理 4](#_Toc1218)

[在程序中PHP异常的自动抛出 5](#_Toc28825)

PHP遇到错误能继续执行的，除非是严重的语法错误。

@错误控制操作符可以实现这样的功能。  
@符号可以忽略错误,有抑制错误的功能。

如果连接数据库不成功的，前面的“@”就能把错误显示给抑制住，也就是不会显示错误，然后再抛出异常，显示自己定义的异常处理，建议最好少用，这样会增加一定的系统开销。

## PHP错误级别

Parse error > Fatal Error > Waning > Notice > Deprecated

Deprecated 最低级别的错误(不推荐，不建议)  
使用一些过期函数的时候会出现，程序继续执行

Notice 通知级别的错误  
使用一些未定义变量、常量或者数组key没有加引号的时候会出现，程序继续执行

Waning 警告级别的错误  
程序出问题了，需要修改代码！！！程序继续执行

Fatal Error 错误级别的错误  
程序直接报错，需要修改代码！！！中断程序执行，可使用register\_shutdown\_function()函数在程序终止前触发一个函数

Parse error 语法解析错误  
语法检查阶段报错，需要修改代码！！！中断程序执行，除了修改ini文件，将错误信息写到日志中，什么也做不了

E\_USER\_相关的错误  
用户定义的错误，用户手动抛出错误，进行自定义错误处理

## PHP错误相关函数

ini\_set('display\_errors', 0); //关闭错误输出(开发环境开启，生产环境关闭)

error\_reporting(E\_ALL&~E\_NOTICE); //设置错误报告级别

ini\_set('error\_reporting',0); //设置错误报告级别

## PHP异常

PHP的异常是后来新增特性，与JAVA/C#的异常不同，PHP异常需要手动抛出throw new Exception，而不是系统自动抛出

PHP错误与异常的区别，他们是2个不同的概念，但有共同的地方：

如果异常不捕获处理，程序将会终止，并报出Fatal Error 错误，看到这里大家就会觉得异常是不是错误的一种，这是一种错觉，但这样理解也可以。但异常捕获后程序可以继续执行，而真正的Fatal Error错误出现后程序就必须终止

异常可以使用 try{}catch(){} 来捕获捕获，捕获之后后续代码可以继续执行；而错误是无法使用 try{}catch(){} 捕获的

如果抛出了异常，就必须捕获它,否则程序终止执行。

## PHP异常与错误的抛出

异常抛出：throw new Exception('Some Error Message');

错误抛出：trigger\_error()

trigger\_error() 触发的错误不会被 try-catch 异常捕获语句捕获

## PHP错误处理

set\_error\_handler()

只能处理Deprecated、Notice、Waning这三种级别错误，而且处理后，脚本将会继续执行发生错误的后一行

register\_shutdown\_function()

这个方法是脚本结束前的最后一个回调函数，所以无论是die()/错误（异常）/还是脚本正常结束都会调用

## PHP异常处理

set\_exception\_handler()

设置默认的异常处理程序，有try/catch捕获的话这个函数就不会执行，反之就会执行，而且执行的话，脚本将不会继续执行发生异常的后一行代码，程序马上中止

set\_exception\_handler()注意事项

set\_exception\_handler(“myException”) 不仅可以接受****函数名****，还可以接受 ****类的方法****（公开的静态方法 及 公开的非静态方法 都可以），但需要以 ****数组形式**** 传递，数组的第一值为“****类名****”，第二个参数为“****方法名****”，如下代码所示：

## 在程序中PHP异常的自动抛出

由于PHP异常是后面版本新增的特性，设计上与JAVA/C#的异常不一样，JAVA的异常大部分是系统自动抛出，而PHP异常不是系统自动抛出，需要手动抛出，导致PHP异常在程序中的作用减半(异常就是意料之外的事情，根本我们意料不到的，如果用手动抛出，证明已经预先预料到了，那异常的意义就变味了)

在PHP中异常是手动抛出的，而错误是系统自动抛出的（也可手动抛）

我们需要把异常做成系统自动抛出接（例如JAVA）就必须****借助****错误（这三种错误Deprecated、Notice、Waning，其他的错误不行，因为会终止程序运行）

在PHP7之后，出现了一个异常与错误通用的接口Throwable，Exception类与Error类都实现了该接口，导致Error类或Error类的派生类的错误对象（大部分Fatel Error,而之前三类错误不变）也可以像Exception一样被捕获（2种捕获方法：1、try/catch 2、set\_exception\_handler（））

示例代码

**try**{

go();*//该函数未定义*

}**catch**(Exception $e){

*//捕获异常*

}**catch**(Error $er){

*//捕获错误*

}