循环的中断

循环中,有两种中断语句可以使用:

Break:

用于完全终止某个循环,让执行流程进入循环语句后面的语句;

Continue:

用于停止当前卡在进行的当次循环,而进入到循环的”下一次”过程中去(通常就是循环的开始位置);

载入语句的区别

失败时的区别

Include 报一个”提示错误”,然后继续执行后续代码;

Require 报错,并立即终止执行;

通常,require 用于程序中,后续代码依赖于载入文件的时候;

Include_once 和 require_once 会有一个内部判断机制,如果代码已经载入过,就不会重复载入,只会执行一次;

错误报告的显示问题

是否显示错误报告(display_errors):

有 2 种做法可以来设定是否显示:

方法 1:

在 php.ini 文件中,设定 display_erros 的值为 on(显示),或为 off(不显示);

注意:前提条件都是我们 apache 已经装载了 php.ini 文件,

方法 2:

直接在 php 的脚本文件中使用函数 ini_set()来对其进行设置;

语法形式:

Ini_set(“display_errors”,0);

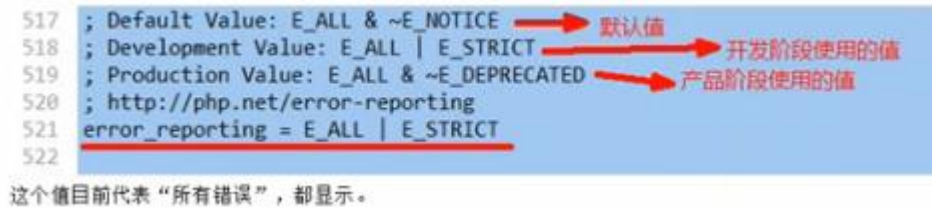
如果设置为”1”,就是显示!

显示哪些级别的错误报告(error_reporting):

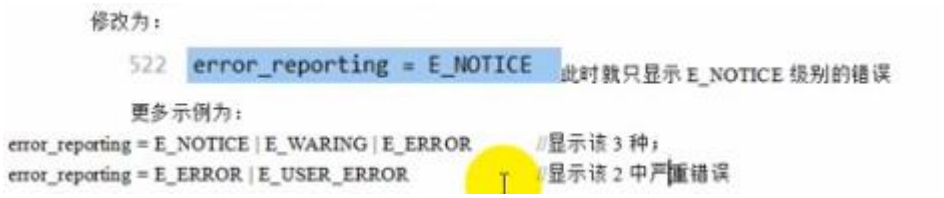
两个方法

方法 1:

在 php.ini 找到 error_reporting;



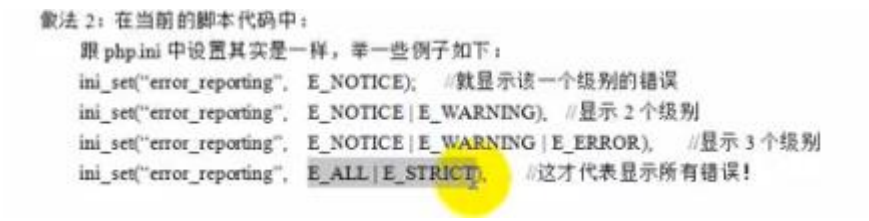
这个值目前代表“所有错误”，都显示。



方法:在当前的脚本代码中:

语法形式:

Ini_set(“error_reporting”,E_ALL|E_STRICT);



错误日志的记录

是否记录 log_error

Php.ini 中

Log_errrors = On 或 Off

脚本中

Ini_set(“log_errors”,1) 或 0;

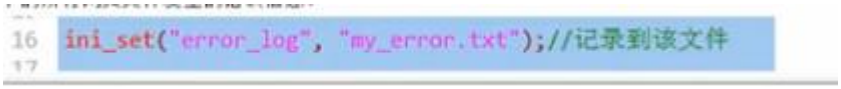
补充

1: ini_set(“php 配置项”,值) // 用于脚本中设置 php.ini 中某项的值;

2: $v1= ini_get(“php 配置项”); // 用于获取 php.ini 中某项的值;

记录到哪里 error_log

写法 1: 直接使用一个文件名,此时系统会自动在每个文件夹下都建立该文件名,并用其记录该文件夹下的所有网页文件发生的错误信息.



写法 2: 使用一个特殊的名字”syslog”,则此时所有错误信息都会记录到系统日志中.

自定义错误处理

1．设定要用于错误的函数名:

set_error_handler(“my_error_handler”) ;

2. 定义一个函数:

Function my_error_handler (){

}

```php
//我们准备要自己来定义错误"处理器"了：
//第1步：设定要作为错误处理的函数名：
set_error_handler("my_error_handler");
//第2步：定义该函数：
//该函数需要定义4个形参，分别代表：
//$errCode：代表错误代号（级别）
//$errMsg：  代表错误信息内容
//$errFile：代表发生错误的文件名
//$errLine：代表发生错误的行号
//注意，该函数我们不要在程序中调用，而是，一发生错误就会被自动调用
//而且会传入该4个实参数据
function my_error_handler($errCode, $errMsg, $errFile, $errLine){
    $str = "";
    $str .= "<p><font color='red'>大事不好，发生错误：</font>";
    $str .= "<br />错误代号为：" . $errCode;
    $str .= "<br />错误内容为：" . $errMsg;
    $str .= "<br />错误文件为：" . $errFile;
    $str .= "<br />错误行号为：" . $errLine;
    $str .= "<br />发生时间为：" . date("Y-d-m H:i:s");
    $str .= "</p>";
    echo $str;  //输出该"构建"的错误完整处理结果
    //也可以将该内容"写入"到某个文件中去，也就是所谓记录错误日志！
    //但，今天我们就做不到了—这涉及到文件操作！
}
```