错误处理

1. 错误分类

(1) 语法错误：很显然的，程序运行前都需要通过php引擎检查语法，有错误会立即报错，并且不会去执行

(2) 运行时错误：就是在程序语法检查通过后，php程序运行起来的过程中遇到的错误，这种错误包括三种情况：

1) 提示性错误：

2) 警告性错误：

3) 致命错误：

(3) 逻辑错误

指的是程序本身没问题，但是结果算错了，如前端请求回来给产品，一看，发现不对，后台某某数据肯定算错了！

2. 运行时错误分级

php语言中，将各种错误进行了不同级别的分类归纳，并形成大约有10几个级别的错误，这就是技术层面的错误分级。每一级别的错误，都有一个代号，其实就是系统内部的“常量”，比如：

常见错误：

E_ERRR：致命性错误

E_WARNIN：警告性错误

E_NOTICE：提示性错误

用户可自定义的错误：

E_USER_ERROR：自定义致命性错误

E_USER_WARNING：自定义警告性错误

E_USER_NOTICE：自定义提示性错误

其他错误：

E_STRICT:严谨性语法检查错误

E_ALL:代表所有错误

下面看看这些代号的实际常量数值

```php
function GetBinStr($e){
    $s = decbin($e);//这是一个二进制数字字符串
    //str_pad($str1, 长度n, $str2, 位置w)函数的作用是:
    //将字符串$str1,用字符串$str2填充到指定的长度n,
    //而且可以指定填充的位置w: 左边填充还是右边填充
    $s1 = str_pad($s, 16, "0", STR_PAD_LEFT);
    return $s1;
}
echo "<pre>";
echo "<br />E_ERROR=" . E_ERROR . ", \t\t其对应二进制值为: " . GetBinStr(E_ERROR);
echo "<br />E_WARNING=" . E_WARNING. ", \t\t其对应二进制值为: " . GetBinStr(E_WARNING);
echo "<br />E_NOTICE=" . E_NOTICE. ", \t\t其对应二进制值为: " . GetBinStr(E_NOTICE);
echo "<br />E_USER_NOTICE=" . E_USER_NOTICE. ", \t其对应二进制值为: " . GetBinStr(E_USER_NOTICE);
echo "<br />E_ALL=" . E_ALL. ", \t\t其对应二进制值为: " . GetBinStr(E_ALL);
echo "</pre>";
```

运行结果为：

```
E_ERROR=1,              其对应二进制值为: 0000000000000001
E_WARNING=2,            其对应二进制值为: 0000000000000010
E_NOTICE=8,             其对应二进制值为: 0000000000001000
E_USER_NOTICE=1024,     其对应二进制值为: 0000010000000000
E_ALL=30719,            其对应二进制值为: 0111011111111111
```

3. 错误的触发

就是让错误触发，有两种触发方式：

(1) 系统触发

程序运行到某行代码，确实出现某种错误，此时系统会报错，这就是触发了系统错误，有以下三种：

1) 致命错误：fatal error导致程序无法继续执行，如：调用了未定义的函数

2) 提示性错误：会输出错误提示，并继续执行后续代码比如使用了不存在的常量或变量

3) 警告性错误:会输出错误提示,并继续执行后续代码(也可能看具体情况，如：require一个不存在的文件)

(2) 自定义触发

当我们处理某些数据的时候，本来数据本身是没有错误的，但根据业务逻辑需要，会要求数据满足某种条件，而该数据并不满足的时候，我们就可以在程序中主动去触发(创建)一个错误，以表明该数据的非法性

示例：年龄0-120，当年龄为800，我们就触发自定义错误

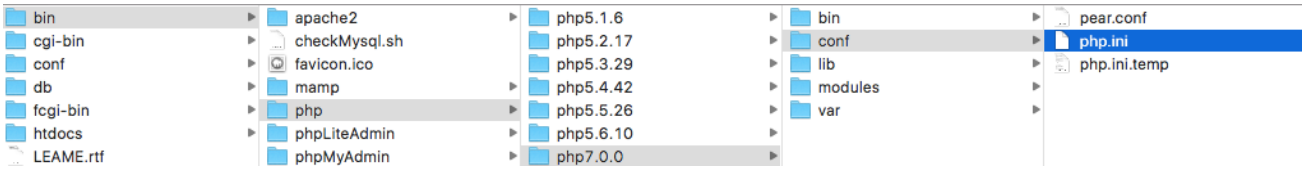
```php
$age = 800;
if($age > 120){
    trigger_error("年龄不能超过120! ", E_USER_ERROR);
}else{
    echo "你的年龄为: $age";
}

echo("FAFA");
```

4. 错误报告显示问题：

(1) 是否显示错误报告设置：mac下不输出错误日志，这是因为默认错误日志被关闭了

需要做如下设置开启：



这到这个文件，打开搜索error_log，然后将开关置为yes，重启apache服务后重新运行下即可。

还有另外一种形式设置，如下：php文件中

ini_set(“display_errors”, 1);

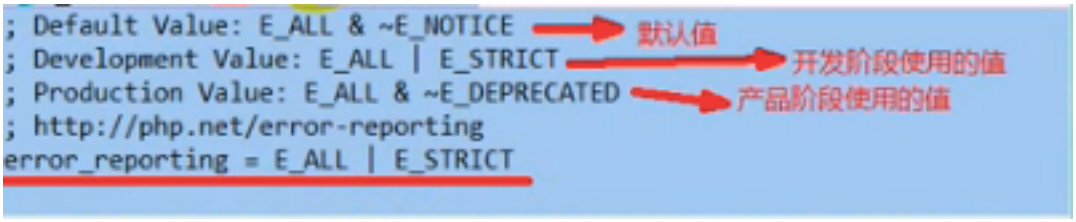
(2) 显示哪些错误报告设置

显然，前提是“display_errors”中设置为on，表示可以显示

显示哪些级别的错误报告也有两种做法：

做法1:在ini文件中

error_reporting = EALL | E_STRICT



这个值代表所有错误都显示，修改为E_ERROR，则只显示致命错误

做法2：在php文件中，还是通过ini_set函数来设置

ini_set(“error_reporting”, E_NOTICE);

5. 错误日志记录问题

错误日志其实就是错误报告，只是它会写入到文件中，此时就称为错误日志！

也有两个问题，2种做法：

(1) 是否记录：

php.ini中：

log_errors = On

脚本中：

ini_set(“log_errors”,1);

补充一句：

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

$v1 = ini_get(“php配置项”); //用于获取脚本设置中php.ini中某项的值

(2) 记录到哪里：

一般来说只有两个写法：

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面，都建立该文件名，并用其记录该目录下所有网页文件中所产生的所有错误日志：

写法2:使用一个特殊名字syslog，则此时所有错误信息都会记录到系统的错误日志中

示例：

ini_set(“error_log”, “my_error.txt”);

ini_set(“error_log”, “syslog”);

6. 自定义错误处理器

什么叫错误处理器？

一旦发生错误，用来处理该错误的一个函数

自定义错误处理，就是指

让系统不去处理错误了，完全由我们开发者来对错误进行处理(显示和记录)

做法很简单，只要两步：

(1) 设定要用于处理的函数名

set_error_handler(“my_error_handler”);

(2) 去定义该函数

function f1(){

//这里可以写任意代码：自然是去显示和记录错误日志

}

```php
<?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'>太卑不好，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 .= "</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！
```