## 在被载入文件中 return 语句的作用

1，一个载入语句，如果载入成功，其实是有返回值的，为 1，如果载入失败，则返回的是 false。（虽然我们通常不去使用该返回值）。
但，如果被载入文件中有 return 语句，此时就有另外的机制和作用：
2，return 语句此时的作用是终止载入过程——该 return 语句的后续被载入文件的代码不再载入。
3，return 语句也可以用于该被载入文件载入时返回一个数据，形式为：return　XX 数据；

# 错误处理

## 错误的分类

通常分 3 种：

语法错误：
　　程序运行之前，都要先检查语法。如果语法有错误，就会立即报错，并且不会去执行程序。

运行时错误：
　　就是在程序语法检查通过后，，开始运行程序并在此过程中遇到的错误。常见的有 3 中：
　　提示性错误：
　　警告性错误：
　　致命错误：

逻辑错误：
　　指的是，程序本身可以正常执行，没有报错——但“计算结果”却错了。

## 错误的分级

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”，这个代号其实也就是一个系统内部的“常量而已”。比如：

系统常见错误：
E_ERROR：　　致命错误
E_WARNING:　　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：
E_STRICT:　　严谨性语法检查错误
E_ALL　　代表“所有错误”。

## 错误的触发

错误的触发，就是让错误“发生”。

有两种方式会触发错误：

## 系统触发

程序运行到某行代码，确实出现了某种错误，此时系统就会报错——这就是触发了系统错误。
系统触发的典型错误有这 3 种：
E_NOTICE:　　提示性错误：会输出错误提示，并继续执行后续代码；

E_WARNING:　　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，比如 require)

E_ERROR：　　致命错误：导致程序无法执行后续语句；

## 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，但根据具体应用（业务）的需要，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”。
语法形式：
trigger_error(“错误提示信息内容”,　3 中用户错误代号之一);
其中触发了用户的致命错误(E_USER_ERROR）,也会终止程序的后续执行。

## 错误报告的显示问题

所谓错误报告，就是显示在网页上的错误提示内容！

有关错误报告，有 2 个问题需要处理：

# 是否显示错误报告（display_errors)：

有 2 种做法可以来设定是否显示：
做法 1：
在 php.ini 文件中，设定 display_erros 的值，为 on（显示），或为 off（不显示）
当然，作为开发阶段，我们都应该显示错误信息。

注意：前提条件都是我们 apache 已经装载了 php.ini 文件——这一点，需要在 apache 的配置文件 httpd.config 中加入如下一行：
PHPIniDir “php.ini 文件的了位置（路径）”

方法 2：
直接在 php 的脚本文件中设使用函数 ini_set()来对其进行设置
注意：
1，不管哪种形式，该单词是一样的：display:errors
2，使用 php.ini 配置，影响的是全局（即所有 php 网页）；
3，在某个脚本代码中使用 ini_set()设置，就只影响该脚本代码本身——这是常用的方式。
4，脚本中的设置优先于 php.ini 中的设置。

# 显示哪些级别的错误报告（error_reporting)：

显然，前提是“display_errors”设置为 On(或 1），表示可以显示。

显示哪些级别的错误报告，也有 2 个做法：
**做法 1：在 php.ini 文件中；**

**做法 2：在当前的脚本代码中：**
跟 php.ini 中设置其实是一样，举一些例子如下：
ini_set(“error_reporting”, E_NOTICE); //就显示该一个级别的错误
ini_set(“error_reporting”, E_NOTICE | E_WARNING), //显示 2 个级别
ini_set(“error_reporting”, E_NOTICE | E_WARNING | E_ERROR), //显示 3 个级别
ini_set(“error_reporting”, E_ALL | E_STRICT), //这才代表显示所有错误！

# 错误日志的记录问题

错误日志其实就是错误报告，只是它会“写入文件中”，此时就称为错误日志！

也有 2 个问题，每个问题也有 2 种做法：

# 是否记录 log_errors：

php.ini 中：

log_errors = On　　或 Off

脚本中：

ini_set(“log_erros”,　1);　　或 0

补充一句：

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

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

# 记录到哪里 error_log：

一般就只有 2 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的“日志文件”中。

系统日志文件在这里：控制面板》管理工具》事件查看器》window 日志》应用程序

# 自定义错误处理器

什么叫错误处理器？

就是一旦发生错误，用来处理该错误的一种“机器”——其实就是一个函数。

自定义错误处理，就是指：

让系统不要去处理错误了，而完全由我们（开发者）来对错误进行处理：显示和记录。

做法，其实非常简单，就 2 步：

第一步：

设定要用于处理错误的函数名！

```
set_error_handler(“f1”);
```

第二步：

去定义该函数！

```
function f1(){
    //这里可以任意写代码：自然正常是去显示错误报告，和记录错误日志。
}
```