第7章 PHP的错误和异常处理

- PHP的错误和异常处理是PHP中常用的模块之一，在开发项目的时候合理使用异常处理，将有利于发现错误和加快我们的开发速度。本章就从最基本的认识PHP的错误类型开始，再到以后的处理异常逐步深入讲解。

7.1 错误类型

+ 任何程序员在开发时候都会或多或少地有一些失误，碰到一些错误或者其他原因导致的错误。当然也有用户不愿意遵守程序的约束，也会引发一些错误。程序的错误一般分为三类：语法错误、执行时错误和逻辑错误。下面依次讲解这三种错误。

7.1.1 语法错误

+ 语法错误是我们在编程中最容易碰到也是最容易解决的一种错误。这种错误会停止程序的执行，显示出错误信息。我们可以根据错误信息改正程序重新执行即可。
+ (1)演示常见语法错误和相关的错误信息。

7.1.2 执行时错误

- 执行时错误也是就在执行的时候的错误。这种程序的语法没有错误，但是会在执行的过程中，PHP会发现程序有些不合理的地方，会提示出警告信息。并且，程序会继续向下执行。
- (1)演示把0作为除数的错误程序，以及运行时输出的错误信息。

7.1.3 逻辑错误

+ 逻辑错误是发生在程序员思想上的错误。这种代码语法错误和运行时错误都是不存在的。因此程序在执行执行中不会报出任何的错误信息，并且程序会正常执行。只是输出的结果不是我们期望的结果而已。
+ (1)演示一段逻辑错误的代码，以及不符合期望的输出结果。

7.2 异常产生

- 在运行PHP脚本的时候，PHP的解析器会尽可能地报告它遇到的问题的。而这些错误报告的行为都是与PHP的配置文件php.ini中的配置指令相关的。这个文件我们可以通过XAMPP的控制面板很容易的找到。
- 我们使用notepad++打开这个文件。

7.2 异常产生

- 另外PHP还有多种错误级别，我们可以根据不同的报告采取不同的调试方法。当然我们也是可以自己调整这些错误级别显示与否。表中所示就是PHP中大多数的错误报告级别。

错误级别	说明
E_ALL	所有的错误和警告(不包括 E_STRICT)
E_ERROR	致命的运行时错误，程序会停止运行
E_WARNING	运行时警告
E_PARSE	编译时解析错误
E_NOTICE	运行时错误提醒
E_STRICT	编码标准化警告，允许PHP建议如何修改代码以确保最佳的互操作性向前兼容性
E_CORE_ERROR	PHP启动时初始化过程中的致命错误
E_CORE_WARNING	PHP启动时初始化过程中的警告
E_COMPILE_ERROR	编译时致命性错误
E_COMPILE_WARNING	编译时警告
E_USER_ERROR	用户自定义的错误消息
E_USER_WARNING	用户自定义的警告消息
E_USER_NOTICE	用户自定义的提醒消息

7.2 异常产生

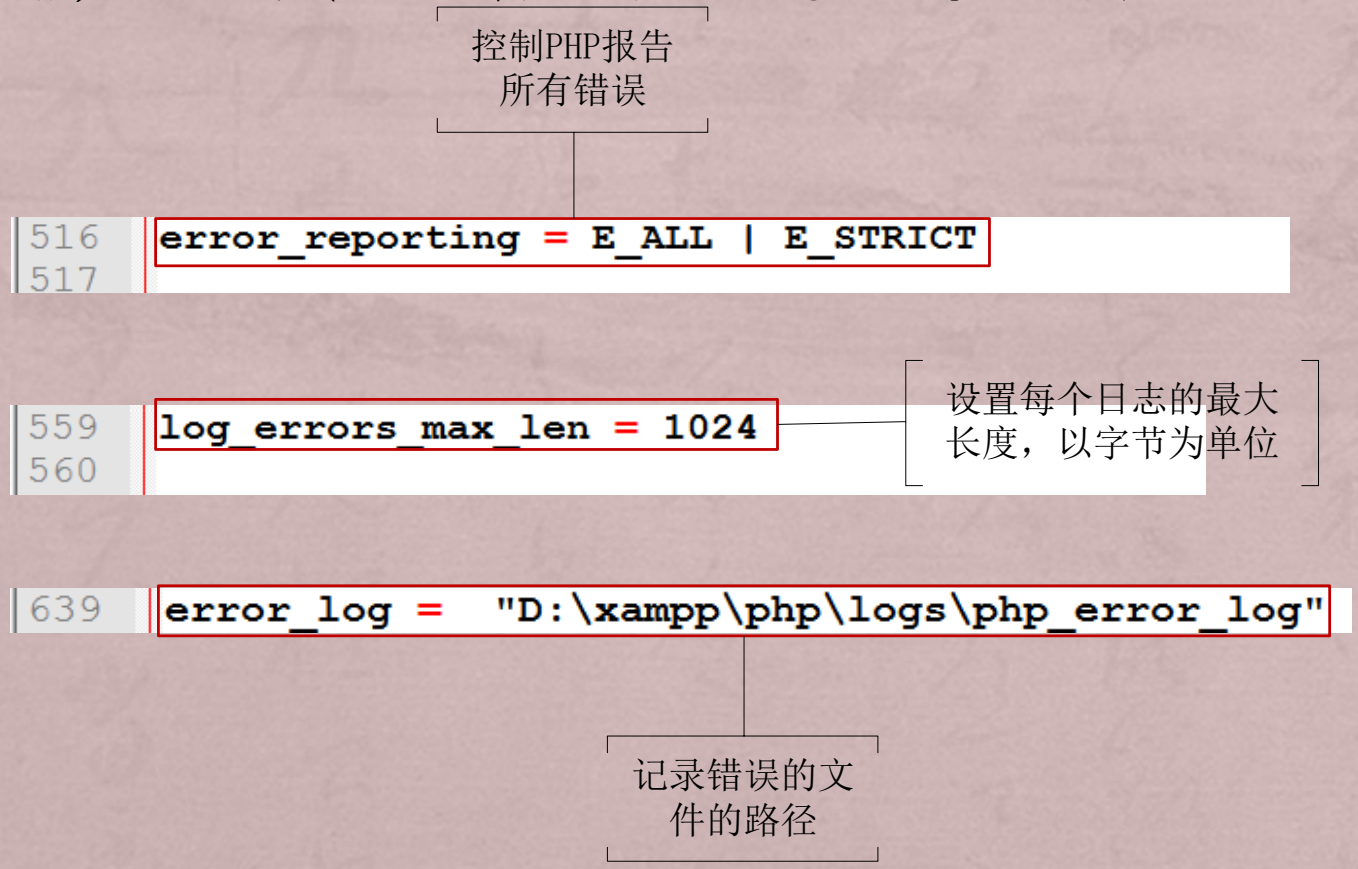
+ 这些错误级别对应php.ini的显示。
+ （1）演示使用error_reporting()关闭错误。

7.3 错误日志

+ 对于开发者来说，在开发的产品投入使用后，通常就会把所有的错误提示都关闭。因为那些错误提示对于开发者来说是好事。对于消费者来说只会影响到对产品的体验。同时也避免错误信息透露的路径、数据库链接等信息而遭到黑客攻击。但是在一个产品投入使用后，难免会出现一些错误。错误日志就可以把这些错误保存到单独的文本文件中而不会显示在浏览器中。

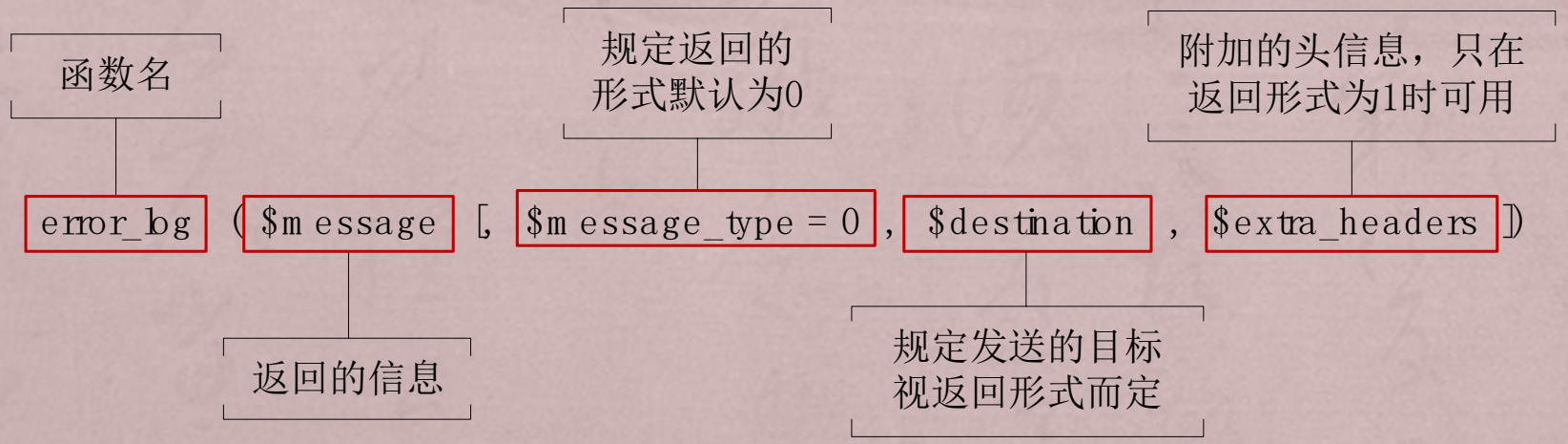
7.3.1 使用指定的文件记录错误报告日志

- 如果需要将错误信息写入文本文件中我们就要在PHP配置文件中把log_errors开启。虽然在我们使用的集成环境中是开启的。但是这里也展示一下log_errors开启后的形式。
- 在我们的集成环境中，无需我们做任何改动就已经可以实现使用指定的文件记录错误日志的了。但是在默认的PHP配置中，我们还需要知道一些地方需要做更改，如下所示就是应该修改的地方，虽然我们现在不需要去修改，但是这里我们以备读者在以后自己配置环境的时候使用。如图所示。



7.3.1 使用指定的文件记录错误报告日志

- 这样PHP的所有错误都会记录在php_error_log这个文件中了。由于自从装好环境后它就在记录日志了。因此我们当然可以打开这个文件看看它里面的内容。
- 我们可以使用error_log()来自定义错误信息。它的语法如图所示。
- 这个函数执行完毕会返回一个布尔值，因此我们可以使用判断语句判断它是否成功执行。在图中所示的语法中$message_type有如下几种形式：
- 0：把日志存放到操作系统的日志中。这是默认值。
- 1：把日志发送到$destination指定的电子邮件中。
- 3：在$destination指定的文件中不换行加入日志消息。
- 4：附加头信息。只在类型为1时可用。



7.3.1 使用指定的文件记录错误报告日志

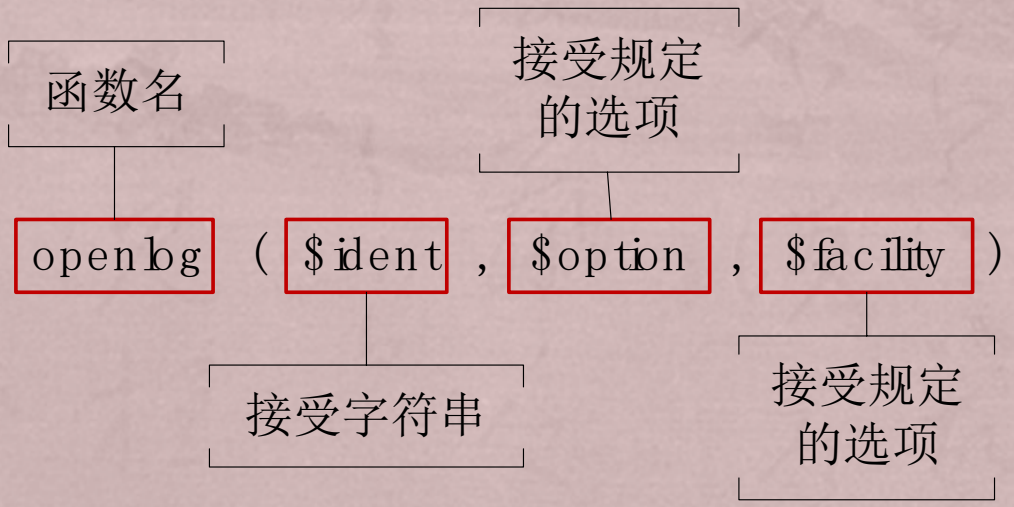
+ （1）演示把自定义日志信息加入到PHP配置中的文件中。

7.3.2 日志信息记录到操作系统日志

+ 在前以章节的学习中我们已经学习了把自定义的日志信息保存到指定文件中。本节中我们要学习的是把日志信息记录到操作系统日志中。虽然使用error_log()也可以把日志信息记录到系统日志中。但是通常情况下我们使用三个新的函数联合起来实现这个功能。使用这些函数的作用是使日志记录的信息更加详细和给开发人员更大的自定义空间。下面我们就来认识这是三个新的函数。

1.OPENLOG()打开日志连接

+ openlog()函数通过指定几个将在日志中文使用的参数，为向系统系统日志插入消息做好准备。它的语法如图所示。执行成功则返回TRUE，失败则返回FALSE。

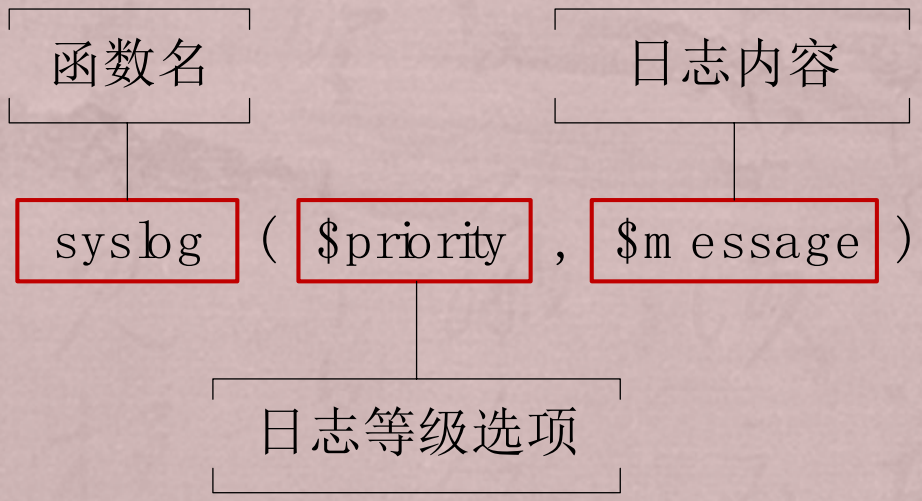


1.OPENLOG()打开日志连接

- 图中各个参数的解释如下：
- ident：增加到每一项开始处的标识符，通常设置为程序名。
- option：确定生成消息时使用哪些日志选项。
- facility：这个参数用于指定日志消息类型。

2.SYSLOG()生成日志消息

+ syslog()用于向syslog发送一条定制消息。执行成功则返回TRUE，失败则返回FALSE。它的语法如图所示。



2.SYSLOG()生成日志消息

+ 在图中的语法中，等级选项可以接受的参数如表所示。

LOG_EMERG	系统奔溃
LOG_ALERT	立即执行动作
LOG_CRIT	危险的状态
LOG_ERR	错误信息
LOG_WARNING	警告信息
LOG_NOTICE	提醒信息
LOG_INFO	报告信息
LOG_DEBUG	调试信息

3.CLOSELOG()关闭日志连接

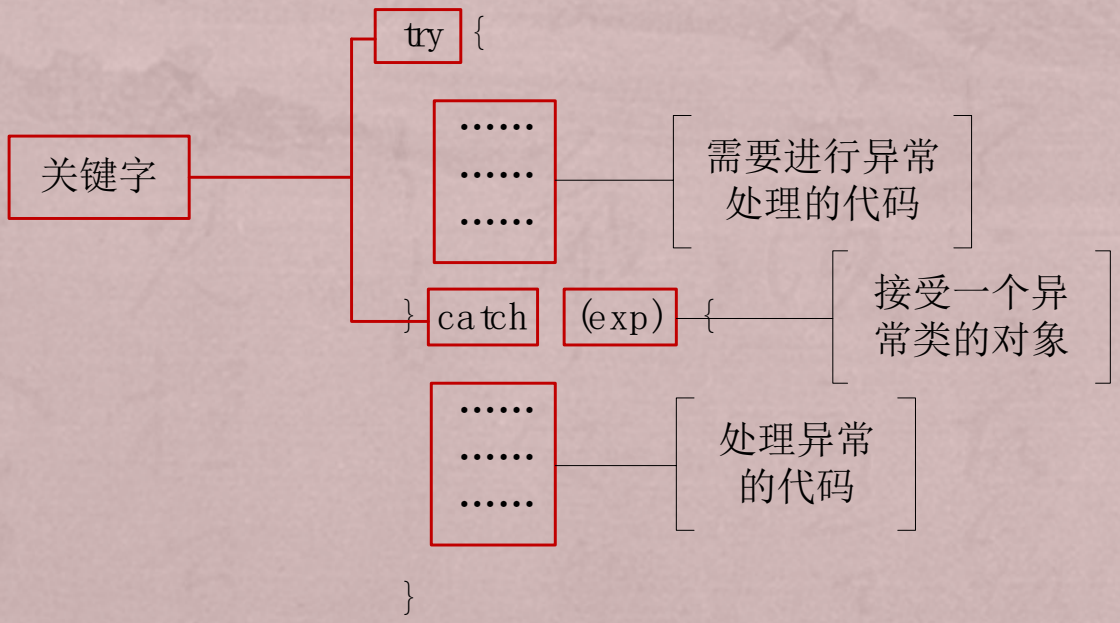
- closelog()用于关闭日志连接。它不接受任何参数，并且这个函数是可选的。如果我们不主动调用，系统也会在相关函数停止执行后自动调用closelog()。
- （1）演示使用openlog()、syslog()和closelog()将日志信息记录到操作系统。
- 我们只用最基本的三个函数完成了操作。因此浏览器不会有任何输出，但是日志已经写入到了操作系统的日志中了。我们可以来查看一下。

7.4 异常处理

+ 异常处理就是用于在指定的错误发生的时候改变程序的正常运行流程，是PHP 5中的一个新的重要的特性。异常处理是一种可扩展、易维护的错误处理统一机制，并且提供了新的面向对象的错误处理方式。

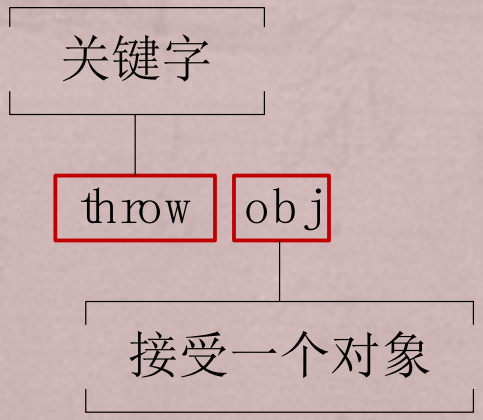
7.4.1 异常处理实现

+ 异常就是在程序执行过程中出现的一些预料之外的事件，如果不对此事件进行处理，则程序在运行过程中遇到异常将会奔溃。在PHP中使用以下语句处理异常，如图所示。



7.4.1 异常处理实现

- 在PHP代码中的异常可以被throw语句抛出而被catch语句捕获。throw是一个语句结构而不是一个函数，但必须给他传递一个对象作为值。它的语法如图所示。
- 在PHP中如果try代码块中出现了一个错误，我们就可以执行一个抛出异常的操作。PHP中的异常必须手动抛出。最简单的情况下我们可以
- （1）以下代码使用异常处理语句处理一个异常。



7.4.2 扩展PHP内置异常处理类

+ 虽然内置的异常处理类已经有非常不错的特性了，但在某些情况下需要使用更多的功能，这就需要通过扩展异常处理来解决了。由于内置的异常类是所有异常类的基类，因此我们只要通过继承异常类来扩展。我们先来看以下PHP内置异常类可以被继承的成员：

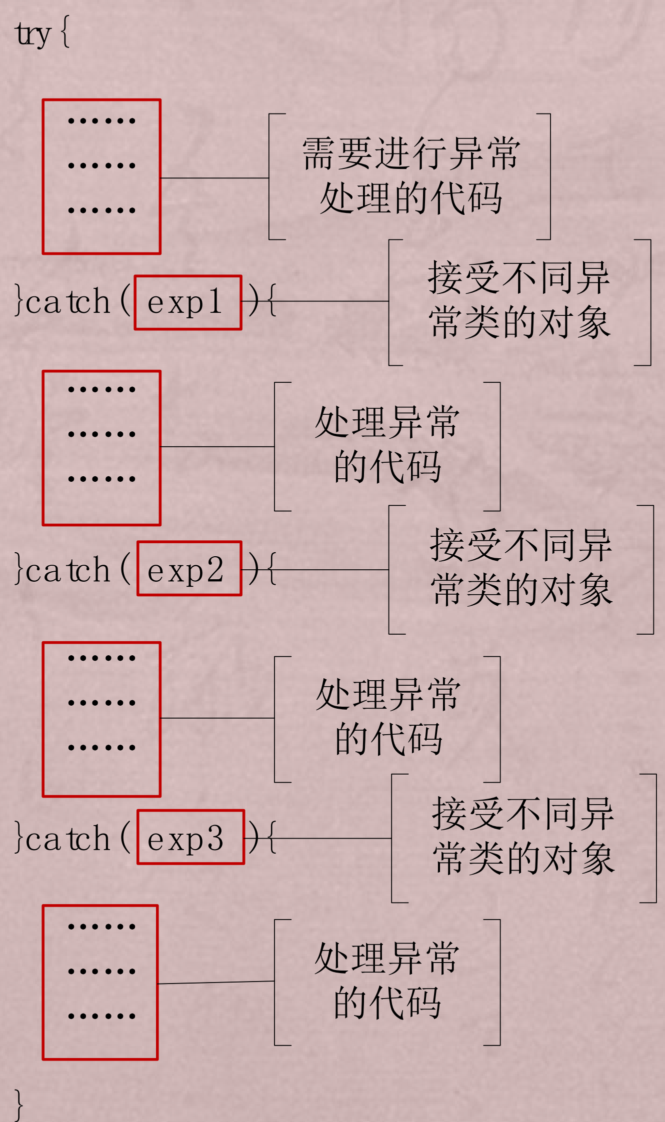
```
+ class Exception{
+     protected $message = 'Unknown exception';   // 异常信息
+     protected $code = 0;                        // 用户自定义异常代码
+     protected $file;                            // 出现异常的文件
+     protected $line;                            // 出现异常的代码所在的行
+     public function __construct($message = null, $code = 0, $previous = null){};
+     final public  function getMessage();        // 返回异常信息
+     final public  function getCode();           // 返回异常代码
+     final public  function getFile();           // 返回发生异常的文件名
+     final public  function getLine();           // 返回发生异常的代码行号
+     final public  function getTrace();          // 以数组形式返回异常传递的路线
+     final public  function getPrevious();       // 返回格式化的异常
+     final public  function getTraceAsString(); // 返回格式化成字符串的getTrace函数信息
+     public function __toString();               // 可重载，用于返回可输出的字符串
+ }
```

7.4.2 扩展PHP内置异常处理类

+ （1）通过继承内置异常类并扩展它。
+ （2）通过一条件判断语句判断是否抛出异常。
+ （3）定义$a，并输出结果。

7.4.3 捕获多个异常

- 在异常处理的语句中，虽然try语句和catch语句不可以单独出现。但是try语句是可以联合多个catch语句实现捕获多个异常的。它的语法形式如图所示。



7.4.3 捕获多个异常

+ 捕获多个异常没有新的知识，我们直接来看一个使用它的示例。
+ （1）使用异常处理语句实现捕获多个异常。

7.5 小结

+ 本章我们学习的是PHP的错误和异常处理，主要的宗旨是帮助读者了解一下PHP的错误机制，已经PHP错误环境的控制配置。以及学会简单的异常处理。总体来说内容还是比较少的，而且也没有太过难的知识点。多是我们前面所学知识的一个综合，相信读者是可以比较轻松地掌握本章知识的。还是要提醒读者，不要记代码，一定要亲手做章节后面的习题。