Các ngoại lệ được sử dụng để thay đổi dòng chảy bình thường của một kịch bản nếu một lỗi cụ thể xảy ra.

Hàm xử lý lỗi và log trong PHP

Đây là những hàm xử lý lỗi và xử lý các vấn đề về log trong PHP. Chúng cho phép bạn định nghĩa các qui tắc xử lý lỗi cho riêng bạn, cũng như chỉnh sửa cách lỗi đó có thể được log. Điều này cho phép bạn thay đổi và cải thiện việc báo cáo lỗi phù hợp với yêu cầu của bạn.

Sử dụng các hàm xử lý log này, bạn có thể gửi message một cách trực tiếp tới thiết bị khác, tới một Email, các các System log, … để bạn có thể log và giám sát các phần quan trọng của ứng dụng.

## Cấu hình Runtime

Chế độ vận hành của các hàm xử lý lỗi và xử lý log trong PHP phụ thuộc vào các cài đặt trong php.ini. Các cài đặt này được định nghĩa ở bảng sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên** | **Mặc định** | **Có thể thay đổi** | **Changelog** |
| error\_reporting | NULL | PHP\_INI\_ALL |  |
| display\_errors | "1" | PHP\_INI\_ALL |  |
| display\_startup\_errors | "0" | PHP\_INI\_ALL | Có sẵn từ PHP 4.0.3. |
| log\_errors | "0" | PHP\_INI\_ALL |  |
| log\_errors\_max\_len | "1024" | PHP\_INI\_ALL | Có sẵn từ PHP 4.3.0. |
| ignore\_repeated\_errors | "0" | PHP\_INI\_ALL | Có sẵn từ PHP 4.3.0. |
| ignore\_repeated\_source | "0" | PHP\_INI\_ALL | Có sẵn từ PHP 4.3.0. |
| report\_memleaks | "1" | PHP\_INI\_ALL | Có sẵn từ PHP 4.3.0. |
| track\_errors | "0" | PHP\_INI\_ALL |  |
| html\_errors | "1" | PHP\_INI\_ALL | PHP\_INI\_SYSTEM trong PHP <= 4.2.3. Có sẵn từ PHP 4.0.2. |
| docref\_root | "" | PHP\_INI\_ALL | Có sẵn từ PHP 4.3.0. |
| docref\_ext | "" | PHP\_INI\_ALL | Có sẵn từ PHP 4.3.2. |
| error\_prepend\_string | NULL | PHP\_INI\_ALL |  |
| error\_append\_string | NULL | PHP\_INI\_ALL |  |
| error\_log | NULL | PHP\_INI\_ALL |  |
| warn\_plus\_overloading | NULL |  | Tùy chọn này không còn có sẵn ở trong PHP 4.0.0 |

## Hằng về Error và Logging trong PHP

Cột **PHP** chỉ phiên bản PHP sớm nhất hỗ trợ hằng đó.

Bạn có thể sử dụng bất kỳ hằng nào trong khi cấu hình php.ini file của bạn.

|  |  |  |  |
| --- | --- | --- | --- |
| **Giá trị** | **Hằng** | **Miêu tả** | **PHP** |
| 1 | E\_ERROR | Fatal run-time error. Đây là các lỗi nghiêm trọng, và việc thực thi của script bị ngừng lại |  |
| 2 | E\_WARNING | Non-fatal run-time error. Đây là các lỗi không nghiêm trọng, và việc thực thi của script không bị ngừng lại |  |
| 4 | E\_PARSE | Compile-time parse error. Lỗi về Parse trong khi biên dịch, và nên chỉ được tạo bởi Parser |  |
| 8 | E\_NOTICE | Run-time notice. Script tìm thấy cái gì đó có thể là một lỗi, nhưng cũng có thể xảy ra khi đang chạy một Script một cách bình thường |  |
| 16 | E\_CORE\_ERROR | Fatal error tại quá trình khởi động PHP. Lỗi này khá giống một E\_ERROR trong PHP Core | 4 |
| 32 | E\_CORE\_WARNING | Non-fatal error tại quá trình khởi động PHP. Lỗi này khá giống một E\_WARNING trong PHP Core | 4 |
| 64 | E\_COMPILE\_ERROR | Fatal compile-time error. Các lỗi này khá giống một E\_ERROR được tạo bởi Zend Scripting Engine | 4 |
| 128 | E\_COMPILE\_WARNING | Non-fatal compile-time error. Các lỗi này khá giống một E\_WARNING được tạo bởi Zend Scripting Engine | 4 |
| 256 | E\_USER\_ERROR | Fatal user-generated error. Một lỗi nghiêm trọng do người dùng tạo, giống một E\_ERROR được thiết lập bởi lập trình viên bởi sử dụng hàm function trigger\_error() trong PHP | 4 |
| 512 | E\_USER\_WARNING | Non-fatal user-generated warning. Do người dùng tạo, giống một E\_WARNING được thiết lập bởi lập trình viên bởi sử dụng hàm trigger\_error() trong PHP | 4 |
| 1024 | E\_USER\_NOTICE | User-generated notice. Do người dùng tạo, giống một E\_NOTICE được thiết lập bởi lập trình viên bởi sử dụng hàm trigger\_error() trong PHP | 4 |
| 2048 | E\_STRICT | Run-time notice. PHP gợi ý thay đổi code của bạn để làm code có tính tích hợp và hoạt động liên kết hơn | 5 |
| 4096 | E\_RECOVERABLE\_ERROR | Catchable fatal error. Giống một E\_ERROR nhưng có thể được bắt bởi handle do người dùng định nghĩa (bạn cũng tham khảo hàm set\_error\_handler()) | 5 |
| 8191 | E\_ALL | Tất cả Error và warning, ngoại trừ E\_STRICT | 5 |

## Danh sách hàm xử lý lỗi và xử lý log trong PHP

|  |  |  |
| --- | --- | --- |
| **Hàm** | **Miêu tả** | **PHP** |
| [**Hàm debug\_backtrace()**](http://vietjack.com/php/ham_debug_backtrace_trong_php.jsp) | Tạo một Backtrace | 4 |
| [**Hàm debug\_print\_backtrace()**](http://vietjack.com/php/ham_debug_print_backtrace_trong_php.jsp) | In một Backtrace | 5 |
| [**Hàm error\_get\_last()**](http://vietjack.com/php/ham_error_get_last_trong_php.jsp) | Lấy lỗi cuối cùng đã xảy ra | 5 |
| [**Hàm error\_log()**](http://vietjack.com/php/ham_error_log_trong_php.jsp) | Gửi một lỗi tới Server Error-Log, tới một File hoặc một Remote Destination | 4 |
| [**Hàm error\_reporting()**](http://vietjack.com/php/ham_error_reporting_trong_php.jsp) | Xác định lỗi nào được báo cáo | 4 |
| [**Hàm restore\_error\_handler()**](http://vietjack.com/php/ham_restore_error_handler_trong_php.jsp) | Phục hồi Error Handler trước đó | 4 |
| [**Hàm restore\_exception\_handler()**](http://vietjack.com/php/ham_restore_exception_handler_trong_php.jsp) | Phục hồi Exception Handler trước đó | 5 |
| [**Hàm set\_error\_handler()**](http://vietjack.com/php/ham_set_error_handler_trong_php.jsp) | Thiết lập một hàm do người dùng tạo để xử lý lỗi | 4 |
| [**Hàm set\_exception\_handler()**](http://vietjack.com/php/ham_set_exception_handler_trong_php.jsp) | Thiết lập một hàm do người dùng tạo để xử lý ngoại lệ | 5 |
| [**Hàm trigger\_error()**](http://vietjack.com/php/ham_trigger_error_trong_php.jsp) | Tạo một error message được định nghĩa bởi người dùng | 4 |
| [**Hàm user\_error()**](http://vietjack.com/php/ham_user_error_trong_php.jsp) | Alias của hàm trigger\_error() | 4 |

Example #4 Exception handling with a *finally* block

<?php  
function inverse($x) {  
    if (!$x) {  
        throw new Exception('Division by zero.');  
    }  
    return 1/$x;  
}  
  
try {  
    echo inverse(5) . "\n";  
} catch (Exception $e) {  
    echo 'Caught exception: ',  $e->getMessage(), "\n";  
} finally {  
    echo "First finally.\n";  
}  
  
try {  
    echo inverse(0) . "\n";  
} catch (Exception $e) {  
    echo 'Caught exception: ',  $e->getMessage(), "\n";  
} finally {  
    echo "Second finally.\n";  
}  
  
// Continue execution  
echo "Hello World\n";  
?>

The above example will output:

0.2

First finally.

Caught exception: Division by zero.

Second finally.

Hello World