

php | tek - Chicago, May 19-23, 2009

Exceptional PHP

When Good Code Goes Bad

Jeff Moore

<http://www.mashery.com/>

<http://www.procata.com/blog/>



What Impression Does this Make?

- **Warning:** mysql_connect(): Too many connections in **/www2/anonymous/php/_Database.inc** on line **18** Too many connections
- **Database error:** Invalid SQL: INSERT INTO ...

Hide Your Errors

- Production
 - `display_errors = Off`
- Development
 - `display_errors = On`
 - `display_errors = Off`

Don't Bury Your Head in the Sand

- Production
 - `log_errors = On`
- Development
 - `log_errors = On`

Error Levels

1	<code>E_ERROR</code>	128	<code>E_COMPILE_WARNING</code>
2	<code>E_WARNING</code>	256	<code>E_USER_ERROR</code>
4	<code>E_PARSE</code>	512	<code>E_USER_WARNING</code>
8	<code>E_NOTICE</code>	1024	<code>E_USER_NOTICE</code>
16	<code>E_CORE_ERROR</code>	2048	<code>E_STRICT</code>
32	<code>E_CORE_WARNING</code>	4096	<code>E_RECOVERABLE_ERROR</code>
64	<code>E_COMPILE_ERROR</code>		

Error Reporting

- `error_reporting = E_ALL & ~E_NOTICE // Default`
- `error_reporting = E_ALL`
- `error_reporting = E_ALL | E_STRICT`

Error Suppression Operator (@)

- `@operation();`

Same as

- `$level = error_reporting(0);
operation();
error_reporting($level);`

Be Careful Not to Over Suppress

- `@include_once "file.php";`
- `@store($filename, $data)`

Prevent Instead of Suppress

- Prevent
 - `$x = isset($var) ? $var : NULL;`
- Suppress
 - `$x = @$var;`

Handleable Errors

	E_ERROR		E_COMPILE_WARNING
2	E_WARNING	256	E_USER_ERROR
	E_PARSE	512	E_USER_WARNING
8	E_NOTICE	1024	E_USER_NOTICE
	E_CORE_ERROR	2048	E_STRICT
	E_CORE_WARNING	4096	E_RECOVERABLE_ERROR
	E_COMPILE_ERROR		

Error Handler

```
function MyErrorHandler (
    $type, $message, $file, $line, $context) {

    if ( $type == E_USER_ERROR ) {
        while (ob_get_level()) {
            ob_end_clean();
        }
        echo "500 - Internal Server Error";
    }

    return FALSE; // Continue with PHP Handler
}

ob_start();
set_error_handler("MyErrorHandler")
```

Handling Fataals

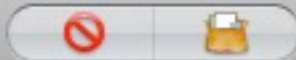
```
function myFatalHandler($buffer) {  
    global $is_complete;  
    if ($is_complete) {  
        return $buffer;  
    } else {  
        return "500 - Internal Server Error\n";  
    }  
}  
  
$is_complete = FALSE;  
ob_start("myFatalHandler");  
  
// ...  
  
$is_complete = TRUE;
```



Inbox (2214 messages, 1636 unread)



Get Mail



Delete

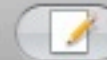
Junk



Reply

Reply All

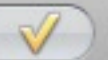
Forward



New Message



Note



To Do

MAILBOXES

- ▶ Inbox 1636
- ▶ Sent
- ▶ Trash
- ▶ Junk 140
- ▼ REMINDERS
 - Notes
- ▼ ON MY MAC
 - ▶ Archives
 - ▶ Clients
 - ▶ Lists 567
 - ▶ Notices 187
 - ▶ ToDo

#	Date Received	Mailbox	Subject
2194	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2195	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2196	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2197	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2198	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2199	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2200	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2201	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2202	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2203	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2204	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2205	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2206	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2207	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2208	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2209	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2210	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2211	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2212	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!
2213	Yesterday 2:41 PM	Inbox - Jeff	vBulletin Database Error!



Formally Defining Program Correctness

- Precondition
 - Must be true before the code can be executed
- Invariants
 - Must be true while the code is executed
- Postconditions
 - Must be true after the code is done executing

What Can Go Wrong?

- Programming errors
- Violation of preconditions
- Violation of postconditions
- Violation of invariants
- Invalid input
- Resource failures
- Hardware failures
- Who knows what else?

Defensive Programming

- Check everything, trust nothing
- Check pre-conditions and invariants
- Most errors conditions never occur
 - More code
 - Slower code
 - Error checking code can have bugs, too

Trusting Your Caller

- Trust your caller to get it right
- Document pre-conditions and invariants
- Faster code
- Leaner code
- Harder to debug

Which Approach?

- Defend
 - Defend yourself at package boundaries
 - Banking
- Trust
 - Trust within your own package
 - Don't trust another organization
 - Bookmarking

Defending Exit Points

- Every function or method call is an exit point
- Do you verify the post conditions and invariants after the call?
 - A.K.A Check return values for error codes?
- How do you indicate an error?

Context Matters

- Low Level
 - Not enough perspective to know what to do
- High Level
 - Too late and too far away to react

Is it an Error?

- `php -r "var_dump(unlink('foo'));"`
- Warning: unlink(foo): No such file or directory in Command line code on line 1
- Warning: unlink(foo): Permission denied in Command line code on line 1
- Both return FALSE

Communicating Errors

- Function return value
- Exceptions
- Pass by reference parameters
- Error callbacks
- `trigger_error`

Return Values

- FALSE on error
 - `return FALSE;`
- Error codes
 - `return MY_ERROR_CODE;`
- Error objects
 - `return new PEAR_Error();`

Return Values

- Overloading return values can be confusing
 - `strpos` `FALSE` versus `0`
- Hard to propagate
 - One omission breaks the chain
 - Hard to distinguish intentional and accidental omission
 - Extra propagation code inline

Return Values

```
function store($filename, $data) {
    $dir = dirname($filename);
    if ((!@is_dir($dir)) {
        if (!mkdir($dir, 0777, TRUE) {
            return FALSE;
        }
    }
    $tmpfile = tempnam($dir, '.tmp');
    if ($tmpfile) {
        if (!file_put_contents($tmpfile, $data)) {
            return FALSE;
        }
        return rename($tmpfile, $filename);
    }
    return FALSE;
}
```

Exceptions

- The class is the error code
- Great ability to convey information

Throwing an Exception

- We only need to define code at the point where the error occurs.

```
function show_feed_sidebar() {  
    throw new Exception("oops.");  
}
```

Catching An Exception

```
ob_start();  
try {  
    show_feed_sidebar();  
    ob_end_flush();  
} catch (Exception $e) {  
    ob_end_clean();  
    echo "Unavailable";  
}
```

Defining An Exception

```
class MyBad extends Exception {
    protected $extraInfo;

    function __construct($message,
                        $extraInfo) {
        $this->extraInfo = $extraInfo;
        parent::__construct($message);
    }
    function getExtraInfo() {
        return $this->extraInfo;
    }
}
```

Throwing a Custom Exception

```
function show_feed_sidebar() {  
    throw new MyBad("oops", $data);  
}
```

Catching A Specific Exception

```
ob_start();  
try {  
    show_feed_sidebar();  
    ob_end_flush();  
} catch (MyBad $e) {  
    ob_end_clean();  
    error_log($e);  
    echo "Unavailable";  
}
```

Uncaught Exceptions

- Get turned into fatal errors
- Can install a handler with `set_exception_handler`

Control Flow

- No extra error propagation code required
- Minimizes the distance between the event and the reaction
- Allows the reaction to take place in the appropriate context

Exceptions Reveal the Intent of the Code

```
// Hypothetical --  
// What if php functions threw exceptions?  
  
function store($filename, $data) {  
    $dir = dirname($filename);  
  
    if (!is_dir($dir)) {  
        mkdir($dir, 0777, TRUE);  
    }  
    $tmpfile = tempnam($dir, '.tmp');  
    file_put_contents($tmpfile, $data);  
    rename($tmpfile, $filename);  
}
```

Obscured Intent?

```
function store($filename, $data) {
    $dir = dirname($filename);
    if ((!@is_dir($dir)) {
        if (!mkdir($dir, 0777, TRUE) {
            return FALSE;
        }
    }
    $tmpfile = tempnam($dir, '.tmp');
    if ($tmpfile) {
        if (!file_put_contents($tmpfile, $data)) {
            return FALSE;
        }
        return rename($tmpfile, $filename);
    }
    return FALSE;
}
```

Converting Errors to Exceptions

- You can convert PHP errors to exceptions in an error handler
- You shouldn't do this
 - Most handleable error types are non-fatal
 - Other people's code won't know what to do
 - Not interoperable

Inappropriate Level of Abstraction

- File system driven cache:
 - `throw new DiskFullException();`
- Database driven cache:
 - `throw new DbConnectionException();`
- Should a Feed class that uses this cache API have to know about these exceptions?

Building Firewalls

- Convert errors from low level errors to high level errors
- Add information at each stage
- Build firewalls at system boundaries

Wrapping with PEAR_Exception

```
class CacheWriteException extends  
    PEAR_Exception {}  
  
try {  
    // Write to the cache with plugin  
} catch (Exception $e) {  
    throw new CacheWriteException(  
        "Item Not Cached", $e);  
}
```

Control Flow

- Don't use exceptions for "normal" control flow
- Consider a find() method:
 - Throw on Not Found?
 - Return FALSE on Not Found?

Not Everything Has to Be An Exception

```
if ($error_condition) {  
    trigger_error("message", E_USER_NOTICE);  
}
```

Convert Exceptions to Warnings

```
try {  
    // Do something  
} catch (Exception $e) {  
    trigger_error($e->getMessage(), E_USER_NOTICE);  
}
```

User Error Levels

- E_USER_NOTICE is ok
- E_USER_WARNING is ok
- Avoid E_USER_ERROR
 - Use an exception instead

Anti-Pattern: Exception Eater

```
try {  
    doSomething();  
} catch (Exception $e) {  
    // Do nothing  
}
```

Anti-Pattern: Unnecessary Catch

- ```
try {
 doSomething();
} catch (Exception $e) {
 throw $e;
}
```
- ```
try {  
    doSomething();  
} catch (Exception $e) {  
    die($e);  
}
```

Anti-Pattern: Generic Catch

- ```
try {
 doSomething();
} catch (Exception $e) {
 if ($e instanceof OtherException) {
 // Do something
 }
}
```
- ```
try {
    doSomething();
} catch (OtherException $e) {
    // Do something
}
```

Anti-Pattern: Catch Test

```
try {  
    // do stuff  
} catch (Exception $e) {  
    if (strpos($e->getMessage(), 'message text') {  
        // Do something  
    }  
    throw $e;  
}
```

Two Styles

```
throw new SomeKindOfException();
```

```
throw new SomeOtherKindOfException();
```

```
throw new Exception("Some kind of error");
```

```
throw new Exception("Some other kind of error");
```

Anti-Pattern: Exception Factory

```
function factory($message) {  
    return new Exception($message);  
}  
  
throw factory();
```

When To Throw

- `throw`
 - Fatal or complex errors
- `trigger_error`
 - Non-fatal warnings and notices

Catch When...

- ... you can do something about the exception
- ... leaving a system boundary
- ... you want to add more information
- ...The exception doesn't really indicate an error
- ...The exception should not be fatal

Questions?

- <http://www.procata.com/blog/>
- <http://www.mashery.com/>

