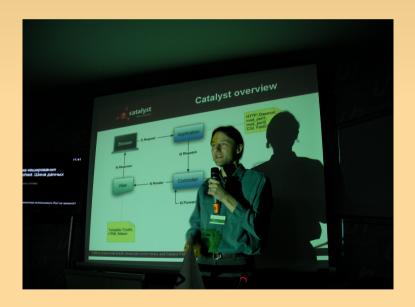
Exceptions

- Lightning Talk on Exceptions
- MiltonKeynes.pm 12-Jan-2009
- Peter Edwards peter@dragonstaff.co.uk



Slides at http://perl.dragonstaff.co.uk

What Are Exceptions

- Defensive programming
 - expected behaviour
 - you should anticipate bugs or incorrect input
 - recovery from or reporting of "exceptions" to expected behaviour

Exception Handling

- detect
 - check data and die/throw exception when invalid
 - wrap calls in eval:
 eval { \$foo->bah() };
 die "foo failed: \$@" if \$@;
- classify
 - user error program error
 - severity level (debug, info, warn, error, fatal)

Exception Handling

- report
 - warn to STDERR and continue
 - die with message to STDOUT
 - print to screen
 - logging with log4perl severity level

Exception Methods

- Return codes
 - don't do this!
 - is it 0 or 1 or a special value that is failure?
 - system calls vary
 - easy to forget to check return code in caller leading to hidden error
 - difficult to return code and message
 - sub a { return (1, "Invalid file format, file path: \$fpath"); }
 - my \$rc = a(); # oops, only gets message not code

Exception Methods

hierarchy of die/rethrow inside eval

```
sub a {
    die "oops"; # should be a croak() for context
    return 1;
 sub b {
    my $rc;
    eval { $rc = a() }; # undef
    die "a() failed: $@" if $@; # rethrow
    return $rc:
 eval { print "b returns " . b() };
 print "b() failed: $@" if $@;
 >>> b() failed: a() failed: oops at t.pl line 2.
```

Better Exceptions

- eval/die/rethrow is a bit clunky
- built-in error object methods would be nice
- languages like java or javascript give you try... catch... throw
- so use a CPAN exception module... which one?

Modules available

- Fatal makes failed builtins throw exceptions
 - use Fatal qw(open close);
- Error
 - gives you try... catch... throw
 - obscure nested closure memory leak

Modules available

- Class::Throwable
 - lightweight
 - easy to use
 - controllable levels of verbosity
 - extensible exception objects
 - I use this one

Modules Available

- Exception::Class
 - full-powered
 - complex
 - may be a performance overhead

Modules Available

Others include: Exception

 Good discussion of pros/cons in Class::Throwable peridoc

Gotcha

DIE blocks can reset \$@

```
sub check {
   croak "invalid parameter" unless $ [0];
sub DESTROY {
   local $@; # forget me and you've had it
   eval { somefoo() };
   print "in DESTROY\n";
eval { check(undef) };
print "$@\n";
>>> (empty)
```

CPAN solution: Devel::EvalError

Perl Best Practices

- "Perl Best Practices" pp.273-296 first ed. 2005,
 D.Conway pub. O'Reilly
 - Ten essential development practices
 #8 Throw exceptions instead of returning special values or setting flags
- See also Summary of Chapter 13, Error Handling (too many points to list here!)
- -end-