Perl Design Patterns

brian d foy
Stonehenge Consulting Services
February 24, 2006
Grand Rapids Perl Mongers

Sponsored by PriorityHealth

Patterns aren't a ...

* code cookbook
* module on CPAN
* methodology
* goal
* religion

Patterns are a ...

* abstraction for things we need to do
* solutions for common problems
* name for a design element
* common language for design discussions
* way to apply a common design to code
* relationships

Three parts

* Context
* System of forces
* Solution

Nothing's Free

* Actions have reactions
* Complexity turns up somewhere else
* Choices have consequences

Who's responsible?

- Our application needs config information
- But we have several modules
- Who loads the data?
- How do the other modules get it?

A pattern

* Load the configuration from anywhere
* But load it only one time
* Anyone else gets a reference to it
* The order doesn't matter

What’s in a name

* A couple words instead of sentences
* We agree on what the name implies
* Others know what we mean
* “A rose is a rose is a rose”

The Singleton

* There is only one configuration
* Let's call it a singleton
* ... or a highlander
* We don't have an implementation
* Just a name with implied design elements

An implementation

```
package My::Config;

my $singleton = undef;

sub new {
   my $class = shift;

   return $singleton if defined $singleton;

   $singleton = bless {}, $class;
   }
```

A use

```
package My::Database;
use My::Config;

my $config = My::Config->new( ... );

package My::Network;
use My::Config;

my $config = My::Config->new( ... );
```

That isn't the only way

* The pattern is not a prescription
* It's an option
* Maybe another pattern works better.

A Meta Class

* Write a meta class that contains all of the object parts
* Objects talk to the meta class to communicate with the other parts

```
use My::Controller;

my $controller = My::Controller->new(...);

my $value = $controller->config->get( ... );
```

Delegates

```
package My::Controller;

sub new {
    my $class = shift;

    my $self = bless {}, $class;

    # weaken some of these circular refs
    @$self{ qw(_config _database _network ) } = (
        My::Config->new( controller => $self ),
        DBI->new( ... ),
        My::Socket->new( controller => $self ),
        );

    $self;
    }

sub config { $_[0]{_config} }
```

Some Perl Modules

- Apache::DBI
- Netscape::Bookmarks
- CGI::Prototype
- many things in Class::*, almost

Beware

- Patterns are not code
- No matter what the Gang of Four say
- Class::* is code
- Ergo, ...

Perl is Better

* People like Patterns because they get code
* C++, Java suck at some things (Iterators)
* Perl doesn't suck at the same things

Further Reading

* The Perl Review (lots of articles (by me))
* "Design Patterns Aren't" by Mark Jason Dominus
    * http://perl.plover.com/yak/design/
* Design Patterns - Erich Gamma, et al. (Gang of Four)
* Perl Design Patterns Wiki
    * http://perldesignpatterns.com/