MOOSE AND ME

Introduction to Moose

ACKNOWLEDGEMENTS

- Linux Journal Jan, 2012
  - Moose - Henry Van Styn

Welcome to the Perl Renaissance

- Perl 5.20
- Moose
- DBIx::Class
- Catalyst
- Perl 6?

PERL OBJECTS

- Perl 5
  - Blessed References
  - Implementation is left to the user
- Moose
  - Blessed references
  - Types and constraints

WHAT ABOUT CPAN?

- Compatible
- Modules are being converted
- Better for Perl in general

SO LET’S GET STARTED

CLASS EXAMPLE

INTERFACE USAGE

```
use MyApp::Rifle;
use strict;

my $rifle = MyApp::Rifle->new( rounds => 5 );
print "There are " . $rifle->rounds . " rounds in the rifle\n";
$rifle->fire;
print "Now there are " . $rifle->rounds . " rounds in the rifle\n";
```

THE OLD WAY

Blessed Hashes

```
package MyApp::Rifle;
use strict;
sub new {
    my ($class, %opts) = @_;
    $opts{rounds} = 0 unless ($opts{rounds});
    my $self = bless( {}, $class );
    $self->rounds($opts{rounds}); return $self;
}

sub rounds {
    my ($self, $rounds) = @_;
    $self->{_rounds} = $rounds if (defined $rounds);
    return $self->{_rounds};
}

sub fire {
    my $self = shift; die "out of ammo!" unless ($self->rounds > 0);
    print "bang!\n"; $self->rounds( $self->rounds - 1 );
}

1;
```

THE NEW WAY

Moose is still blessed hashes

```
package MyApp::Rifle;
use Moose;

has 'rounds' => ( is => 'rw', isa => 'Int', default => 0 );

sub fire {
        my $self = shift;
        die "out of ammo!" unless ($self->rounds > 0);
        print "bang!\n";
        $self->rounds( $self->rounds - 1 );
}

1;
```

USING MOOSE

Moose

Attributes

- Example
  - has 'first_name' => (is => 'rw');
- has
  - is => [ro/rw]
  - isa => Int, String,etc.
  - default => <value>
  - builder => <method>
  - lazy => [0,1]
  - required => [0,1]
  - lazy_build => [0,1]

MOOSE

Attribute Lazy Builder

- Lazy Build
  - Named build_<attribute>

```
has 'first_name' => ( is => 'ro', lazy_build => 1 );
sub _build_first_name {
        my $self = shift;
        return $self->some_lookup('some data');
}
```

Moose

Objects as Attributes

```
package MyApp::Rifle;
use Moose;
use DateTime;

has 'rounds' => ( is => 'rw', isa => 'Int', default => 0 );
has 'fired_dt' => ( is => 'rw', isa => 'DateTime' );

sub fire {
        my $self = shift;
        die "out of ammo!" unless ($self->rounds > 0);

        my $dt = DateTime->now( time_zone => 'local' );
        $self->fired_dt($dt);

        print "bang!\n";
        print "fired at " . $self->fired_dt->datetime . "\n";

        $self->rounds( $self->rounds - 1 );
}

1;
```

MOOSE

Delegation

```
has 'fired_dt' => (
        is => 'rw',
        isa => 'DateTime',
        handles => {
                last_fired => 'datetime'
        }
);

-----

$self->last_fired

vs.

$self->fired_dt->datetime
```

MOOSE

Built-in Types

```
Any
    Item
        Bool
        Maybe[`a]
        Undef
        Defined
            Value
                Str
                    Num
                        Int
                    ClassName
                    RoleName
            Ref
                ScalarRef[`a]
                ArrayRef[`a]
                HashRef[`a]
                CodeRef
                RegexpRef
                GlobRef
                FileHandle
                Object
```

Moose

Other Type Talk

```
Bool | Ref

Maybe[Num]

ArrayRef[Int]

ArrayRef[HashRef[Str]]

Also See:
    Moose::Util::TypeConstraints
```

MOOSE

INHERITANCE

```
package MyApp::AutomaticRifle;
use Moose;
extends 'MyApp::Rifle';

has '+rounds' => ( default => 50 );
has 'last_burst_num' => ( is => 'rw', isa => 'Int'
);

sub burst_fire {
        my ($self, $num) = @_;

        $self->last_burst_num($num);

        for (my $i=0; $i<$num; $i++) {
                $self->fire;
        }
}

1;
```

MOOSE

INHERITANCE USAGE

```
use strict;
use MyApp::AutomaticRifle;

my $rifle = MyApp::AutomaticRifle->new;
print "There are " . $rifle->rounds . " rounds in the rifle\n";
$rifle->burst_fire(35);
print "Now there are " . $rifle->rounds . " rounds in the rifle\n";
```

MOOSE

ROLE

```
package MyApp::FireAll;
use strict;
use Moose::Role;

requires 'fire', 'rounds';

sub fire_all {
        my $self = shift;
        $self->fire while($self->rounds > 0);
}

1;

---
with 'MyApp::FireAll';
```

MOOSE

Method Modifiers

```
before

after

around

---

before 'fire_all' => sub {
        my $self = shift;
        print "Say hello to my little friend!\n";
};

around 'fire_all' => sub {
       my ($orig, $self, @args) = @_;
       return $self->$orig(@args);
};
```

MOOSE

Method Modifiers Example Role

```
package MyApp::MightJam;
use Moose::Role;
use Moose::Util::TypeConstraints;

requires 'fire';

subtype 'Probability' => (
        as 'Num',
        where { $_ >= 0 && $_ <= 1 },
        message { "$_ is not a number between 0 and 1" }
);

has 'jam_probability' => (
        is => 'ro',
        isa => 'Probability',
        default => .01
);

sub roll_dice {
        my $self = shift;
        return 1 if ( rand(1) < $self->jam_probability );
        return 0;
}

before 'fire' => sub {
        my $self = shift;
        die "Jammed!!!\n" if ($self->roll_dice);
};

1;
```

Moose

Method Modifiers Example Role

```
package MyApp::CrappyRifle;
use strict;
use Moose;
extends 'MyApp::AutomaticRifle';
with 'MyApp::MightJam';

has '+jam_probability' => ( default => .5 );

1;

---

package MyApp::NiceRifle;
use strict;
use Moose;
extends 'MyApp::AutomaticRifle';
with 'MyApp::MightJam';

has '+jam_probability' => ( default => .001 );

1;
```

MORE?

- Moose CPAN Page: http://search.cpan.org/perldoc?Moose
- Moose Manual: http://search.cpan.org/perldoc?Moose::Manual
- Moose::Util::TypeConstraints Documentation: http://search.cpan.org/perldoc?Moose::Util::TypeConstraints
- Moose IRC Channel: #moose on irc.perl.org
- perlreftut—Perl Reference Tutorial: http://perldoc.perl.org/perlreftut.html