

Manager's Guide To Moose

Plus An Introduction

(Heavily derived from Stevan Little's talk of the same
name.)

Cory G Watson: 'gphat'

- Director of Development: Magazines.com
- Catalyst, DBIx::Class, Graphics::Primitive
- <http://www.onemogin.com/blog/>
- JAPH

What Is Moose?

What Moose Is Not

- An experiment or prototype
- A toy
- Yet Another Accessor Builder
- A source filter
- Perl 6 in Perl 5

What Moose Is

- An extension of Perl 5.00
- Stable, production ready (over 2 years)
- Lessons learned from lots of other languages
- A step toward Perl 6

Class::MOP

- CLOS for Perl 5
- Formalization of Perl 5 OO
- 20 year old Lisp technology
- Moose is syntactic sugar

Simple Example

```
package Shape;  
use Moose;
```

Simple Example (cont)

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

What Did That Do?

- Created a class
- used 'strict' and 'warnings'
- Added an 'x' attribute
- Made 'x' read-write
- Requires a type of 'Int'
- Defaults 'x' to 0

How Is It Used?

```
my $shape = Shape->new(  
  x => 5  
);  
say $shape->x;  
$shape->x(6);
```

Moar Oh-Oh

```
package Rectangle;  
use Moose;
```

```
extends 'Shape';
```

```
has 'height' => ...
```

Extends?

- Subclasses the given class
- Lets you add new attributes
- allows you to modify existing attributes...

What? Modify?

```
has '+x' (  
  default => 5  
);
```

Neat!

- Indeed!
- 'x' now defaults to 5.
- Check the docs. You can do lots of neat stuff like that.

Type-ography

- Moose has lots of types
- Types are subclassable
- Any class works (`isa => 'MyClass'`)
- `ArrayRef[MyClass]` DWIMs
- `Maybe[MyClass]` handles possible undef

Meta-Shmeta

- All classes have a `->meta`
- `Meta` is a metaclass
- `Shape->meta->name # Shape`
- `Shape->meta->attribute_list`
`# [x]`
- `Rectangle->meta->superclasses`
`# [Shape]`

Phenomenal Meta Power

- `meta->add_method(...);`
- `meta->remove_method(...);`
- `meta->add_attribute(...);`
- `meta->remove_attribute(...);`

But Why?

- Perl is Old and Ugly
- OO involves lots of yak shaving
- CPAN is full of junk
- Line-noise

TMTOWTDI (BCINABT)

- There's More Than One Way To Do It
- But Consistency Is Not A Bad Thing

That's A Problem?

- Hard to hire Perl programmers
- Many are scripters
- Everyone does it differently

OOP Madness

Class::Accessor Class::MakeMethods base.pm Spiffy
Class::HPLOO Class::Base Object::Tiny Object::Lexical
EO Class::Accessor::Fast Class::Closure Class::Meta
Class::Simple Class::Gomor Rose::Object Class::Builder
Class::InsideOut Object::LocalVars Oak::Object OOP
Object::InsideOut Class::Dot Class::NamedParms Myco
Class::Structured Class::Classless parent.pm Eobj
Class::Prototyped Class::Init Class::Maker Class::Object
Fukurama::Class Class::Declare Class::Std Object::Declare
Class::Struct Class::AutoClass Class::Root Badger Oryx
Object::Prototype Basset Object::Accessor Class::Lego
Class::Container Tangram OO::Closures Class::Trait MOP
Object::MultiType SLOOPS Class::TOM Class::PObject
Moose
... and many more

Why Moose?

- Consistency eq Maintainability
- Moose is Extensible (MooseX::*)
- Incremental
- Bridge to Perl 6
- Less Testing

More Moose: attributes

```
has 'required_field' => (  
    required => 1  
);
```

```
has 'big_object' => (  
    lazy => 1,  
    default => sub {  
        return BigObject->new  
    }  
);
```

More Moose: coercions

```
class_type 'DateTime::Duration';  
coerce 'DateTime::Duration'  
  => from 'Int'  
  => via {  
    DateTime::Duration->new(  
      years => $_  
    )  
  };
```

More Moose: delegation

```
has age => (  
    is          => 'rw',  
    isa        => 'DateTime::Duration',  
    default    => sub {  
        DateTime::Duration->new  
    },  
    handles    => {  
        'years_old' => 'years'  
    }  
);
```

More Moose: modifiers

- after
- before
- around

More Moose: Roles

```
package Foo;  
use Moose::Role;  
  
requires 'something';  
  
has 'age' => (  
...  
);
```