## NAME

Mouse::Meta::Attribute − The Mouse attribute metaclass

## VERSION

This document describes Mouse version 0.97

## DESCRIPTION

This is a meta object protocol for Mouse attributes, which is a subset of Moose::Meta::Attribute.

## METHODS

`new(%options) −> Mouse::Meta::Attribute`

Instantiates a new Mouse::Meta::Attribute. Does nothing else.

It adds the following options to the constructor:

`is => 'ro', 'rw', 'bare'`

This provides a shorthand for specifying the `reader`, `writer`, or `accessor` names. If the attribute is read-only ('ro') then it will have a `reader` method with the same attribute as the name.

If it is read-write ('rw') then it will have an `accessor` method with the same name. If you provide an explicit `writer` for a read-write attribute, then you will have a `reader` with the same name as the attribute, and a `writer` with the name you provided.

Use 'bare' when you are deliberately not installing any methods (accessor, reader, etc.) associated with this attribute; otherwise, Moose will issue a deprecation warning when this attribute is added to a metaclass.

`isa => Type`

This option accepts a type. The type can be a string, which should be a type name. If the type name is unknown, it is assumed to be a class name.

This option can also accept a Moose::Meta::TypeConstraint object.

If you *also* provide a `does` option, then your `isa` option must be a class name, and that class must do the role specified with `does`.

`does => Role`

This is short-hand for saying that the attribute's type must be an object which does the named role.

**This option is not yet supported.**

`coerce => Bool`

This option is only valid for objects with a type constraint (`isa`). If this is true, then coercions will be applied whenever this attribute is set.

You can make both this and the `weak_ref` option true.

`trigger => CodeRef`

This option accepts a subroutine reference, which will be called after the attribute is set.

`required => Bool`

An attribute which is required must be provided to the constructor. An attribute which is required can also have a `default` or `builder`, which will satisfy its required-ness.

A required attribute must have a `default`, `builder` or a non−`undef` `init_arg`

`lazy => Bool`

A lazy attribute must have a `default` or `builder`. When an attribute is lazy, the default value will not be calculated until the attribute is read.

`weak_ref => Bool`

If this is true, the attribute's value will be stored as a weak reference.

`auto_deref => Bool`

If this is true, then the reader will dereference the value when it is called. The attribute must have a type constraint which defines the attribute as an array or hash reference.

`lazy_build => Bool`

Setting this to true makes the attribute lazy and provides a number of default methods.

```
has 'size' => (
    is         => 'ro',
    lazy_build => 1,
);
```

is equivalent to this:

```
has 'size' => (
    is        => 'ro',
    lazy      => 1,
    builder   => '_build_size',
    clearer   => 'clear_size',
    predicate => 'has_size',
);
```

`associate_method(MethodName)`
: Associates a method with the attribute. Typically, this is called internally when an attribute generates its accessors.

  Currently the argument *MethodName* is ignored in Mouse.

`verify_against_type_constraint(Item) -> TRUE | ERROR`
: Checks that the given value passes this attribute's type constraint. Returns `true` on success, otherwise `confess`es.

`clone_and_inherit_options(options) -> Mouse::Meta::Attribute`
: Creates a new attribute in the owner class, inheriting options from parent classes. Accessors and helper methods are installed. Some error checking is done.

`get_read_method_ref`
`get_write_method_ref`
: Returns the subroutine reference of a method suitable for reading or writing the attribute's value in the associated class. These methods always return a subroutine reference, regardless of whether or not the attribute is read- or write-only.

## SEE ALSO

Moose::Meta::Attribute

Class::MOP::Attribute