NAME

MooX::Types::MooseLike::Base − A set of basic Moose−like types for Moo

SYNOPSIS

```
package MyPackage;
use Moo;
use MooX::Types::MooseLike::Base qw(:all);

has "beers_by_day_of_week" => (
    isa => HashRef
);

has "current_BAC" => (
    isa => Num
);

# Also supporting is_$type.  For example, is_Int() can be used as follows
has 'legal_age' => (
    is => 'ro',
    isa => sub { die "$_[0] is not of legal age"
                 unless (is_Int($_[0]) && $_[0] > 17) },
);
```

DESCRIPTION

Moo attributes (like Moose) have an 'isa' property. This module provides some basic types for this property. One can import all types with ':all' tag or import a list of types like:

```
use MooX::Types::MooseLike::Base qw/HashRef ArrayRef/;
```

so one could then declare some attributes like:

```
has 'contact' => (
  is => 'ro',
  isa => HashRef,
);
has 'guest_list' => (
  is => 'ro',
  isa => ArrayRef[HashRef],
);
```

These types provide a check that the *contact* attribute is a `hash` reference, and that the *guest_list* is an `array of hash` references.

TYPES (1st class functions − return a coderef)

Any

Any type (test is always true)

Item

Synonymous with Any type

Undef

A type that is not defined

Defined

A type that is defined

Bool

A boolean 1|0 type

Value

A non-reference type

Ref

A reference type

Str
A non-reference type where a reference to it is a SCALAR

Num
A number type

Int
An integer type

ArrayRef
An ArrayRef (ARRAY) type

HashRef
A HashRef (HASH) type

CodeRef
A CodeRef (CODE) type

RegexpRef
A regular expression reference type

GlobRef
A glob reference type

FileHandle
A type that is either a builtin perl filehandle or an IO::Handle object

Object
A type that is an object (think blessed)

PARAMETERIZED TYPES

Parameterizing Types With a Single Type

The following types can be parameterized with other types.

ArrayRef

For example, ArrayRef[HashRef]

HashRef

ScalarRef

Maybe

For example, Maybe[Int] would be an integer or undef

Parameterizing Types With Multiple Types

AnyOf

Check if the attribute is any of the listed types (think union). Takes a list of types as the argument, for example:

```
isa => AnyOf[Int, ArrayRef[Int], HashRef[Int]]
```

Note: AnyOf is passed an ArrayRef[CodeRef]

AllOf

Check if the attribute is all of the listed types (think intersection). Takes a list of types as the argument. For example:

```
isa => AllOf[
  InstanceOf['Human'],
  ConsumerOf['Air'],
  HasMethods['breath', 'dance']
],
```

Parameterizing Types With (Multiple) Strings

In addition, we have some parameterized types that take string arguments.

InstanceOf

Check if the attribute is an object instance of one or more classes. Uses `blessed` and `isa` to do so. Takes a list of class names as the argument. For example:

```
isa => InstanceOf['MyClass','MyOtherClass']
```

Note: InstanceOf is passed an ArrayRef[Str]

ConsumerOf

Check if the attribute is blessed and consumes one or more roles. Uses `blessed` and `does` to do so. Takes a list of role names as the arguments. For example:

```
isa => ConsumerOf['My::Role', 'My::AnotherRole']
```

HasMethods

Check if the attribute is blessed and has one or more methods. Uses `blessed` and `can` to do so. Takes a list of method names as the arguments. For example:

```
isa => HasMethods[qw/postulate contemplate liberate/]
```

Enum

Check if the attribute is one of the enumerated strings. Takes a list of possible string values. For example:

```
isa => Enum['rock', 'spock', 'paper', 'lizard', 'scissors']
```

SEE ALSO

MooX::Types::MooseLike::Numeric − an example of building subtypes.

MooX::Types::SetObject − an example of building parameterized types.

MooX::Types::MooseLike::Email, MooX::Types::MooseLike::DateTime

AUTHOR

Mateu Hunter `hunter AT missoula DOT org`

THANKS

mst has provided critical guidance on the design

COPYRIGHT

LICENSE