NAME

Type::Tiny::Manual::UsingWithMoo – how to use Type::Tiny and Type::Library with Moo

SYNOPSIS

```
{
   package Person;

   use Moo 1.006000;
   use Sub::Quote qw( quote_sub );
   use Types::Standard qw( Str Int );

   has name => (
      is      => "ro",
      isa     => Str,
   );

   my $PositiveInt = Int
      -> where( quote_sub '$_ > 0' )
      -> plus_coercions( Int, sub { abs $_ } );

   has age => (
      is      => "rwp",
      isa     => $PositiveInt,
      coerce  => 1,
   );

   sub get_older {
      my $self = shift;
      my ($years) = @_;
      $PositiveInt->assert_valid($years);
      $self->_set_age($self->age + $years);
   }
}
```

DESCRIPTION

Type::Tiny is tested with Moo 1.001000 and above.

Type::Tiny overloads `&{}`. Moo supports using objects that overload `&{}` as `isa` constraints, so Type::Tiny objects can directly be used in `isa`.

Moo prior to 1.006000 doesn't support `coerce => 1`, instead requiring a coderef to use as a coercion. However, Type::Tiny can provide you with a suitable coderef to use (actually an object that overloads `&{}`). Just use:

```
has age => (
   is      => "rwp",
   isa     => $PositiveInt,
   coerce  => $PositiveInt->coercion,
);
```

If you can upgrade to the latest Moo, and use `coerce => 1` you'll have a lot more fun though. :–)

Type::Tiny hooks into Moo's HandleMoose interface to ensure that type constraints get inflated to Moose type constraints if and when Moo inflates your class to a full Moose class.

Optimization

The usual advice for optimizing type constraints applies: use type constraints which can be inlined whenever possible, and define coercions as strings rather than coderefs.

Upgrading to Moo 1.002000 or above should provide a slight increase in speed for type constraints, as it

allows them to be inlined into accessors and constructors.

If creating your own type constraints using `Type::Tiny->new`, then consider using Sub::Quote to quote the coderef; this allows you to take advantage of inlining without having to write your own inlining routines.

See also Type::Tiny::Manual::Optimization.

SEE ALSO

For examples using Type::Tiny with Moo see the SYNOPSIS sections of Type::Tiny and Type::Library, and the Moo integration tests <https://github.com/tobyink/p5-type-tiny/tree/master/t/30-integration/Moo> in the test suite.

AUTHOR

Toby Inkster <tobyink@cpan.org>.

COPYRIGHT AND LICENCE

DISCLAIMER OF WARRANTIES