NAME

Type::Tiny::Manual::UsingWithOther – how to use Type::Tiny and Type::Library with other OO frameworks

DESCRIPTION

Class::InsideOut

You want Class::InsideOut 1.13 or above, which has support for blessed and overloaded objects (including Type::Tiny type constraints) for the `get_hook` and `set_hook` options.

```
{
   package Person;

   use Class::InsideOut qw( public );
   use Types::Standard qw( Str Int );
   use Type::Utils qw( declare as where inline_as coerce from );

   public name => my %_name, {
      set_hook => Str,
   };

   my $PositiveInt = declare
      as        Int,
      where     {  $_ > 0  },
      inline_as { "$_ =~ /^[0-9]+\$/ and $_ > 0" };

   coerce $PositiveInt, from Int, q{ abs $_ };

   public age => my %_age, {
      set_hook => sub { $_ = $PositiveInt->assert_coerce($_) },
   };

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

I probably need to make coercions a little prettier.

See also: `t/25_accessor_hooks_typetiny.t` and `t/Object/HookedTT.pm` in the Class::InsideOut test suite; and the Class-InsideOut integration tests <https://github.com/tobyink/p5-type-tiny/tree/master/t/30-integration/Class-InsideOut> in the Type::Tiny test suite.

Params::Check and Object::Accessor

The Params::Check `allow()` function, the `allow` option for the Params::Check `check()` function, and the input validation mechanism for Object::Accessor all work in the same way, which is basically a limited pure-Perl implementation of the smart match operator. While this doesn't directly support Type::Tiny constraints, it does support coderefs. You can use Type::Tiny's `compiled_check` method to obtain a suitable coderef.

Param::Check example:

```
my $tmpl = {
   name => { allow => Str->compiled_check },
   age  => { allow => Int->compiled_check },
};
check($tmpl, { name => "Bob", age => 32 })
   or die Params::Check::last_error();
```

Object::Accessor example:

```
my $obj = Object::Accessor->new;
$obj->mk_accessors(
   { name => Str->compiled_check },
   { age  => Int->compiled_check },
);
```

Caveat: Object::Accessor doesn't die when a value fails to meet its type constraint; instead it outputs a warning to STDERR. This behaviour can be changed by setting `$Object::Accessor::FATAL = 1`.

See also: The Object-Accessor integration tests <https://github.com/tobyink/p5-type-tiny/tree/master/t/30-integration/Object-Accessor> in the Type::Tiny test suite.

Validation::Class::Simple

You want Validation::Class::Simple 7.900017 or above.

The `to_TypeTiny` function from Types::TypeTiny can be used to create a Type::Tiny type constraint from a Validation::Class::Simple object (and probably from Validation::Class, but this is untested).

```
use Types::TypeTiny qw( to_TypeTiny );
use Validation::Class::Simple;

my $type = to_TypeTiny Validation::Class::Simple->new(
   fields => {
      name => {
         required => 1,
         pattern  => qr{^\w+(\s\w+)*$},
         filters  => ["trim", "strip"],
      },
      email => { required => 1, email => 1 },
      pass  => { required => 1, min_length => 6 },
   },
);

# true
$type->check({
   name   => "Toby Inkster",
   email  => "tobyink@cpan.org",
   pass   => "foobar",
});

# false
$type->check({
   name   => "Toby Inkster ",    # trailing whitespace
   email  => "tobyink@cpan.org",
   pass   => "foobar",
});

# coercion from HashRef uses the filters defined above
my $fixed = $type->coerce({
   name   => "Toby Inkster ",    # trailing whitespace
```

```
   email  => "tobyink@cpan.org",
   pass   => "foobar",
});

# true
$type->check($fixed);
```

Type constraints built with Validation::Class::Simple are not inlinable, so won't be as fast as `Dict` from Types::Standard, but the filters are a pretty useful feature. (Note that filters are explicitly *ignored* for type constraint checking, and only come into play for coercion.)

See also: The Validation-Class-Simple integration tests <https://github.com/tobyink/p5-type-tiny/tree/master/t/30-integration/Validation-Class-Simple> in the Type::Tiny test suite.

AUTHOR

Toby Inkster <tobyink@cpan.org>.

COPYRIGHT AND LICENCE

DISCLAIMER OF WARRANTIES