NAME

Type::Tiny::Manual – an overview of Type::Tiny

SYNOPSIS

Type::Tiny is a small class for writing type constraints, inspired by Moose's type constraint API. It has only one non-core dependency (and even that is simply a module that was previously distributed as part of Type::Tiny but has since been spun off), and can be used with Moose, Mouse and Moo (or none of the above).

Type::Tiny is bundled with Type::Library a framework for organizing type constraints into collections.

Also bundled is Types::Standard, a Moose-inspired library of useful type constraints.

Type::Params is also provided, to allow very fast checking and coercion of function and method parameters.

SEE ALSO

- Libraries – how to build a type library with Type::Tiny, Type::Library and Type::Utils
- Coercions – adding coercions to type constraints
- Using with Moose – how to use Type::Tiny and Type::Library with Moose
- Using with Mouse – how to use Type::Tiny and Type::Library with Mouse
- Using with Moo – how to use Type::Tiny and Type::Library with Moo
- Using with Other OO Frameworks – how to use Type::Tiny and Type::Library with other OO frameworks
- Type::Tiny and friends don't need to be used within an OO framework. See FreeMind::Node for an example that does not.
- Processing arguments to subs – coerce and validate arguments to functions and methods.
- Other modules using Type::Tiny in interesting ways: Type::Tie, Test::Mocha, Scalar::Does, Set::Equivalence...
- Optimization – squeeze the most out of your CPU.
- Type::Tiny maintenance policies – the stability policy.

DEPENDENCIES

Type::Tiny requires at least Perl 5.6.1, though certain Unicode-related features (e.g. non-ASCII type constraint names) may work better in newer versions of Perl.

Type::Tiny requires Exporter::Tiny, a module that was previously bundled in this distribution, but has since been spun off as a separate distribution. Don't worry – it's quick and easy to install.

At run-time, Type::Tiny also requires the following modules: B, B::Deparse, Carp, Data::Dumper, Scalar::Util, Text::Balanced, overload, strict and warnings. All of these come bundled with Perl itself. Prior to Perl 5.8, Scalar::Util and Text::Balanced do not come bundled with Perl and will need installing separately from the CPAN.

Certain features require additional modules. Tying a variable to a type constraint (e.g. `tie my $count, Int`) requires Type::Tie; stack traces on exceptions require Devel::StackTrace. The Reply::Plugin::TypeTiny plugin for Reply requires Reply (obviously). Devel::LexAlias may *slightly* increase the speed of some of Type::Tiny's compiled coderefs.

Type::Tiny::XS is not required, but if available provides a speed boost for some type checks. (Setting the environment variable `PERL_TYPE_TINY_XS` to false, or setting `PERL_ONLY` to true will suppress the use of Type::Tiny::XS, even if it is available.)

The test suite additionally requires Test::More, Test::Fatal and Test::Requires. Test::More comes bundled with Perl, but if you are using a version of Perl older than 5.14, you will need to upgrade to at least Test::More version 0.96. Test::Requires and Test::Fatal (plus Try::Tiny which Test::Fatal depends on) are bundled with Type::Tiny in the `inc` directory, so you do not need to install them separately.

If using Type::Tiny in conjunction with Moo, then at least Moo 1.000000 is recommended. If using

Type::Tiny with Moose, then at least Moose 2.0000 is recommended. If using Type::Tiny with Mouse, then at least Mouse 1.00 is recommended. Type::Tiny is mostly untested against older versions of these packages.

TYPE::TINY VERSUS X

Specio

Type::Tiny is similar in aim to Specio. The major differences are

- Type::Tiny is "tiny" (Specio will eventually have fewer dependencies than it currently does, but is unlikely to ever have as few as Type::Tiny);
- Specio has a somewhat nicer API (better method names; less duplication), and its API is likely to improve further. Type::Tiny's aims at complete compatibility with current versions of Moose and Mouse, so there is a limit to how much I can deviate from the existing APIs of (Moose|Mouse)::Meta::TypeConstraint.

MooseX::Types

Type::Tiny libraries expose a similar interface to MooseX::Types libraries. In most cases you should be able to rewrite a MooseX::Types library to use Type::Tiny pretty easily.

MooX::Types::MooseLike

Type::Tiny is faster and supports coercions.

Scalar::Does

Scalar::Does is somewhat of a precursor to Type::Tiny, but has now been rewritten to use Type::Tiny internally.

It gives you a `does($value, $type)` function that is roughly equivalent to `$type->check($value)` except that `$type` may be one of a list of pre-defined strings (instead of a Type::Tiny type constraint); or may be a package name in which case it will be assumed to be a role and checked with `$value->DOES($type)`.

BUGS

Please report any bugs to <http://rt.cpan.org/Dist/Display.html?Queue=Type-Tiny>.

SUPPORT

IRC: support is available through in the *#moops* channel on irc.perl.org <http://www.irc.perl.org/channels.html>.

AUTHOR

Toby Inkster <tobyink@cpan.org>.

COPYRIGHT AND LICENCE

DISCLAIMER OF WARRANTIES