NAME

Exporter::Tiny::Manual::Etc – odds and ends

DESCRIPTION

Utility Functions

Exporter::Tiny is itself an exporter!

These functions are really for internal use, but can be exported if you need them:

`mkopt(\@array)`
: Similar to `mkopt` from Data::OptList. It doesn't support all the fancy options that Data::OptList does (`moniker`, `require_unique`, `must_be` and `name_test`) but runs about 50% faster.

`mkopt_hash(\@array)`
: Similar to `mkopt_hash` from Data::OptList. See also `mkopt`.

History

Type::Library had a bunch of custom exporting code which poked coderefs into its caller's stash. It needed this to be something more powerful than most exporters so that it could switch between exporting Moose, Mouse and Moo-compatible objects on request. Sub::Exporter would have been capable, but had too many dependencies for the Type::Tiny project.

Meanwhile Type::Utils, Types::TypeTiny and Test::TypeTiny each used the venerable Exporter.pm. However, this meant they were unable to use the features like Sub::Exporter-style function renaming which I'd built into Type::Library:

```
## import "Str" but rename it to "String".
use Types::Standard "Str" => { -as => "String" };
```

And so I decided to factor out code that could be shared by all Type-Tiny's exporters into a single place: Exporter::TypeTiny.

As of version 0.026, Exporter::TypeTiny was also made available as Exporter::Tiny, distributed independently on CPAN. CHOCOLATEBOY had convinced me that it was mature enough to live a life of its own.

As of version 0.030, Type-Tiny depends on Exporter::Tiny and Exporter::TypeTiny is being phased out.

Obligatory Exporter Comparison

Exporting is unlikely to be your application's performance bottleneck, but nonetheless here are some comparisons.

Comparative sizes according to Devel::SizeMe:

```
Exporter                     217.1Kb
Sub::Exporter::Progressive   263.2Kb
Exporter::Tiny               267.7Kb
Exporter + Exporter::Heavy   281.5Kb
Exporter::Renaming           406.2Kb
Sub::Exporter                701.0Kb
```

Performance exporting a single sub:

```
               Rate     SubExp    ExpTiny SubExpProg      ExpPM
SubExp       2489/s         --       -56%       -85%       -88%
ExpTiny      5635/s       126%         --       -67%       -72%
SubExpProg  16905/s       579%       200%         --       -16%
ExpPM       20097/s       707%       257%        19%         --
```

(Exporter::Renaming globally changes the behaviour of Exporter.pm, so could not be included in the same benchmarks.)

(Non-Core) Dependencies:

```
Exporter                    -1
Exporter::Renaming           0
Exporter::Tiny               0
Sub::Exporter::Progressive   0
Sub::Exporter                3
```

Features:

	ExpPM	ExpTiny	SubExp	SubExpProg
Can export code symbols	Yes	Yes	Yes	Yes
Can export non-code symbols	Yes	Yes		
Groups/tags	Yes	Yes	Yes	Yes
Export by regexp	Yes	Yes		
Bang prefix	Yes	Yes		
Allows renaming of subs		Yes	Yes	Maybe
Install code into scalar refs		Yes	Yes	Maybe
Can be passed an "into" parameter		Yes	Yes	Maybe
Can be passed an "installer" sub		Yes	Yes	Maybe
Config avoids package variables			Yes	
Supports generators		Yes	Yes	
Sane API for generators		Yes	Yes	
Unimport		Yes		

(Certain Sub::Exporter::Progressive features are only available if Sub::Exporter is installed.)

SEE ALSO

Exporter::Shiny, Exporter::Tiny.

AUTHOR

Toby Inkster <tobyink@cpan.org>.

COPYRIGHT AND LICENCE

DISCLAIMER OF WARRANTIES