NAME

Sub::Install – install subroutines into packages easily

VERSION

version 0.928

SYNOPSIS

```
use Sub::Install;

Sub::Install::install_sub({
  code => sub { ... },
  into => $package,
  as   => $subname
});
```

DESCRIPTION

This module makes it easy to install subroutines into packages without the unsightly mess of `no strict` or typeglobs lying about where just anyone can see them.

FUNCTIONS

install_sub

```
Sub::Install::install_sub({
 code => \&subroutine,
 into => "Finance::Shady",
 as   => 'launder',
});
```

This routine installs a given code reference into a package as a normal subroutine. The above is equivalent to:

```
no strict 'refs';
*{"Finance::Shady" . '::' . "launder"} = \&subroutine;
```

If `into` is not given, the sub is installed into the calling package.

If `code` is not a code reference, it is looked for as an existing sub in the package named in the `from` parameter. If `from` is not given, it will look in the calling package.

If `as` is not given, and if `code` is a name, `as` will default to `code`. If `as` is not given, but if `code` is a code ref, Sub::Install will try to find the name of the given code ref and use that as `as`.

That means that this code:

```
Sub::Install::install_sub({
  code => 'twitch',
  from => 'Person::InPain',
  into => 'Person::Teenager',
  as   => 'dance',
});
```

is the same as:

```
package Person::Teenager;

Sub::Install::install_sub({
  code => Person::InPain->can('twitch'),
  as   => 'dance',
});
```

reinstall_sub

This routine behaves exactly like `"install_sub"`, but does not emit a warning if warnings are on and the destination is already defined.

install_installers

This routine is provided to allow Sub::Install compatibility with Sub::Installer. It installs `install_sub` and `reinstall_sub` methods into the package named by its argument.

```
Sub::Install::install_installers('Code::Builder'); # just for us, please
Code::Builder->install_sub({ name => $code_ref });

Sub::Install::install_installers('UNIVERSAL'); # feeling lucky, punk?
Anything::At::All->install_sub({ name => $code_ref });
```

The installed installers are similar, but not identical, to those provided by Sub::Installer. They accept a single hash as an argument. The key/value pairs are used as the `as` and `code` parameters to the `install_sub` routine detailed above. The package name on which the method is called is used as the `into` parameter.

Unlike Sub::Installer's `install_sub` will not eval strings into code, but will look for named code in the calling package.

EXPORTS

Sub::Install exports `install_sub` and `reinstall_sub` only if they are requested.

exporter

Sub::Install has a never-exported subroutine called `exporter`, which is used to implement its `import` routine. It takes a hashref of named arguments, only one of which is currently recognize: `exports`. This must be an arrayref of subroutines to offer for export.

This routine is mainly for Sub::Install's own consumption. Instead, consider Sub::Exporter.

SEE ALSO

Sub::Installer
: This module is (obviously) a reaction to Damian Conway's Sub::Installer, which does the same thing, but does it by getting its greasy fingers all over UNIVERSAL. I was really happy about the idea of making the installation of coderefs less ugly, but I couldn't bring myself to replace the ugliness of typeglobs and loosened strictures with the ugliness of UNIVERSAL methods.

Sub::Exporter
: This is a complete Exporter.pm replacement, built atop Sub::Install.

EXTRA CREDITS

Several of the tests are adapted from tests that shipped with Damian Conway's Sub-Installer distribution.

AUTHOR

Ricardo SIGNES <rjbs@cpan.org>

COPYRIGHT AND LICENSE