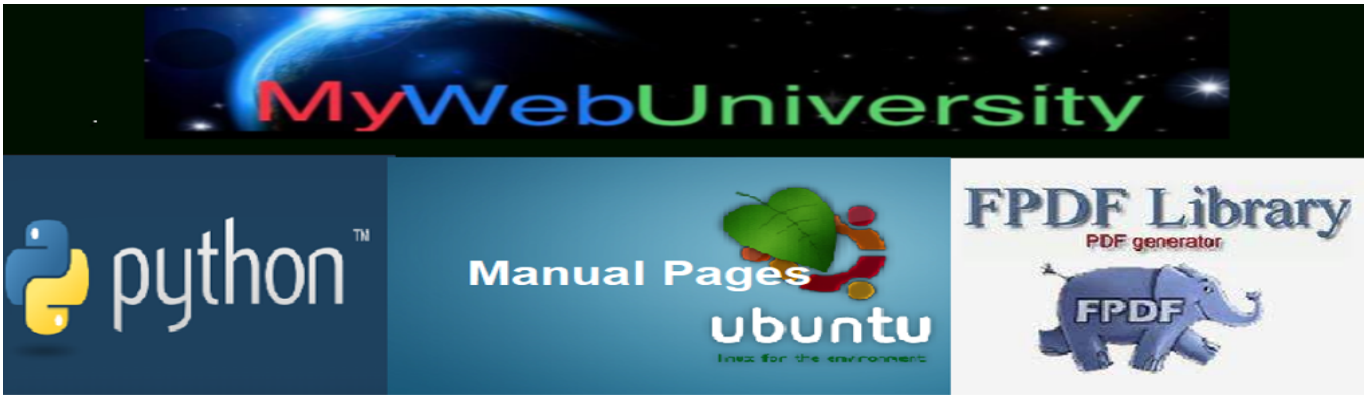

Rocky Enterprise Linux 9.2 Manual Pages on command 'DBD::Gofer::Transport::corostream.3pm'

$ man DBD::Gofer::Transport::corostream.3pm

DBD::Gofer::Transport::corostreUserpContributed Perl DocumeDBD::Gofer::Transport::corostream(3pm)

NAME

DBD::Gofer::Transport::corostream - Async DBD::Gofer stream transport using Coro and AnyEvent

SYNOPSIS

```
DBI_AUTOPROXY="dbi:Gofer:transport=corostream" perl some-perl-script-using-dbi.pl
```

or

```
$dsn = ...; # the DSN for the driver and database you want to use
$dbh = DBI->connect("dbi:Gofer:transport=corostream;dsn=$dsn", ...);
```

DESCRIPTION

The BIG WIN from using Coro is that it enables the use of existing DBI frameworks like DBIx::Class.

KNOWN ISSUES AND LIMITATIONS

- Uses Coro::Select so alters CORE::select globally

  Parent class probably needs refactoring to enable a more encapsulated approach.

- Doesn't prevent multiple concurrent requests

  Probably just needs a per-connection semaphore

- Coro has many caveats. Caveat emptor.

STATUS

THIS IS CURRENTLY JUST A PROOF-OF-CONCEPT IMPLEMENTATION FOR EXPERIMENTATION.

Please note that I have no plans to develop this code further myself. I'd very much welcome contributions. Interested? Let me know!

AUTHOR

Tim Bunce, <http://www.tim.bunce.name>

LICENCE AND COPYRIGHT

SEE ALSO

DBD::Gofer::Transport::stream

DBD::Gofer

APPENDIX

Example code:

```
#!perl
use strict;
use warnings;
use Time::HiRes qw(time);
BEGIN { $ENV{PERL_ANYEVENT_STRICT} = 1; $ENV{PERL_ANYEVENT_VERBOSE} = 1; }
use AnyEvent;
BEGIN { $ENV{DBI_TRACE} = 0; $ENV{DBI_GOFER_TRACE} = 0; $ENV{DBD_GOFER_TRACE} = 0; };
use DBI;
$ENV{DBI_AUTOPROXY} = 'dbi:Gofer:transport=corostream';
my $ticker = AnyEvent->timer( after => 0, interval => 0.1, cb => sub {
    warn sprintf "-tick- %.2f\n", time
} );
warn "connecting...\n";
my $dbh = DBI->connect("dbi:NullP:");
warn "...connected\n";
for (1..3) {
    warn "entering DBI...\n";
    $dbh->do("sleep 0.3"); # pseudo-sql understood by the DBD::NullP driver
    warn "...returned\n";
}
warn "done.";
```

Example output:

```
$ perl corogofer.pl
```

```
    connecting...
    -tick- 1293631437.14
    -tick- 1293631437.14
    ...connected
    entering DBI...
    -tick- 1293631437.25
    -tick- 1293631437.35
    -tick- 1293631437.45
    -tick- 1293631437.55
    ...returned
    entering DBI...
    -tick- 1293631437.66
    -tick- 1293631437.76
    -tick- 1293631437.86
    ...returned
    entering DBI...
    -tick- 1293631437.96
    -tick- 1293631438.06
    -tick- 1293631438.16
    ...returned
    done. at corogofer.pl line 39.
```

You can see that the timer callback is firing while the code 'waits' inside the do() method for the response from the database. Normally that would block.