NAME

DBI::Gofer::Execute - Executes Gofer requests and returns Gofer responses

SYNOPSIS

```
$executor = DBI::Gofer::Execute->new( { ...config... });

$response = $executor->execute_request( $request );
```

DESCRIPTION

Accepts a DBI::Gofer::Request object, executes the requested DBI method calls, and returns a DBI::Gofer::Response object.

Any error, including any internal 'fatal' errors are caught and converted into a DBI::Gofer::Response object.

This module is usually invoked by a 'server-side' Gofer transport module. They usually have names in the `DBI::Gofer::Transport::*` namespace. Examples include: DBI::Gofer::Transport::stream and DBI::Gofer::Transport::mod_perl.

CONFIGURATION

check_request_sub

If defined, it must be a reference to a subroutine that will 'check' the request. It is passed the request object and the executor as its only arguments.

The subroutine can either return the original request object or die with a suitable error message (which will be turned into a Gofer response).

It can also construct and return a new request that should be executed instead of the original request.

check_response_sub

If defined, it must be a reference to a subroutine that will 'check' the response. It is passed the response object, the executor, and the request object. The sub may alter the response object and return undef, or return a new response object.

This mechanism can be used to, for example, terminate the service if specific database errors are seen.

forced_connect_dsn

If set, this DSN is always used instead of the one in the request.

default_connect_dsn

If set, this DSN is used if `forced_connect_dsn` is not set and the request does not contain a DSN itself.

forced_connect_attributes

A reference to a hash of *connect()* attributes. Individual attributes in `forced_connect_attributes` will take precedence over corresponding attributes in the request.

default_connect_attributes

A reference to a hash of *connect()* attributes. Individual attributes in the request take precedence over corresponding attributes in `default_connect_attributes`.

max_cached_dbh_per_drh

If set, the loaded drivers will be checked to ensure they don't have more than this number of cached connections. There is no default value. This limit is not enforced for every request.

max_cached_sth_per_dbh

If set, all the cached statement handles will be cleared once the number of cached statement handles rises above this limit. The default is 1000.

forced_single_resultset

If true, then only the first result set will be fetched and returned in the response.

forced_response_attributes

A reference to a data structure that can specify extra attributes to be returned in responses.

```
forced_response_attributes => {
DriverName => {
dbh => [ qw(dbh_attrib_name) ],
sth => [ qw(sth_attrib_name) ],
},
},
```

This can be useful in cases where the driver has not implemented the *private_attribute_info()* method and DBI::Gofer::Execute's own fallback list of private attributes doesn't include the driver or attributes you need.

track_recent

If set, specifies the number of recent requests and responses that should be kept by the *update_stats()* method for diagnostics. See DBI::Gofer::Transport::mod_perl.

Note that this setting can significantly increase memory use. Use with caution.

forced_gofer_random

Enable forced random failures and/or delays for testing. See "DBI_GOFER_RANDOM" below.

DRIVER-SPECIFIC ISSUES

Gofer needs to know about any driver-private attributes that should have their values sent back to the client.

If the driver doesn't support *private_attribute_info()* method, and very few do, then the module fallsback to using some hard-coded details, if available, for the driver being used. Currently hard-coded details are available for the mysql, Pg, Sybase, and SQLite drivers.

TESTING

DBD::Gofer, DBD::Execute and related packages are well tested by executing the DBI test suite with DBI_AUTOPROXY configured to route all DBI calls via DBD::Gofer.

Because Gofer includes timeout and 'retry on error' mechanisms there is a need for some way to trigger delays and/or errors. This can be done via the `forced_gofer_random` configuration item, or else the DBI_GOFER_RANDOM environment variable.

DBI_GOFER_RANDOM

The value of the `forced_gofer_random` configuration item (or else the DBI_GOFER_RANDOM environment variable) is treated as a series of tokens separated by commas.

The tokens can be one of three types:

fail=R%
: Set the current failure rate to R where R is a percentage. The value R can be floating point, e.g., `fail=0.05%`. Negative values for R have special meaning, see below.

err=N
: Sets the current failure err value to N (instead of the DBI's default 'standard err value' of 2000000000). This is useful when you want to simulate a specific error.

delayN=R%
: Set the current random delay rate to R where R is a percentage, and set the current delay duration to N seconds. The values of R and N can be floating point, e.g., `delay0.5=0.2%`. Negative values for R have special meaning, see below.

  If R is an odd number (R % 2 == 1) then a message is logged via *warn()* which will be returned to, and echoed at, the client.

methodname
: Applies the current fail, err, and delay values to the named method. If neither a fail nor delay have been set yet then a warning is generated.

For example:

```
$executor = DBI::Gofer::Execute->new( {
forced_gofer_random => "fail=0.01%,do,delay60=1%,execute",
} );
```

will cause the *do()* method to fail for 0.01% of calls, and the *execute()* method to fail 0.01% of calls and be delayed by 60 seconds on 1% of calls.

If the percentage value (`R`) is negative then instead of the failures being triggered randomly (via the *rand()* function) they are triggered via a sequence number. In other words `fail=-20%` will mean every fifth call will fail. Each method has a distinct sequence number.

AUTHOR

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

LICENCE AND COPYRIGHT