NAME

DBI::DBD::Metadata - Generate the code and data for some DBI metadata methods

SYNOPSIS

The idea is to extract metadata information from a good quality ODBC driver and use it to generate code and data to use in your own DBI driver for the same database.

To generate code to support the get_info method:

```
perl -MDBI::DBD::Metadata -e "write_getinfo_pm('dbi:ODBC:dsn-name','user','pass','Driver')"

perl -MDBI::DBD::Metadata -e write_getinfo_pm dbi:ODBC:foo_db username password Driver
```

To generate code to support the type_info method:

```
perl -MDBI::DBD::Metadata -e "write_typeinfo_pm('dbi:ODBC:dsn-name','user','pass','Driver')"

perl -MDBI::DBD::Metadata -e write_typeinfo_pm dbi:ODBC:dsn-name user pass Driver
```

Where `dbi:ODBC:dsn-name` is the connection to use to extract the data, and `Driver` is the name of the driver you want the code generated for (the driver name gets embedded into the output in numerous places).

Generating a GetInfo package for a driver

The `write_getinfo_pm` in the DBI::DBD::Metadata module generates a DBD::Driver::GetInfo package on standard output.

This method generates a DBD::Driver::GetInfo package from the data source you specified in the parameter list or in the environment variable DBI_DSN. DBD::Driver::GetInfo should help a DBD author implement the DBI *get_info()* method. Because you are just creating this package, it is very unlikely that DBD::Driver already provides a good implementation for *get_info()*. Thus you will probably connect via DBD::ODBC.

Once you are sure that it is producing reasonably sane data, you should typically redirect the standard output to lib/DBD/Driver/GetInfo.pm, and then hand edit the result. Do not forget to update your Makefile.PL and MANIFEST to include this as an extra PM file that should be installed.

If you connect via DBD::ODBC, you should use version 0.38 or greater;

Please take a critical look at the data returned! ODBC drivers vary dramatically in their quality.

The generator assumes that most values are static and places these values directly in the `%info` hash. A few examples show the use of CODE references and the implementation via subroutines. It is very likely that you will have to write additional subroutines for values depending on the session state or server version, e.g. SQL_DBMS_VER.

A possible implementation of *DBD::Driver::db::get_info()* may look like:

```
sub get_info {
my($dbh, $info_type) = @_;
require DBD::Driver::GetInfo;
my $v = $DBD::Driver::GetInfo::info{int($info_type)};
$v = $v->($dbh) if ref $v eq 'CODE';
return $v;
}
```

Please replace Driver (or "<foo>") with the name of your driver. Note that this stub function is generated for you by write_getinfo_pm function, but you must manually transfer the code to Driver.pm.

Generating a TypeInfo package for a driver

The `write_typeinfo_pm` function in the DBI::DBD::Metadata module generates on standard output the data needed for a driver's type_info_all method. It also provides default

implementations of the type_info_all method for inclusion in the driver's main implementation file.

The driver parameter is the name of the driver for which the methods will be generated; for the sake of examples, this will be ''Driver''. Typically, the dsn parameter will be of the form ''dbi:ODBC:odbc_dsn'', where the odbc_dsn is a DSN for one of the driver's databases. The user and pass parameters are the other optional connection parameters that will be provided to the DBI connect method.

Once you are sure that it is producing reasonably sane data, you should typically redirect the standard output to lib/DBD/Driver/TypeInfo.pm, and then hand edit the result if necessary. Do not forget to update your Makefile.PL and MANIFEST to include this as an extra PM file that should be installed.

Please take a critical look at the data returned! ODBC drivers vary dramatically in their quality.

The generator assumes that all the values are static and places these values directly in the `%info` hash.

A possible implementation of *DBD::Driver::type_info_all()* may look like:

```
sub type_info_all {
my ($dbh) = @_;
require DBD::Driver::TypeInfo;
return [ @$DBD::Driver::TypeInfo::type_info_all ];
}
```

Please replace Driver (or ''<foo>'') with the name of your driver. Note that this stub function is generated for you by the write_typeinfo_pm function, but you must manually transfer the code to Driver.pm.

AUTHORS

Jonathan Leffler <jleffler@us.ibm.com> (previously <jleffler@informix.com>), Jochen Wiedmann <joe@ispsoft.de>, Steffen Goeldner <sgoeldner@cpan.org>, and Tim Bunce <dbi-users@perl.org>.