NAME

sqlsharp − Mono SQL Query command-line tool

SYNOPSIS

sqlsharp [−f filename] [−o filename] [−s]

DESCRIPTION

sqlsharp is a Mono SQL tool used for entering SQL queries to a database using Mono data providers.

OPTIONS

The following options are supported:

-f filename
Output file to load SQL commands from.

-o filename
Output file to send results.

-s Silent mode.

HOW TO USE

The SQL tool accepts commands via its command line interface. Commands begin with a backslash followed by the command name.

Example:
\open

Basically, there are five commands a user should know: \provider, \connectionstring, \open, \quit, and \help

To connect to a database, you need to do the following:

1. set your data provider via \provider

Example:
SQL# \provider mysql

2. set your connection string via \connectionstring

Example:
SQL# \connectionstring Database=test

3. open a connection to the database via \open

Example:
SQL# \open

CONNECTION AND PROVIDER COMMANDS

These commands are used to setup the provider, connection string, and open/close the database connnection

ConnectionString
Sets the Connection String

Example:
SQL# \ConnectionString Database=testdb
or
SQL# \cs Database=testdb

For more examples, see section CONNECTION STRING EXAMPLES.

Provider

Sets the Provider of the Data Source. For list of Providers, see section PROVIDERS.

Example: to set the provider for MySQL:

SQL# \provider mysql

or

SQL# \p mysql

Note: if you need to load an external provider in SQL#,
see the SQL# command \loadextprovider

ListProviders

List ADO.NET 2.0 Providers available

Example:

SQL# \ListProviders

or

SQL# \listp

BCS Prompts you for building each connection parameter and builds the connection string and also allows you to enter a password wich does not echo.

Example:

SQL# \bcs

ConnectionString Option: Data Source [] SQL# blazer

ConnectionString Option: Persist Security Info [False] SQL#

ConnectionString Option: Integrated Security [False] SQL#

ConnectionString Option: User ID [] SQL# scott

Password: *****

ConnectionString Option: Enlist [False] SQL#

ConnectionString Option: Pooling [True] SQL#

ConnectionString Option: Min Pool Size [0] SQL#

ConnectionString Option: Max Pool Size [100] SQL#

ConnectionString Option: Unicode [False] SQL#

ConnectionString Option: Load Balance Timeout [0] SQL#

ConnectionString Option: Omit Oracle Connection Name [False] SQL#
ConnectionString is set.

LoadExtProvider

ASSEMBLY CLASS to load an external provider. Use the complete name of its assembly and its Connection class.

Example: to load the MySQL provider Mono.Data.MySql
SQL# \loadextprovider Mono.Data.MySql Mono.Data.MySql.MySqlConnection

Open Opens a connection to the database

Example:
SQL# \open

Close Closes the connection to the database

Example:
SQL# \close

Default
show default variables, such as, Provider and ConnectionString.

Example:
SQL# \defaults

Q Quit

Example:
SQL# \q

SQL EXECUTION COMMANDS

Commands to execute SQL statements

e execute SQL query (SELECT)

Example: to execute a query

```
SQL# SELECT * FROM EMPLOYEE
SQL# \e
```

Note: to get \e to automatically work after entering a query, put a semicolon ; at the end of the query.

Example: to enter and exectue query at the same time

```
SQL# SELECT * FROM EMPLOYEE;
```

exenonquery
execute a SQL non query (not a SELECT)

Example: to insert a row into a table:

```
SQL# INSERT INTO SOMETABLE (COL1, COL2) VALUES('ABC','DEF')
SQL# \exenonquery
```

Note: this can be used for those providers that are new and do not have the ability to execute queries yet.

exescalar
execute SQL to get a single row and single column.

Example: to execute a Maxium aggregate
SQL# SELECT MAX(grade) FROM class
SQL# \exescalar

exexml FILENAME to execute SQL and save output to XML file

Example:
SQL# SELECT fname, lname, hire_date FROM employee
SQL# \exexml employee.xml

Note: this depends on DataAdapter, DataTable, and DataSet
to be working properly

FILE COMMANDS

Commands for importing commands from file to SQL# and vice versa

f FILENAME to read a batch of SQL# commands from file

Example:
SQL# \f batch.sql#

Note: the SQL# commands are interpreted as they are read. If there is
any SQL statements, the are executed.

o FILENAME to write result of commands executed to file.

Example:
SQL# \o result.txt

load FILENAME to load from file SQL commands into SQL buffer.

Example:
SQL# \load commands.sql

save FILENAME to save SQL commands from SQL buffer to file.

Example:
SQL# \save commands.sql

GENERAL PURPOSE COMMANDS

General commands to use.

h show help (all commands).

Example:
SQL# \h

s TRUE, FALSE to silent messages.

Example 1:
SQL# \s true

Example 2:
SQL# \s false

r reset or clear the query buffer.

Example:
SQL# \r

print show what's in the SQL buffer now.

Example:
SQL# \print

SH VARIABLES WHICH CAN BE USED AS PARAMETERS Commands to set variables which can be used as Parameters in an SQL statement. If the SQL contains any parameters, the parameter does not have a variable set, the user will be prompted for the value for each missing parameter.

set NAME VALUE to set an internal variable.

Example:
SQL# \set sFirstName John

unset NAME to remove an internal variable.

Example:
SQL# \unset sFirstName

variable
NAME to display the value of an internal variable.

Example:
SQL# \variable sFirstName

PROVIDER SUPPORT OPTIONS

Enable or Disble support for a particular provider option

UseParameters
TRUE,FALSE to use parameters when executing SQL which use the variables that were set.

If this option is true, the SQL contains parameters, and for each parameter which does not have a SQL# variable set, the user will be prompted to enter the value For that parameter.

Example:
SQL# \useparameter true

Default: false

UseSimpleReader
TRUE,FALSE to use simple reader when displaying results.

Example:

SQL# \usesimplereader true

Default: false. Mostly, this is dependent on the provider. If the provider does not have enough of IDataReader implemented to have the normal reader working, then the simple reader can be used. Providers like SqlClient, MySQL, and PostgreSQL have this ption defaulting to true.

PROVIDERS

PROVIDER NAME NAMESPACE ASSEMBLY

oracle Oracle 8i-11g System.Data.OracleClient System.Data.OracleClient
postgresql NetPostgreSQL Npgsql Npgsql
bytefx ByteFX MySQL ByteFX.Data.MySqlClient ByteFX.Data
sqlclient MS SQL 7-2008 System.Data.SqlClient System.Data
odbc ODBC System.Data.Odbc System.Data
sqlite SQL Lite Mono.Data.SqliteClient Mono.Data.SqliteClient
sybase Sybase Mono.Data.SybaseClient Mono.Data.SybaseClient
firebird Firebird SQL FirebirdSql.Data.FirebirdSql FirebirdSql.Data.Firebird
mysql MySQL AB MySql.Data.MySqlClient MySql.Data

NOTES:

Ngsql is the .Net Data Provider for PosgreSQL. The
latest version can be downloaded from
http://npgsql.projects.postgresql.org/

MySql.Data is the MySQL Connector/Net for connecting to MySql databases.
For MySQL, it is strongly recommend to use MySql.Data instead of the old
ByteFX.Data provider. Unfortunately, MySql.Data is not included with Mono.
You can download the latest MySQL Connector/Net from MySQL AB at
http://dev.mysql.com/downloads/

FirebirdSql.Data.Firebird can be downloaded from here:
http://www.firebirdsql.org/index.php?op=files&id=netprovider

CONNECTION STRING SAMPLES

Example connection strings for various providers to be used via the command \ConnectionString

Example of usage:

\connectionstring Database=testdb

Connection String examples:

Microsoft SQL Server via System.Data.SqlClient

Server=DANPC;Database=pubs;User ID=saPassword=;

For Integrated Security, bear in mind that Mono is not
integrated with Windows, SQL Server client nor server, nor
Windows Server. Therefore, you must provide the Windows Domain
name and domain user name and password for this user.

Server=DANPC;Database=pubs;User ID=DOMAINser;Password=pass;Integrated Security=SSPI

For a server locally, you can use localhost.

ODBC via System.Data.Odbc provider using
a DSN named "MSSQLDSN" I set up
in the Windows control panel's ODBC Data Sources
which connects to Microsoft SQL Server 2000:

DSN=MSSQLDSN;UID=danmorg;PWD=freetds

To use ODBC ON Unix, consider unixODBC from http://www.unixodbc.org/
or use iODBC from http://www.iodbc.org/

SQL Lite via Mono.Data.SqliteClient
provider which connects to the
database file SqliteTest.db; if not found,
the file is created:

URI=file:SqliteTest.db

Oracle via System.Data.OracleClient

Data Source=testdb;User ID=scott;Password=tiger

If you prefer to not use a tnsnames.ora file, you can
use a connection string which allows a
TNS network description that is parentheses delimited
like the following which has the host, port, and
service name. For host, you can specify an IP address
instead of a hostname.

User ID=SCOTT;
Password=TIGER;
Data Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=192.168.1.101)(PORT=1521))(CONNECT_DA

Npgsql (.NET PostgreSQL) from
http://gborg.postgresql.org/project/npgsql/projdisplay.php

Server=localhost;Database=test;User ID=postgres;Password=fun2db

ByteFX (ByteFX MySQL) from

Please use MySql.Data instead.

Server=localhost;Database=test;User ID=mysql;Password=

FirebirdSql via FirebirdSql.Data.Firebird (download latest form FirebirdSql.org)

Database=C:\FIREBIRD\EXAMPLES\EMPLOYEE.FDB;User=SYSDBA;Password=masterkey;Dialect=3;Serv

MySQL via (MySql.Data) MySQL Connector/Net from http://www.mysql.com/

Server=localhost;Database=test;User ID=mysql;Password=mypass;Pooling=false

TRACING SUPPORT

No support for tracing right now.

AUTHORS

The Mono SQL Query Tool was written
by Daniel Morgan <monodanmorg@yahoo.com>

LICENSE

The Mono SQL Query Tool is released under the terms of the GNU GPL. Please read the accompanying 'COPYING' file for details. Alternative licenses are available from Novell or Daniel Morgan.

BUGS

To report bugs in the compiler, you can file bug reports in our bug tracking system:
https://github.com/mono/mono/issues

MAILING LISTS

For details, visit:
http://lists.ximian.com/mailman/listinfo/mono-devel-list

WEB SITE

For details, visit:
http://www.mono-project.com

SEE ALSO

mono(1)