Package 'RJDBC'

March 10, 2020

Version 0.2-8

Title Provides Access to Databases Through the JDBC Interface

Author Simon Urbanek <Simon.Urbanek@r-project.org>

Maintainer Simon Urbanek <Simon.Urbanek@r-project.org>

Depends methods, DBI, rJava (>= 0.4-15), R (>= 2.4.0)

Description The RJDBC package is an implementation of R's DBI interface using JDBC as a back-end. This allows R to connect to any DBMS that has a JDBC driver.

License MIT + file LICENSE

URL http://www.rforge.net/RJDBC/

NeedsCompilation no

Repository CRAN

Date/Publication 2020-03-10 07:57:59 UTC

R topics documented:

JDBC *JDBC engine*

Description

JDBC creates a new DBI driver that can be used to start JDBC connections.

Usage

```
JDBC (driverClass = "", classPath = "", identifier.quote = NA)
```

Arguments

driverClass	name of the Java class of the JDBC driver to load. If empty, it is assumed that corresponding JDBC drivers were loaded by other means.
classPath	class path that needs to be appended in order to load the desired JDBC driver. Usually it is the path to the JAR file containing the driver.
identifier.quote	character to use for quoting identifiers in automatically generated SQL statements or NA if the back-end doesn't support quoted identifiers. See details section below.

Details

JDBC function has two purposes. One is to initialize the Java VM and load a Java JDBC driver (not to be confused with the JDBCDriver R object which is actually a DBI driver). The second purpose is to create a proxy R object which can be used to a call dbConnect which actually creates a connection.

JDBC requires a JDBC driver for a database-backend to be loaded. Usually a JDBC driver is supplied in a Java Archive (jar) file. The path to such a file can be specified in classPath. The driver itself has a Java class name that is used to load the driver (for example the MySQL driver uses com.mysql.jdbc.Driver), this has to be specified in driverClass.

Due to the fact that JDBC can talk to a wide variety of databases, the SQL dialect understood by the database is not known in advance. Therefore the RJDBC implementation tries to adhere to the SQL92 standard, but not all databases are compliant. This affects mainly functions such as dbWriteTable that have to automatically generate SQL code. One major ability is the support for quoted identifiers. The SQL92 standard uses double-quotes, but many database engines either don't support it or use other character. The identifier.quote parameter allows you to set the proper quote character for the database used. For example MySQL would require identifier.quote="`". If set to NA, the ability to quote identifiers is disabled, which poses restrictions on the names that can be used for tables and fields. Other functionality is not affected.

As of RDJBC 0.2-2 JDBC-specific stored procedure calls starting with {call and {?= call are supported in the statements.

Value

Returns a JDBCDriver object that can be used in calls to dbConnect.

See Also

dbConnect

Examples

```
## Not run:
drv <- JDBC("com.mysql.jdbc.Driver",
  "/etc/jdbc/mysql-connector-java-3.1.14-bin.jar", "`")
conn <- dbConnect(drv, "jdbc:mysql://localhost/test")
dbListTables(conn)
data(iris)
dbWriteTable(conn, "iris", iris)
dbGetQuery(conn, "select count(*) from iris")
d <- dbReadTable(conn, "iris")

## End(Not run)
```

JDBCConnection-class *Class "JDBCConnection"*

Description

Class representing a (DBI) database connection which uses JDBC to connect to a database.

Generators

Objects can be created by call to dbConnect of a JDBC driver.

Slots

jc: Java object representing the connection.

identifier.quote: Quote character to use for quoting identifiers it automatically generated SQL statements or NA if the back-end doesn't support quoted identifiers. Usually the value is inherited from the "JDBCDriver".

Extends

Class "DBIConnection-class", directly. Class "DBIObject-class", by class "DBIConnection", distance 2.

Methods

No methods defined with class "JDBCConnection" in the signature.

See Also

JDBC, "JDBCDriver"

JDBCConnection-methods

Methods for the class 'JDBCConnect' in Package 'RJDBC'

Description

Methods for the class 'JDBCConnection' in Package 'RJDBC'.

`dbSendQuery` and `dbSendUpdate` submit a SQL query to the database. The difference between the two is only that `dbSendUpdate` is used with DBML queries and thus doesn't return any result set.

`dbGetTables` and `dbGetFields` are similar to `dbListTables` and `dbListFields` but the result is a data frame with all available details (whereas the latter return only a character vector of the names).

Usage

```
dbSendUpdate (conn, statement, ...)
dbGetTables (conn, ...)
dbGetFields (conn, ...)
```

Arguments

`conn`	connection object
`statement`	SQL statement to execute
`...`	additional arguments to prepared statement substituted for "?"

Methods

dbCommit `signature(conn = "JDBCConnection",...)`
dbDataType `signature(dbObj = "JDBCConnection",obj = "ANY",...)`
dbDisconnect `signature(conn = "JDBCConnection",...)`
dbExistsTable `signature(conn = "JDBCConnection",name = "character",...)`
dbGetException `signature(conn = "JDBCConnection",...)`
dbGetFields `signature(conn = "JDBCConnection",...)`
dbGetInfo `signature(conn = "JDBCConnection",...)`
dbGetQuery `signature(conn = "JDBCConnection",statement = "character",...)`
dbGetTables `signature(conn = "JDBCConnection",...)`
dbListFields `signature(conn = "JDBCConnection",...)`
dbListResults `signature(conn = "JDBCConnection",...)`
dbListTables `signature(conn = "JDBCConnection",...)`
dbReadTable `signature(conn = "JDBCConnection",...)`
dbRemoveTable `signature(conn = "JDBCConnection",...)`
dbRollback `signature(conn = "JDBCConnection",...)`
dbSendQuery `signature(conn = "JDBCConnection",statement = "character",...)`
dbSendUpdate `signature(conn = "JDBCConnection",statement = "character",...)`
dbWriteTable `signature(conn = "JDBCConnection",...)`

JDBCDriver-class *Class "JDBCDriver"*

Description

A DBI driver that uses any JDBC driver to access databases.

Generators

Objects can be created by calls to JDBC or dbDriver.

Slots

identifier.quote: Quote character to use for identifiers in automatically generated SQL statements or NA if quoted identifiers are not supported by the back-end.

jdrv: Java object reference to an instance of the driver if the driver can be instantiated by a default constructor. This object is only used as a fall-back when the driver manager fails to find a driver.

Extends

Class "DBIDriver-class", directly. Class "DBIObject-class", by class "DBIDriver", distance 2.

Methods

No methods defined with class "JDBCDriver" in the signature.

See Also

JDBC

JDBCDriver-methods *Methods for the class JDBCDriver in Package 'RJDBC'*

Description

Methods for the class 'JDBCDriver' in Package 'RJDBC'.

Most prominent method is dbConnect, it creates a new JDBC connection using the specified driver. Due to the fact that the actual JDBC driver is selected in the URL, the JDBCDriver object itself has little significance and is not used to determine the connection type.

dbListConnections always return NULL with a warning, because JDBC connecteions are not tracked.

dbGetInfo returns very basic information, because the JDBC driver is not loaded until a connection is created.

dbUnloadDriver is a no-op in the current implementation, because drivers are never removed from the JVM.

Methods

dbConnect `signature(drv = "JDBCDriver",...)`

dbListConnections `signature(drv = "JDBCDriver",...)`

dbGetInfo `signature(drv = "JDBCDriver",...)`

dbUnloadDriver `signature(drv = "JDBCDriver",...)`

JDBCResult-class *Class "JDBCResult"*

Description

Representation of a DBI result set returned from a JDBC connection.

Generators

Objects can be created by call to `dbSendQuery`.

Slots

`jr`: Java reference to the JDBC result set

`md`: Java reference to the JDBC result set meta data

`env`: Environment holding cached objects (currently the result helper object used by `fetch()`)

`stat`: Java reference to the JDBC statement which generated this result

Extends

Class `"DBIResult-class"`, directly. Class `"DBIObject-class"`, by class "DBIResult", distance 2.

Methods

No methods defined with class "JDBCResult" in the signature.

See Also

`JDBC`, `dbSendQuery`

JDBCResult-methods *Methods for the class JDBCResult in Package 'RJDBC' ~~*

Description

Methods for the class 'JDBCResult' in Package 'RJDBC'.

`fetch` retrieves the content of the result set in the form of a data frame. If `n` is -1 then the current implementation fetches 32k rows first and then (if not sufficient) continues with chunks of 512k rows, appending them. If the size of the result set is known in advance, it is most efficient to set `n` to that size.

Additional argument `block` can be used to inform the driver that pre-fetching of a certain block of records is desirable (see `setFetchSize()` in JDBC) leading to possibly faster pulls of large queries. The default is to pre-fetch 2048 records. Note that some databases (like Oracle) don't support a fetch size of more than 32767. If set to `NA` or anything less than 1 then the fetch size is not changed. This is only a hint, drivers are free to ignore it.

Finally, `use.label` logical argument determines whether column lables are used for naming (`TRUE`, default) or column names should be used (`FALSE`) since some database drivers do not implement labels correctly.

`dbClearResult` releases the result set.

`dbColumnInfo` returns basic information about the columns (fields) in the result set.

`dbGetInfo` returns an empty list.

`dbListResults` returns an empty list and warns that JDBC doesn't track results.

Methods

fetch `signature(res = "JDBCResult",...)`

dbClearResult `signature(res = "JDBCResult",...)`

dbColumnInfo `signature(res = "JDBCResult",...)`

dbGetInfo `signature(res = "JDBCResult",...)`

dbListResults `signature(res = "JDBCResult",...)`

Index