Package: sjdbc (via r-universe)

October 26, 2026

Version 1.6.1

Title JDBC Driver Interface

Author TIBCO Software Inc.

Maintainer Joe Roberts <jorobert@tibco.com>

Description Provides a database-independent JDBC interface.

License BSD_3_clause + file LICENSE

Depends rJava

NeedsCompilation no

Repository https://cran.r-universe.dev

Date/Publication 2021-04-30 21:40:02 UTC

RemoteUrl https://github.com/cran/sjdbc

RemoteRef HEAD

RemoteSha 64623d8f2900f9aa7b2db4aa6586e7cd6a85cb02

Contents

executeJDBC 2
exportJDBC 3
importJDBC 5
jdbcTimeDate 6
jdbcTypeInfo 7
loadJDBCDriver 8
sjdbc 8
sjdbcCloseConnection 9
sjdbcGetResultSet 10
sjdbcOptions 11

Index **13**

executeJDBC *Execute SQL Query on a JDBC-Compatible Database*

Description

Executes a SQL command on a JDBC-Compatible database.

Usage

```
executeJDBC(sqlQuery, driverClass, con, user, password, keepAlive)
```

Arguments

sqlQuery	a string containing the SQL query to execute.
driverClass	a string containing the name of the Java class for required JDBC driver.
con	a string containing the JDBC connection string.
user	a string containing the user name with access to database.
password	a string containing the password for the given user name on the database.
keepAlive	a logical. If TRUE, keeps the database connection alive after executing the query. The default is FALSE.

Details

Executes the command on the database using the JDBC driver specified in driverClass. The required JDBC driver must be loaded in sjdbc before it is used. See loadJDBCDriver for details.

Database connections are closed by default after executing the query, unless keepAlive is set to TRUE. If keepAlive = TRUE, the connection remains open, and successive database commands can reuse the open connection if and only if the same values for driverClass, con, user, and password are supplied.

Value

returns the number of rows affected, if applicable.

Note

Some arguments can also be set using sjdbcOptions.

See Also

loadJDBCDriver, sjdbcOptions

Examples

```
## Not run:
executeJDBC(driverClass="com.microsoft.sqlserver.jdbc.SQLServerDriver",
        con="jdbc:sqlserver://qadb-s2k:1433;databaseName=testdb;user=testqa;password=testqa;",
            user="testqa", password="testqa",
            sqlQuery="UPDATE TEST1 SET Weight = NULL WHERE Weight < 2500")

executeJDBC(driverClass="com.microsoft.sqlserver.jdbc.SQLServerDriver",
        con="jdbc:sqlserver://qadb-s2k:1433;databaseName=testdb;user=testqa;password=testqa;",
            user="testqa", password="testqa",
            sqlQuery="DROP TABLE TEST1")

## End(Not run)
```

`exportJDBC` *Export To a JDBC-Compatible Database*

Description

Exports data to a database using JDBC drivers.

Usage

```
exportJDBC(data, table, appendToTable = TRUE,
    driverClass = sjdbcOptions()$driverClass, con = sjdbcOptions()$con,
    user = sjdbcOptions()$user, password = sjdbcOptions()$password,
    keepAlive = sjdbcOptions()$keepAlive, preserveColumnCase = FALSE,
    batchSize = sjdbcOptions()$batchSize,
    useTransaction = sjdbcOptions()$useTransaction)
```

Arguments

`data`	the `data.frame` object to export.
`table`	a string containing the name of the database table.
`appendToTable`	a logical. If `TRUE` (the default), rows are appended to the existing table; if `FALSE`, any existing table is dropped and an empty table is created prior to exporting the data.
`driverClass`	a string containing the name of the Java class for the required JDBC driver.
`con`	a string specifying the JDBC connection string.
`user`	a string containing the user name with access to database.
`password`	a string containing the password for the given user name on the database.
`keepAlive`	a logical. If `TRUE`, keeps the database connection alive after executing the query. The default is `FALSE`.

`preserveColumnCase`
: a logical. If `TRUE`, preserves case-sensitive column names, if supported by database. If `FALSE` (the default), column name case is converted to the database-specific default.

`batchSize`
: an integer specifying the number of rows sent to the database in each batch, if batch updates are supported by the JDBC driver. Default value is `1000`. A value of `0` disables batch exporting.

`useTransaction`
: If `TRUE`, exports the data as a single transaction, otherwise commits throughout export.

Details

Exports data to the database using the JDBC driver specified in `driverClass`. The required JDBC driver must be loaded in `sjdbc` before use. See `loadJDBCDriver` for details.

Database connections are closed by default after the query executes, unless `keepAlive` is set to `TRUE`. If `keepAlive = TRUE`, the connection remains open, and successive database commands can reuse the open connection if and only if the same values for `driverClass`, `con`, `user`, and `password` are supplied.

Setting a larger value for the `batchSize` argument can improve efficiency when you need to export large data tables, if batch updates are supported by the JDBC driver.

Value

returns the number of rows exported.

Note

Some arguments can also be set using `sjdbcOptions`.

When you export to a new table (`appendToTable=FALSE`), you might find that the column types of the resulting table are not as desired. Columns containing text data are of type `VARCHAR(255)` (or database equivalent), and numeric and `timeDate` columns attempt to use appropriate database-specific column types. If you want a specific column type or precision in your tables, you should create the table manually using `executeJDBC`, and then append your data to the existing table.

See Also

`loadJDBCDriver`, `sjdbcOptions`, `executeJDBC`

Examples

```
## Not run:
exportJDBC(data=fuel.frame, driverClass="com.microsoft.sqlserver.jdbc.SQLServerDriver",
        con="jdbc:sqlserver://qadb-s2k:1433;databaseName=testdb;user=testqa;password=testqa;",
            user="testqa", password="testqa",
            table="TEST1", append=F)

## End(Not run)
```

importJDBC *Import From a JDBC-Compatible Database*

Description

Imports data from a database using JDBC drivers.

Usage

```
importJDBC(sqlQuery, table, driverClass = sjdbcOptions()$driverClass,
    con = sjdbcOptions()$con, user = sjdbcOptions()$user,
    password = sjdbcOptions()$password,
    keepAlive = sjdbcOptions()$keepAlive, bigdata = FALSE)
```

Arguments

sqlQuery	the SQL query string describing the data to be retreived from the database. Required if table is not provided.
table	a string specifying the name of the table to import. Required if sqlQuery is not provided. Implies sqlQuery="SELECT * FROM <table>".
driverClass	a string containing the name of the Java class for the required JDBC driver.
con	the JDBC connection string.
user	a string specifying the user name with access to the database.
password	a string containing the password for the given user name on the database.
keepAlive	a logical. If TRUE, keeps the database connection alive after executing the query. The default is FALSE.
bigdata	unsupported in this version. Exists for compatibility with Spotfire S+.

Details

Imports data from the database using the JDBC driver specified in driverClass. The required JDBC driver must be loaded in sjdbc before use. See loadJDBCDriver for details.

Database connections are closed by default after the query executes, unless keepAlive is set to TRUE. If keepAlive = TRUE, the connection remains open, and successive database commands can reuse the open connection if and only if the same values for driverClass, con, user, and password are supplied.

Value

returns a data.frame containing the requested data.

Time Zone Handling

Times, Dates, and Timestamps that the datebase returns are assumed to be GMT. The resulting timeDate objects are created in GMT, without conversion. If you know the time zone of the incoming data, you can specify an alternative time zone for the timeDate objects by setting options("time.zone") prior to import. For further details, see the class.timeDate help file.

Note

Character data can be imported either as `character` or as `factor`. `importJDBC` uses the value of `options(stringsAsFactors)` to determine how to import the data.

Some arguments can also be set using `sjdbcOptions`.

See Also

`loadJDBCDriver`, `sjdbcOptions`

Examples

```
## Not run:
importJDBC(driverClass="com.microsoft.sqlserver.jdbc.SQLServerDriver",
    con="jdbc:sqlserver://qadb-s2k:1433;databaseName=testdb;user=testqa;password=testqa;",
      sqlQuery="SELECT * FROM FUEL_FRAME")

importJDBC(driverClass="COM.ibm.db2.jdbc.net.DB2Driver",
     con="jdbc:db2://qadb1:6789/QATESTDB",
     user="testqa",
     password="testqa",
     sqlQuery="SELECT * FROM FUEL_FRAME")

## End(Not run)
```

---

jdbcTimeDate — *Convert an splusTimeDate::timeDate object to standard JDBC Timestamp string*

Description

Converts a `timeDate` vector to a `character` vector in the standard format expected by `java.sql.Timestamp`: `yyyy-mm-dd hh:mm:ss.fffffffff` (in GMT)

Usage

```
jdbcTimeDate(data)
```

Arguments

`data`	a `timeDate` vector.

Value

returns a `character` vector in the specified format.

See Also

`exportJDBC`

Examples

```
my.td <- as.POSIXct("2011/1/1")
jdbcTimeDate(my.td)
```

---

jdbcTypeInfo *Retrieve Supported Column Type Info from a Database*

Description

Retrieves a table containing the data types supported by the connected database.

Usage

```
jdbcTypeInfo(driverClass, con, user, password, keepAlive)
```

Arguments

driverClass	a string specifying the name of the Java class for the required JDBC driver.
con	the JDBC connection string.
user	a string specifying the user name with access to the database.
password	a string containing the password for given the user name on the database.
keepAlive	a logical. If TRUE, keeps the database connection alive after executing the query. The default is FALSE.

Details

A direct interface to the java.sql.DatabaseMetaData.getTypeInfo() method. See the Java documentation for description of the fields in the table. Useful for debugging.

Value

returns a data.frame containing the entire table.

References

2004. https://docs.oracle.com/javase/1.5.0/docs/api/java/sql/DatabaseMetaData.html#getTypeInfo(). *Java SE Developer Documentation*. Redwood Shores, CA: Oracle Corporation.

Examples

```
## Not run:
jdbcTypeInfo(driverClass="com.microsoft.sqlserver.jdbc.SQLServerDriver",
        con="jdbc:sqlserver://qadb-s2k:1433;databaseName=testdb;user=testqa;password=testqa;",
            user="testqa", password="testqa")

## End(Not run)
```

loadJDBCDriver *Load a JDBC Driver*

Description

Makes a JDBC Driver available to the sjdbc package.

Usage

```
loadJDBCDriver(driverJar)
```

Arguments

driverJar	a vector of one or more strings containing the full paths to JDBC driver jars.

Details

Makes the specified driver jars available to the sjdbc package. The driver must be loaded prior to its first use in the TIBCO Enterprise Runtime for R session.

Note

The JDBC drivers need to be loaded each time you use the sjdbc package. To load a driver automatically when loading the sjdbc package, place it in the in the drivers folder where the sjdbc package is installed.

Examples

```
## Not run:
loadJDBCDriver(file.path("C:", "sqljdbc.jar"))

## End(Not run)
```

sjdbc *SJDBC Package Documentation*

Description

The SJDBC Package provides an interface to databases using Java's JDBC connectivity.

Details

Provides an interface to a databases using JDBC drivers. You can get JDBC drivers from the software providers. Place the JAR or ZIP file containing the JDBC drivers in the `drivers` folder under the package installation directory. All files placed in this directory are added automatically to the Java `CLASSPATH` when the package is loaded. Alternatively, drivers can be loaded explicitly at runtime using `loadJDBCDriver`.

The interface has been tested with the following drivers:

- Microsoft SQL Server 2005
  - Connection String: `jdbc:sqlserver://<host>:1433;databaseName=<database>;user=<username>;passwor`
  - Driver Class: `com.microsoft.sqlserver.jdbc.SQLServerDriver`
- IBM DB2 Universal Database 7.2
  - Connection String: `jdbc:db2://<host>:6789/<database>`
  - Driver Class: `COM.ibm.db2.jdbc.net.DB2Driver`
- MySQL Connector/J 3.1.14
  - Connection String: `jdbc:mysql://<host>:3306/<database>`
  - Driver Class: `com.mysql.jdbc.Driver`
- Oracle 10g Release 2 10.2.0.4 (ojdbc14.jar)
  - Connection String: `jdbc:oracle:thin:@<host>:1521:<databaseSID>`
  - Driver Class: `oracle.jdbc.OracleDriver`
- PostgreSQL 8.3 (JDBC3 driver 8.3-603)
  - Connection String: `jdbc:postgresql://<host>:5432/<database>`
  - Driver Class: `org.postgresql.Driver`

Known Issues

- Missing values might not be handled correctly in all cases. `exportJDBC` handles missing (NA) values for `integer` and `numeric` class columns by creating `NULL` values in the database table. Currently, this does not work for `character` or `factor` columns. NA values are stored as "NA" in the table, but empty strings ("") are stored as empty strings.

`sjdbcCloseConnection` *Close a Persistent Database Connection*

Description

Closes any open persistent database connection.

Usage

```
sjdbcCloseConnection()
```

Details

Closes a connection that was made persistent previously using the `keepAlive` argument to one of the database functions.

This function is used primarily by `importJDBC`, `exportJDBC`, and `executeJDBC` to close connections after execution. It rarely needs to be called directly.

Value

returns no value.

See Also

`importJDBC`, `exportJDBC`, `executeJDBC`

Examples

```
## Not run:
  # close an open connection
  sjdbcCloseConnection()

## End(Not run)
```

`sjdbcGetResultSet` *Get a ResultSet From Static Java Class*

Description

Retreives a ResultSet previously stored in a static instance of `SJDBCResultSetUtilities` class as a `data.frame`.

Usage

```
sjdbcGetResultSet(key, unregister = TRUE, default.num.rows = NULL,
                  start.at.first=TRUE, rows.to.read=-1)
```

Arguments

`key`	a string containing the key into the hash table in `SJDBCResultSetUtilities` where the result was stored previously.
`unregister`	a logical value. If `TRUE` (the default), specifies that the ResultSet should be removed from the hash table after the data is returned.
`default.num.rows`	an integer containing the number of rows. When the ResultSet is of type `ResultSet.TYPE_FORWARD_ONLY`, the number of rows cannot be determined until after all of the data has been retrieved. If the ResultSet has more than the default number, the array sizes are doubled whenever the current capacity is reached. If the ResultSet is not of `TYPE_FORWARD_ONLY`, this argument is not used.

start.at.first	a logical. If TRUE (the default), set the ResultSet to start with the first row before reading. if FALSE, start with the current row.
rows.to.read	an integer specifying the maximum number of rows to read. If less than zero, read all rows in the result set.

Details

This function is called by importJDBC and usually is not called directly.

Value

returns a data.frame containing the ResultSet.

See Also

importJDBC

Examples

```
## Not run:
sjdbcGetResultSet("resultid")

## End(Not run)
```

---

sjdbcOptions *Package Options and Defaults*

Description

Stores presistent options and defaults for sjdbc package functions.

Usage

```
sjdbcOptions(...)
```

Arguments

...	you can provide no arguments. You can provide a list or vector of character strings as the only argument, or you can provide arguments in name=value form. See the **VALUE** and **SIDE EFFECTS** sections for more information.

Value

The sjdbcOptions function always returns a list, even if the list is of length 1.

- if no arguments are given, returns a list of current values for all options.
- if a character vector is given as the only argument, returns a list of current values for the options named in the character vector.
- if an object of mode "list" is given as the only argument, its components become the values for options with the corresponding names. The function returns a list of the option values before they were modified. Usually, the list given as an argument is the return value of a previous call to sjdbcOptions.
- if arguments are given in name=value form, sjdbcOptions changes the values of the specified options and returns a list of the option values before they were modified.

Side Effects

When options are set, the sjdbcOptions function changes a list named .sjdbcOptions in the session frame (frame 0). The components of .sjdbcOptions are all of the currently defined options. If sjdbcOptions is called with either a list as the single argument or with one or more arguments in name=value form, the options specified are changed or created.

Supported Options

driverClass	a string containing the name of the Java class for the required JDBC driver.
con	the JDBC connection string.
user	a string specifying the user name with access to database. *Note:* Some drivers do not require this optio
password	a string containing the password for the given user name on the database. *Note:* Some drivers do not re
keepAlive	a logical. if TRUE, keeps the database connection alive after executing the query. Defaults to FALSE.
batchSize	an integer containing the number of rows exported per batch in exportJDBC. Defaults to 1000.
useTransaction	export data as a single transaction. Defaults to TRUE.

See Also

This function closely mimics the behavior of the options function in base TIBCO Enterprise Runtime for R.

Examples

```
# set a single option
sjdbcOptions(driverClass="COM.ibm.db2.jdbc.net.DB2Driver")

# set multiple options
sjdbcOptions(driverClass="COM.ibm.db2.jdbc.net.DB2Driver",
     con="jdbc:db2://qadb1:6789/QATESTDB",
     user="testqa",
     password="testqa")
```

Index

∗ **file**
  exportJDBC, 3
  importJDBC, 5
  sjdbc, 8
  sjdbcGetResultSet, 10
∗ **interface**
  executeJDBC, 2
  exportJDBC, 3
  importJDBC, 5
  jdbcTimeDate, 6
  jdbcTypeInfo, 7
  loadJDBCDriver, 8
  sjdbc, 8
  sjdbcCloseConnection, 9
  sjdbcGetResultSet, 10
  sjdbcOptions, 11

executeJDBC, 2, *4*, *10*
exportJDBC, 3, *6*, *10*

importJDBC, 5, *10, 11*

jdbcTimeDate, 6
jdbcTypeInfo, 7

loadJDBCDriver, *2*, *4*, *6*, 8, *9*

sjdbc, 8
sjdbcCloseConnection, 9
sjdbcGetResultSet, 10
sjdbcOptions, *2*, *4*, *6*, 11