Package 'rocker'

January 5, 2022

Title Database Interface Class

Version 0.3.0

Description
'R6' class interface for handling relational database connections using 'DBI' package as backend.
The class allows handling of connections to e.g. PostgreSQL, MariaDB and SQLite.
The purpose is having an intuitive object allowing straightforward handling of SQL databases.

License MIT + file LICENSE

URL https://github.com/nikolaus77/rocker

BugReports https://github.com/nikolaus77/rocker/issues

Encoding UTF-8

Imports DBI, R6, sodium

Suggests covr, crayon, knitr, RMariaDB, testthat (>= 3.0.0),
rmarkdown, RPostgres, RSQLite

RoxygenNote 7.1.2

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Nikolaus Pawlowski [aut, cre, cph]

Maintainer Nikolaus Pawlowski <niko@fr33.net>

Repository CRAN

Date/Publication 2022-01-04 23:20:02 UTC

R topics documented:

appendTable 2
begin 3
canConnect 4
clearResult 5
columnInfo 6
commit 7

connect . . . 8
createTable . . . 9
disconnect . . . 10
execute . . . 11
existsTable . . . 12
fetch . . . 13
getInfoCon . . . 14
getInfoDrv . . . 15
getInfoRes . . . 16
getQuery . . . 17
getRowCount . . . 18
getRowsAffected . . . 19
getStatement . . . 20
hasCompleted . . . 21
isValidCon . . . 22
isValidDrv . . . 23
isValidRes . . . 24
listFields . . . 25
listObjects . . . 26
listTables . . . 27
newDB . . . 28
readTable . . . 28
removeTable . . . 29
rocker-R6-class . . . 30
rocker-README . . . 56
rocker-S3-functions . . . 78
rollback . . . 78
sendQuery . . . 79
sendStatement . . . 80
setupDriver . . . 81
setupMariaDB . . . 82
setupPostgreSQL . . . 83
setupSQLite . . . 85
unloadDriver . . . 86
validateCon . . . 86
writeTable . . . 87

Index **89**

`appendTable` *Append data to table.*

Description

Append data to table.

Usage

```
appendTable(db, name, value, ...)
```

Arguments

db	rocker object
name	Table name
value	Values data.frame
...	Optional, additional suitable parameters passed to DBI::dbAppendTable()

Value

Number of appended rows invisibly

See Also

Other rocker-S3-functions: begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::createTable(db, "mtcars", mtcars)
rocker::appendTable(db, "mtcars", mtcars)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

---

begin *Begin transaction.*

Description

Begin transaction.

Usage

```
begin(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to DBI::dbBegin()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::begin(db)
rocker::sendStatement(db, "DELETE FROM mtcars WHERE gear = 3;")
rocker::clearResult(db)
rocker::commit(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

---

canConnect *Test connection parameters.*

Description

Test connection parameters.

Usage

```
canConnect(db, ...)
```

Arguments

db	rocker object
...	Optional, suitable parameters passed to DBI::dbCanConnect()

Value

TRUE or FALSE

See Also

Other rocker-S3-functions: appendTable(), begin(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::canConnect(db)
rocker::unloadDriver(db)
```

---

clearResult *Clear query or statement result.*

Description

Clear query or statement result.

Usage

```
clearResult(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to DBI::dbClearResult()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::sendQuery(db, "SELECT * FROM mtcars;")
output <- rocker::fetch(db)
rocker::clearResult(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

`columnInfo` *Information on query result columns.*

Description

Information on query result columns.

Usage

```
columnInfo(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to `DBI::dbColumnInfo()`

Value

Information table

See Also

Other rocker-S3-functions: `appendTable()`, `begin()`, `canConnect()`, `clearResult()`, `commit()`, `connect()`, `createTable()`, `disconnect()`, `execute()`, `existsTable()`, `fetch()`, `getInfoCon()`, `getInfoDrv()`, `getInfoRes()`, `getQuery()`, `getRowCount()`, `getRowsAffected()`, `getStatement()`, `hasCompleted()`, `isValidCon()`, `isValidDrv()`, `isValidRes()`, `listFields()`, `listObjects()`, `listTables()`, `readTable()`, `removeTable()`, `rocker-README`, `rocker-S3-functions`, `rocker-package`, `rollback()`, `sendQuery()`, `sendStatement()`, `setupDriver()`, `setupMariaDB()`, `setupPostgreSQL()`, `setupSQLite()`, `unloadDriver()`, `validateCon()`, `writeTable()`

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::sendQuery(db, "SELECT * FROM mtcars;")
```

```
rocker::columnInfo(db)
rocker::clearResult(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

`commit` *Commit transaction.*

Description

Commit transaction.

Usage

```
commit(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to `DBI::dbCommit()`

Value

Invisible self

See Also

Other rocker-S3-functions: `appendTable()`, `begin()`, `canConnect()`, `clearResult()`, `columnInfo()`, `connect()`, `createTable()`, `disconnect()`, `execute()`, `existsTable()`, `fetch()`, `getInfoCon()`, `getInfoDrv()`, `getInfoRes()`, `getQuery()`, `getRowCount()`, `getRowsAffected()`, `getStatement()`, `hasCompleted()`, `isValidCon()`, `isValidDrv()`, `isValidRes()`, `listFields()`, `listObjects()`, `listTables()`, `readTable()`, `removeTable()`, `rocker-README`, `rocker-S3-functions`, `rocker-package`, `rollback()`, `sendQuery()`, `sendStatement()`, `setupDriver()`, `setupMariaDB()`, `setupPostgreSQL()`, `setupSQLite()`, `unloadDriver()`, `validateCon()`, `writeTable()`

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::begin(db)
rocker::sendStatement(db, "DELETE FROM mtcars WHERE gear = 3;")
rocker::clearResult(db)
rocker::commit(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

connect *Establish database connection using stored parameters.*

Description

Establish database connection using stored parameters.

Usage

```
connect(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to DBI::dbConnect()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

createTable *Create empty formatted table.*

Description

Create empty formatted table.

Usage

```
createTable(db, name, fields, ...)
```

Arguments

db	rocker object
name	Table name
fields	Template data.frame
...	Optional, additional suitable parameters passed to DBI::dbCreateTable()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::createTable(db, "mtcars", mtcars)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

`disconnect` *Disconnect database.*

Description

Disconnect database.

Usage

```
disconnect(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to `DBI::dbDisconnect()`

Value

Invisible self

See Also

Other rocker-S3-functions: `appendTable()`, `begin()`, `canConnect()`, `clearResult()`, `columnInfo()`, `commit()`, `connect()`, `createTable()`, `execute()`, `existsTable()`, `fetch()`, `getInfoCon()`, `getInfoDrv()`, `getInfoRes()`, `getQuery()`, `getRowCount()`, `getRowsAffected()`, `getStatement()`, `hasCompleted()`, `isValidCon()`, `isValidDrv()`, `isValidRes()`, `listFields()`, `listObjects()`, `listTables()`, `readTable()`, `removeTable()`, `rocker-README`, `rocker-S3-functions`, `rocker-package`, `rollback()`, `sendQuery()`, `sendStatement()`, `setupDriver()`, `setupMariaDB()`, `setupPostgreSQL()`, `setupSQLite()`, `unloadDriver()`, `validateCon()`, `writeTable()`

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

execute	*Execute SQL statement in database. Combination of functions execute and clearResult. If required, database is automatically connected and disconnected.*

Description

Execute SQL statement in database. Combination of functions execute and clearResult. If required, database is automatically connected and disconnected.

Usage

```
execute(db, statement, ...)
```

Arguments

db	rocker object
statement	SQL statement (UPDATE,DELETE,INSERT INTO,...)
...	Optional, additional suitable parameters passed to DBI::dbSendStatement()

Value

Number of affected rows

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::execute(db, "DELETE FROM mtcars WHERE gear = 3;")
rocker::disconnect(db)
rocker::unloadDriver(db)
```

existsTable *Check if table exists.*

Description

Check if table exists.

Usage

```
existsTable(db, name, ...)
```

Arguments

db	rocker object
name	Table name
...	Optional, additional suitable parameters passed to DBI::dbExistsTable()

Value

TRUE or FALSE

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::existsTable(db, "mtcars")
rocker::disconnect(db)
rocker::unloadDriver(db)
```

`fetch` *Fetch SQL query result from database.*

Description

Fetch SQL query result from database.

Usage

```
fetch(db, n = -1, ...)
```

Arguments

db	rocker object
n	Number of record to be fetched
...	Optional, additional suitable parameters passed to `DBI::dbFetch()`

Value

Records

See Also

Other rocker-S3-functions: `appendTable()`, `begin()`, `canConnect()`, `clearResult()`, `columnInfo()`, `commit()`, `connect()`, `createTable()`, `disconnect()`, `execute()`, `existsTable()`, `getInfoCon()`, `getInfoDrv()`, `getInfoRes()`, `getQuery()`, `getRowCount()`, `getRowsAffected()`, `getStatement()`, `hasCompleted()`, `isValidCon()`, `isValidDrv()`, `isValidRes()`, `listFields()`, `listObjects()`, `listTables()`, `readTable()`, `removeTable()`, `rocker-README`, `rocker-S3-functions`, `rocker-package`, `rollback()`, `sendQuery()`, `sendStatement()`, `setupDriver()`, `setupMariaDB()`, `setupPostgreSQL()`, `setupSQLite()`, `unloadDriver()`, `validateCon()`, `writeTable()`

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::sendQuery(db, "SELECT * FROM mtcars;")
output <- rocker::fetch(db)
rocker::clearResult(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

getInfoCon *Information on connection object.*

Description

Information on connection object.

Usage

```
getInfoCon(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to DBI::dbGetInfo()

Value

Information list

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::getInfoCon(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

getInfoDrv *Information on driver object.*

Description

Information on driver object.

Usage

```
getInfoDrv(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to DBI::dbGetInfo()

Value

Information list

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::getInfoDrv(db)
rocker::unloadDriver(db)
```

`getInfoRes` *Information on result object.*

Description

Information on result object.

Usage

```
getInfoRes(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to `DBI::dbGetInfo()`

Value

Information list

See Also

Other rocker-S3-functions: `appendTable()`, `begin()`, `canConnect()`, `clearResult()`, `columnInfo()`, `commit()`, `connect()`, `createTable()`, `disconnect()`, `execute()`, `existsTable()`, `fetch()`, `getInfoCon()`, `getInfoDrv()`, `getQuery()`, `getRowCount()`, `getRowsAffected()`, `getStatement()`, `hasCompleted()`, `isValidCon()`, `isValidDrv()`, `isValidRes()`, `listFields()`, `listObjects()`, `listTables()`, `readTable()`, `removeTable()`, `rocker-README`, `rocker-S3-functions`, `rocker-package`, `rollback()`, `sendQuery()`, `sendStatement()`, `setupDriver()`, `setupMariaDB()`, `setupPostgreSQL()`, `setupSQLite()`, `unloadDriver()`, `validateCon()`, `writeTable()`

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::sendQuery(db, "SELECT * FROM mtcars;")
rocker::getInfoRes(db)
rocker::clearResult(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

getQuery *Retrieve SQL query from database. Combination of functions sendQuery(), fetch() and clearResult(). If required, database is automatically connected and disconnected.*

Description

Retrieve SQL query from database. Combination of functions sendQuery(), fetch() and clearResult(). If required, database is automatically connected and disconnected.

Usage

```
getQuery(db, statement, n = -1, ...)
```

Arguments

db	rocker object
statement	SQL query (SELECT)
n	Number of record to be fetched at once. All records will be fetched.
...	Optional, additional suitable parameters passed to DBI::dbSendQuery()

Value

Records

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
output <- rocker::getQuery(db, "SELECT * FROM mtcars;")
rocker::disconnect(db)
rocker::unloadDriver(db)
```

getRowCount *Information on number of retrieved rows.*

Description

Information on number of retrieved rows.

Usage

```
getRowCount(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to DBI::dbGetRowCount()

Value

Number of retrieved rows

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::sendQuery(db, "SELECT * FROM mtcars;")
output <- rocker::fetch(db)
rocker::getRowCount(db)
rocker::clearResult(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

getRowsAffected *Information on number of affected rows.*

Description

Information on number of affected rows.

Usage

```
getRowsAffected(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to DBI::dbGetRowsAffected()

Value

Number of affected rows

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::sendStatement(db, "DELETE FROM mtcars WHERE gear = 3;")
rocker::getRowsAffected(db)
rocker::clearResult(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

`getStatement` *Information on sent statement.*

Description

Information on sent statement.

Usage

```
getStatement(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to `DBI::dbGetStatement()`

Value

Statement text

See Also

Other rocker-S3-functions: `appendTable()`, `begin()`, `canConnect()`, `clearResult()`, `columnInfo()`, `commit()`, `connect()`, `createTable()`, `disconnect()`, `execute()`, `existsTable()`, `fetch()`, `getInfoCon()`, `getInfoDrv()`, `getInfoRes()`, `getQuery()`, `getRowCount()`, `getRowsAffected()`, `hasCompleted()`, `isValidCon()`, `isValidDrv()`, `isValidRes()`, `listFields()`, `listObjects()`, `listTables()`, `readTable()`, `removeTable()`, `rocker-README`, `rocker-S3-functions`, `rocker-package`, `rollback()`, `sendQuery()`, `sendStatement()`, `setupDriver()`, `setupMariaDB()`, `setupPostgreSQL()`, `setupSQLite()`, `unloadDriver()`, `validateCon()`, `writeTable()`

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::sendQuery(db, "SELECT * FROM mtcars;")
rocker::getStatement(db)
rocker::clearResult(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

`hasCompleted` *Information whether all records were retrieved.*

Description

Information whether all records were retrieved.

Usage

```
hasCompleted(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to `DBI::dbHasCompleted()`

Value

TRUE or FALSE

See Also

Other rocker-S3-functions: `appendTable()`, `begin()`, `canConnect()`, `clearResult()`, `columnInfo()`, `commit()`, `connect()`, `createTable()`, `disconnect()`, `execute()`, `existsTable()`, `fetch()`, `getInfoCon()`, `getInfoDrv()`, `getInfoRes()`, `getQuery()`, `getRowCount()`, `getRowsAffected()`, `getStatement()`, `isValidCon()`, `isValidDrv()`, `isValidRes()`, `listFields()`, `listObjects()`, `listTables()`, `readTable()`, `removeTable()`, `rocker-README`, `rocker-S3-functions`, `rocker-package`, `rollback()`, `sendQuery()`, `sendStatement()`, `setupDriver()`, `setupMariaDB()`, `setupPostgreSQL()`, `setupSQLite()`, `unloadDriver()`, `validateCon()`, `writeTable()`

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::sendQuery(db, "SELECT * FROM mtcars;")
output <- rocker::fetch(db, 5)
rocker::hasCompleted(db)
rocker::clearResult(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

`isValidCon` *Check connection object.*

Description

Check connection object.

Usage

```
isValidCon(db, onLostNull = FALSE, ...)
```

Arguments

db	rocker object
onLostNull	TRUE or FALSE. If connection lost, set .con to NULL
...	Optional, additional suitable parameters passed to DBI::dbIsValid()

Value

TRUE or FALSE

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::isValidCon(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

isValidDrv *Check driver object.*

Description

Check driver object.

Usage

```
isValidDrv(db, onLostNull = FALSE, ...)
```

Arguments

db	rocker object
onLostNull	TRUE or FALSE. If driver lost, set .drv to NULL
...	Optional, additional suitable parameters passed to DBI::dbIsValid()

Value

TRUE or FALSE

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::isValidDrv(db)
rocker::unloadDriver(db)
```

`isValidRes` *Check result object.*

Description

Check result object.

Usage

```
isValidRes(db, onLostNull = FALSE, ...)
```

Arguments

db	rocker object
onLostNull	TRUE or FALSE. If result lost, set .res to NULL
...	Optional, additional suitable parameters passed to DBI::dbIsValid()

Value

TRUE or FALSE

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::sendQuery(db, "SELECT * FROM mtcars;")
rocker::isValidRes(db)
rocker::clearResult(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

`listFields` *List table column names.*

Description

List table column names.

Usage

```
listFields(db, name, ...)
```

Arguments

db	rocker object
name	Table name
...	Optional, additional suitable parameters passed to DBI::dbListFields()

Value

Column names

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::listFields(db, "mtcars")
rocker::disconnect(db)
rocker::unloadDriver(db)
```

`listObjects` *List database objects.*

Description

List database objects.

Usage

```
listObjects(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to `DBI::dbListObjects()`

Value

List of objects

See Also

Other rocker-S3-functions: `appendTable()`, `begin()`, `canConnect()`, `clearResult()`, `columnInfo()`, `commit()`, `connect()`, `createTable()`, `disconnect()`, `execute()`, `existsTable()`, `fetch()`, `getInfoCon()`, `getInfoDrv()`, `getInfoRes()`, `getQuery()`, `getRowCount()`, `getRowsAffected()`, `getStatement()`, `hasCompleted()`, `isValidCon()`, `isValidDrv()`, `isValidRes()`, `listFields()`, `listTables()`, `readTable()`, `removeTable()`, `rocker-README`, `rocker-S3-functions`, `rocker-package`, `rollback()`, `sendQuery()`, `sendStatement()`, `setupDriver()`, `setupMariaDB()`, `setupPostgreSQL()`, `setupSQLite()`, `unloadDriver()`, `validateCon()`, `writeTable()`

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::listObjects(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

`listTables` *List database tables.*

Description

List database tables.

Usage

```
listTables(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to `DBI::dbListTables()`

Value

List of objects

See Also

Other rocker-S3-functions: `appendTable()`, `begin()`, `canConnect()`, `clearResult()`, `columnInfo()`, `commit()`, `connect()`, `createTable()`, `disconnect()`, `execute()`, `existsTable()`, `fetch()`, `getInfoCon()`, `getInfoDrv()`, `getInfoRes()`, `getQuery()`, `getRowCount()`, `getRowsAffected()`, `getStatement()`, `hasCompleted()`, `isValidCon()`, `isValidDrv()`, `isValidRes()`, `listFields()`, `listObjects()`, `readTable()`, `removeTable()`, `rocker-README`, `rocker-S3-functions`, `rocker-package`, `rollback()`, `sendQuery()`, `sendStatement()`, `setupDriver()`, `setupMariaDB()`, `setupPostgreSQL()`, `setupSQLite()`, `unloadDriver()`, `validateCon()`, `writeTable()`

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::listTables(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

newDB *newDB*

Description

Function generates a new R6 database handling interface with DBI backend. For more information, see rocker class description.

Usage

```
newDB(verbose = TRUE, id = NULL, ...)
```

Arguments

verbose	TRUE or FALSE. Switch text output on / off.
id	Optional object ID/name
...	Not used yet

Value

New instance of rocker class

See Also

Other rocker: rocker-R6-class, rocker-README, rocker-S3-functions, rocker-package

Examples

```
db <- rocker::newDB()
```

readTable *Read table.*

Description

Read table.

Usage

```
readTable(db, name, ...)
```

Arguments

db	rocker object
name	Table name
...	Optional, additional suitable parameters passed to DBI::dbReadTable()

Value

Table

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
output <- rocker::readTable(db, "mtcars")
rocker::disconnect(db)
rocker::unloadDriver(db)
```

---

removeTable *Remove table.*

Description

Remove table.

Usage

```
removeTable(db, name, ...)
```

Arguments

db	rocker object
name	Table name
...	Optional, additional suitable parameters passed to DBI::dbRemoveTable()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::removeTable(db, "mtcars")
rocker::disconnect(db)
rocker::unloadDriver(db)
```

---

rocker-R6-class *'rocker' Database Interface R6 Class*

Description

R6 class interface for handling database connections using DBI package as backend. The class allows handling of connections to e.g. PostgreSQL, MariaDB and SQLite.

Format

R6Class

Active bindings

`.drv` Read only DBI::DBIDriver-class object or NULL. It is recommended not to use this object! For advanced user, object can be used for direct execution of functions from DBI package.

`.con` Read only DBI::DBIConnection-class object or NULL. It is recommended not to use this object! For advanced user, object can be used for direct execution of functions from DBI package.

`.res` Read only DBI::DBIResult-class object or NULL. It is recommended not to use this object! For advanced user, object can be used for direct execution of functions from DBI package.

`transaction` Read only TRUE or FALSE. Information on pending transaction.

`info` Read only driver package and connection parameter information list.

`verbose` TRUE or FALSE. Switch text output on / off.

`id` Optional object ID/name

`validateQuery` SQL statement used in validateCon

Methods

Public methods:

- rocker$new()
- rocker$print()
- rocker$setupDriver()
- rocker$setupPostgreSQL()
- rocker$setupMariaDB()
- rocker$setupSQLite()
- rocker$unloadDriver()
- rocker$canConnect()
- rocker$connect()
- rocker$disconnect()
- rocker$sendQuery()
- rocker$getQuery()
- rocker$sendStatement()
- rocker$execute()
- rocker$fetch()
- rocker$hasCompleted()
- rocker$getRowsAffected()
- rocker$getRowCount()
- rocker$columnInfo()
- rocker$getStatement()
- rocker$clearResult()
- rocker$begin()
- rocker$commit()
- rocker$rollback()
- rocker$getInfoDrv()
- rocker$getInfoCon()
- rocker$getInfoRes()
- rocker$isValidDrv()
- rocker$isValidCon()
- rocker$isValidRes()
- rocker$validateCon()
- rocker$createTable()
- rocker$appendTable()
- rocker$writeTable()
- rocker$readTable()
- rocker$removeTable()
- rocker$existsTable()
- rocker$listFields()
- rocker$listObjects()
- rocker$listTables()

Method `new()`**:** Generate new instance of class.

Usage:

```
rocker$new(verbose = TRUE, id = NULL, ...)
```

Arguments:

`verbose` TRUE or FALSE. Switch text output on / off.

`id` Optional object ID/name

`...` Not used yet

Returns: New instance of class

Method `print()`**:** Print object information.

Usage:

```
rocker$print()
```

Returns: Invisible self

Method `setupDriver()`**:** Setup database driver and define connection parameters.

Usage:

```
rocker$setupDriver(drv, protect = c("password", "user"), ...)
```

Arguments:

`drv` Driver object from suitable package e.g. `RPostgres::Postgres()`, `RMariaDB::MariaDB()` and `RSQLite::SQLite()`

`protect` Parameters to be hidden

`...` Suitable parameters passed to `DBI::dbConnect()` e.g. host, port, dbname, user and password

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupDriver(
  drv = RSQLite::SQLite(),
  dbname = ":memory:"
)
db$unloadDriver()
```

Method `setupPostgreSQL()`**:** Setup database driver and define connection parameters for PostgreSQL using RPostgres package. Wrapper for setupDriver() function.

Usage:

```
rocker$setupPostgreSQL(
  host = "127.0.0.1",
  port = "5432",
  dbname = "mydb",
  user = "postgres",
  password = "password",
  protect = c("password", "user"),
  ...
)
```

Arguments:

`host` Host name or IP number

`port` Port number

`dbname` Database name

`user` User name

`password` Password

`protect` Parameters to be hidden

`...` Optional, additional suitable parameters passed to `DBI::dbConnect()`

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupPostgreSQL(
  host = "127.0.0.1", port = "5432", dbname = "mydb",
  user = "postgres", password = "password"
)
db$unloadDriver()
```

Method `setupMariaDB()`**:** Setup database driver and define connection parameters for MariaDB using RMariaDB package. Wrapper for setupDriver() function.

Usage:

```
rocker$setupMariaDB(
  host = "127.0.0.1",
  port = "3306",
  dbname = "mydb",
  user = "root",
  password = "password",
  protect = c("password", "user"),
  ...
)
```

Arguments:

`host` Host name or IP number

`port` Port number

`dbname` Database name

`user` User name

`password` Password

`protect` Parameters to be hidden

`...` Optional, additional suitable parameters passed to `DBI::dbConnect()`

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupMariaDB(
  host = "127.0.0.1", port = "3306", dbname = "mydb",
  user = "root", password = "password"
)
db$unloadDriver()
```

Method `setupSQLite()`**:** Setup database driver and define connection parameters for SQLite using RSQLite package. Wrapper for setupDriver() function.

Usage:

```
rocker$setupSQLite(dbname = ":memory:", protect = c("password", "user"), ...)
```

Arguments:

`dbname` Database name

`protect` Parameters to be hidden

`...` Optional, additional suitable parameters passed to `DBI::dbConnect()`

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite(
  dbname = ":memory:"
)
db$unloadDriver()
```

Method `unloadDriver()`**:** Reset database driver and connection parameters.

Usage:

```
rocker$unloadDriver(...)
```

Arguments:

`...` Optional, suitable parameters passed to `DBI::dbUnloadDriver()`

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$unloadDriver()
```

Method `canConnect()`**:** Test connection parameters.

Usage:

```
rocker$canConnect(...)
```

Arguments:

`...` Optional, suitable parameters passed to `DBI::dbCanConnect()`

Returns: TRUE or FALSE

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$canConnect()
db$unloadDriver()
```

Method `connect()`**:** Establish database connection using stored parameters.

Usage:

```
rocker$connect(...)
```

Arguments:

... Optional, additional suitable parameters passed to DBI::dbConnect()

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$disconnect()
db$unloadDriver()
```

Method `disconnect()`**:** Disconnect database.

Usage:

```
rocker$disconnect(...)
```

Arguments:

... Optional, additional suitable parameters passed to DBI::dbDisconnect()

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$disconnect()
db$unloadDriver()
```

Method `sendQuery()`**:** Send SQL query to database.

Usage:

```
rocker$sendQuery(statement, ...)
```

Arguments:

`statement` SQL query (`SELECT`)

... Optional, additional suitable parameters passed to DBI::dbSendQuery()

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
output <- db$fetch()
db$clearResult()
db$disconnect()
db$unloadDriver()
```

Method `getQuery()`**:** Retrieve SQL query from database. Combination of functions sendQuery(), fetch() and clearResult(). If required, database is automatically connected and disconnected.

Usage:

```
rocker$getQuery(statement, n = -1, ...)
```

Arguments:

`statement` SQL query (`SELECT`)

`n` Number of record to be fetched at once. All records will be fetched.

`...` Optional, additional suitable parameters passed to `DBI::dbSendQuery()`

Returns: Records

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
output <- db$getQuery("SELECT * FROM mtcars;")
db$disconnect()
db$unloadDriver()
```

Method `sendStatement()`**:** Send SQL statement to database.

Usage:

```
rocker$sendStatement(statement, ...)
```

Arguments:

`statement` SQL statement (`UPDATE`,`DELETE`,`INSERT INTO`,`...`)

`...` Optional, additional suitable parameters passed to `DBI::dbSendStatement()`

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendStatement("DELETE FROM mtcars WHERE gear = 3;")
db$clearResult()
db$disconnect()
db$unloadDriver()
```

Method `execute()`**:** Execute SQL statement in database. Combination of functions execute and clearResult. If required, database is automatically connected and disconnected.

Usage:

```
rocker$execute(statement, ...)
```

Arguments:

`statement` SQL statement (`UPDATE`,`DELETE`,`INSERT INTO`,`...`)

`...` Optional, additional suitable parameters passed to `DBI::dbSendStatement()`

Returns: Number of affected rows

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$execute("DELETE FROM mtcars WHERE gear = 3;")
db$disconnect()
db$unloadDriver()
```

Method `fetch()`**:** Fetch SQL query result from database.

Usage:

```
rocker$fetch(n = -1, ...)
```

Arguments:

`n` Number of record to be fetched

`...` Optional, additional suitable parameters passed to `DBI::dbFetch()`

Returns: Records

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
output <- db$fetch()
db$clearResult()
db$disconnect()
db$unloadDriver()
```

Method `hasCompleted()`**:** Information whether all records were retrieved.

Usage:

```
rocker$hasCompleted(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbHasCompleted()`

Returns: TRUE or FALSE

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
output <- db$fetch(5)
db$hasCompleted()
db$clearResult()
db$disconnect()
db$unloadDriver()
```

Method `getRowsAffected()`**:** Information on number of affected rows.

Usage:

```
rocker$getRowsAffected(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbGetRowsAffected()`

Returns: Number of affected rows

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendStatement("DELETE FROM mtcars WHERE gear = 3;")
db$getRowsAffected()
db$clearResult()
db$disconnect()
db$unloadDriver()
```

Method `getRowCount()`**:** Information on number of retrieved rows.

Usage:

```
rocker$getRowCount(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbGetRowCount()`

Returns: Number of retrieved rows

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
output <- db$fetch()
db$getRowCount()
db$clearResult()
db$disconnect()
db$unloadDriver()
```

Method `columnInfo()`**:** Information on query result columns.

Usage:

```
rocker$columnInfo(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbColumnInfo()`

Returns: Information table

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
db$columnInfo()
db$clearResult()
db$disconnect()
db$unloadDriver()
```

Method `getStatement()`**:** Information on sent statement.

Usage:

```
rocker$getStatement(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbGetStatement()`

Returns: Statement text

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
db$getStatement()
db$clearResult()
db$disconnect()
db$unloadDriver()
```

Method `clearResult()`**:** Clear query or statement result.

Usage:

```
rocker$clearResult(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbClearResult()`

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
output <- db$fetch()
db$clearResult()
db$disconnect()
db$unloadDriver()
```

Method `begin()`**:** Begin transaction.

Usage:

```
rocker$begin(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbBegin()`

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$begin()
db$sendStatement("DELETE FROM mtcars WHERE gear = 3;")
db$clearResult()
db$commit()
db$disconnect()
db$unloadDriver()
```

Method `commit()`**:** Commit transaction.

Usage:

```
rocker$commit(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbCommit()`

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$begin()
db$sendStatement("DELETE FROM mtcars WHERE gear = 3;")
db$clearResult()
db$commit()
db$disconnect()
db$unloadDriver()
```

Method `rollback()`**:** Rollback transaction.

Usage:

```
rocker$rollback(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbRollback()`

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$begin()
db$sendStatement("DELETE FROM mtcars WHERE gear = 3;")
db$clearResult()
db$rollback()
db$disconnect()
db$unloadDriver()
```

Method `getInfoDrv()`**:** Information on driver object.

Usage:

```
rocker$getInfoDrv(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbGetInfo()`

Returns: Information list

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$getInfoDrv()
db$unloadDriver()
```

Method `getInfoCon()`**:** Information on connection object.

Usage:

```
rocker$getInfoCon(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbGetInfo()`

Returns: Information list

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$getInfoCon()
db$disconnect()
db$unloadDriver()
```

Method `getInfoRes()`**:** Information on result object.

Usage:

```
rocker$getInfoRes(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbGetInfo()`

Returns: Information list

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
db$getInfoRes()
db$clearResult()
db$disconnect()
db$unloadDriver()
```

Method `isValidDrv()`**:** Check driver object.

Usage:

```
rocker$isValidDrv(onLostNull = FALSE, ...)
```

Arguments:

`onLostNull` TRUE or FALSE. If driver lost, set .drv to NULL

`...` Optional, additional suitable parameters passed to `DBI::dbIsValid()`

Returns: TRUE or FALSE

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$isValidDrv()
db$unloadDriver()
```

Method `isValidCon()`**:** Check connection object.

Usage:

```
rocker$isValidCon(onLostNull = FALSE, ...)
```

Arguments:

`onLostNull` TRUE or FALSE. If connection lost, set .con to NULL

`...` Optional, additional suitable parameters passed to `DBI::dbIsValid()`

Returns: TRUE or FALSE

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$isValidCon()
db$disconnect()
db$unloadDriver()
```

Method `isValidRes()`**:** Check result object.

Usage:

```
rocker$isValidRes(onLostNull = FALSE, ...)
```

Arguments:

`onLostNull` TRUE or FALSE. If result lost, set .res to NULL

`...` Optional, additional suitable parameters passed to `DBI::dbIsValid()`

Returns: TRUE or FALSE

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
db$isValidRes()
db$clearResult()
db$disconnect()
db$unloadDriver()
```

Method `validateCon()`**:** Check if an earlier opened connection is still open.

Usage:

```
rocker$validateCon(statement = NULL, onLostNull = FALSE, ...)
```

Arguments:

`statement` Optional SQL statement. If not set default validateQuery will be used.

`onLostNull` TRUE or FALSE. If connection lost, set .con to NULL

`...` Not used yet

Returns: TRUE or FALSE

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$validateCon()
db$disconnect()
db$unloadDriver()
```

Method `createTable()`**:** Create empty formatted table.

Usage:

```
rocker$createTable(name, fields, ...)
```

Arguments:

`name` Table name

`fields` Template data.frame

`...` Optional, additional suitable parameters passed to `DBI::dbCreateTable()`

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$createTable("mtcars", mtcars)
db$disconnect()
db$unloadDriver()
```

Method `appendTable()`**:** Append data to table.

Usage:

```
rocker$appendTable(name, value, ...)
```

Arguments:

`name` Table name

`value` Values data.frame

`...` Optional, additional suitable parameters passed to `DBI::dbAppendTable()`

Returns: Number of appended rows invisibly

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$createTable("mtcars", mtcars)
db$appendTable("mtcars", mtcars)
db$disconnect()
db$unloadDriver()
```

Method `writeTable()`**:** Write data to table.

Usage:

```
rocker$writeTable(name, value, ...)
```

Arguments:

`name` Table name

`value` Values data.frame

`...` Optional, additional suitable parameters passed to `DBI::dbWriteTable()`

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$disconnect()
db$unloadDriver()
```

Method `readTable()`**:** Read table.

Usage:

```
rocker$readTable(name, ...)
```

Arguments:

`name` Table name

`...` Optional, additional suitable parameters passed to `DBI::dbReadTable()`

Returns: Table

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
output <- db$readTable("mtcars")
db$disconnect()
db$unloadDriver()
```

Method `removeTable()`**:** Remove table.

Usage:

```
rocker$removeTable(name, ...)
```

Arguments:

`name` Table name

`...` Optional, additional suitable parameters passed to `DBI::dbRemoveTable()`

Returns: Invisible self

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$removeTable("mtcars")
db$disconnect()
db$unloadDriver()
```

Method `existsTable()`**:** Check if table exists.

Usage:

```
rocker$existsTable(name, ...)
```

Arguments:

`name` Table name

`...` Optional, additional suitable parameters passed to `DBI::dbExistsTable()`

Returns: TRUE or FALSE

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$existsTable("mtcars")
db$disconnect()
db$unloadDriver()
```

Method `listFields()`**:** List table column names.

Usage:

```
rocker$listFields(name, ...)
```

Arguments:

`name` Table name

`...` Optional, additional suitable parameters passed to `DBI::dbListFields()`

Returns: Column names

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$listFields("mtcars")
db$disconnect()
```

Method `listObjects()`**:** List database objects.

Usage:

```
rocker$listObjects(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbListObjects()`

Returns: List of objects

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$listObjects()
db$disconnect()
```

Method `listTables()`**:** List database tables.

Usage:

```
rocker$listTables(...)
```

Arguments:

`...` Optional, additional suitable parameters passed to `DBI::dbListTables()`

Returns: List of objects

Examples:

```
db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$listTables()
db$disconnect()
```

See Also

Other rocker: `newDB()`, `rocker-README`, `rocker-S3-functions`, `rocker-package`

Examples

```
# New database handling object
db <- rocker::newDB()
# Setup SQLite database
db$setupSQLite()
# Open connection
db$connect()
# Write table
db$writeTable("mtcars", mtcars)
# Get query
output <- db$getQuery("SELECT * FROM mtcars;")
# Close connection
db$disconnect()
# Reset database handling object
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$setupDriver`
## ------------------------------------------------

db <- rocker::newDB()
db$setupDriver(
  drv = RSQLite::SQLite(),
  dbname = ":memory:"
)
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$setupPostgreSQL`
## ------------------------------------------------

db <- rocker::newDB()
db$setupPostgreSQL(
  host = "127.0.0.1", port = "5432", dbname = "mydb",
  user = "postgres", password = "password"
)
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$setupMariaDB`
## ------------------------------------------------

db <- rocker::newDB()
db$setupMariaDB(
  host = "127.0.0.1", port = "3306", dbname = "mydb",
  user = "root", password = "password"
)
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$setupSQLite`
## ------------------------------------------------
```

```
db <- rocker::newDB()
db$setupSQLite(
  dbname = ":memory:"
)
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$unloadDriver`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$canConnect`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$canConnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$connect`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$disconnect`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$sendQuery`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
```

```
output <- db$fetch()
db$clearResult()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$getQuery`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
output <- db$getQuery("SELECT * FROM mtcars;")
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$sendStatement`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendStatement("DELETE FROM mtcars WHERE gear = 3;")
db$clearResult()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$execute`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$execute("DELETE FROM mtcars WHERE gear = 3;")
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$fetch`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
output <- db$fetch()
db$clearResult()
```

```
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$hasCompleted`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
output <- db$fetch(5)
db$hasCompleted()
db$clearResult()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$getRowsAffected`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendStatement("DELETE FROM mtcars WHERE gear = 3;")
db$getRowsAffected()
db$clearResult()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$getRowCount`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
output <- db$fetch()
db$getRowCount()
db$clearResult()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$columnInfo`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
```

```
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
db$columnInfo()
db$clearResult()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$getStatement`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
db$getStatement()
db$clearResult()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$clearResult`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
output <- db$fetch()
db$clearResult()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$begin`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$begin()
db$sendStatement("DELETE FROM mtcars WHERE gear = 3;")
db$clearResult()
db$commit()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$commit`
```

```
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$begin()
db$sendStatement("DELETE FROM mtcars WHERE gear = 3;")
db$clearResult()
db$commit()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$rollback`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$begin()
db$sendStatement("DELETE FROM mtcars WHERE gear = 3;")
db$clearResult()
db$rollback()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$getInfoDrv`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$getInfoDrv()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$getInfoCon`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$getInfoCon()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$getInfoRes`
## ------------------------------------------------

db <- rocker::newDB()
```

```
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
db$getInfoRes()
db$clearResult()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$isValidDrv`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$isValidDrv()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$isValidCon`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$isValidCon()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$isValidRes`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$sendQuery("SELECT * FROM mtcars;")
db$isValidRes()
db$clearResult()
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$validateCon`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$validateCon()
db$disconnect()
db$unloadDriver()
```

```
## ------------------------------------------------
## Method `rocker$createTable`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$createTable("mtcars", mtcars)
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$appendTable`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$createTable("mtcars", mtcars)
db$appendTable("mtcars", mtcars)
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$writeTable`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$readTable`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
output <- db$readTable("mtcars")
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$removeTable`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
```

```
db$connect()
db$writeTable("mtcars", mtcars)
db$removeTable("mtcars")
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$existsTable`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$existsTable("mtcars")
db$disconnect()
db$unloadDriver()

## ------------------------------------------------
## Method `rocker$listFields`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$listFields("mtcars")
db$disconnect()

## ------------------------------------------------
## Method `rocker$listObjects`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$listObjects()
db$disconnect()

## ------------------------------------------------
## Method `rocker$listTables`
## ------------------------------------------------

db <- rocker::newDB()
db$setupSQLite()
db$connect()
db$writeTable("mtcars", mtcars)
db$listTables()
db$disconnect()
```

rocker-README *README*

Description

How-to and examples

Details

Please read me.

Installation

Installation of current released version from CRAN

```
install.packages("rocker")
```

Installation of current development version from GitHub

```
install.packages("devtools")
devtools::install_github("nikolaus77/rocker")
```

New *rocker* class object

Create new *rocker* database handling object

Option 1

```
db <- rocker::newDB() # New database handling object
```

```
## dctr | New object
```

Option 2

```
db <- rocker::rocker$new() # New database handling object
```

```
## dctr | New object
```

Terminal output

Controlling terminal output

```
db <- rocker::newDB(verbose = TRUE) # New database handling object
```

```
## dctr | New object
```

```
db$setupPostgreSQL()
```

```
## Dctr | Driver load RPostgres

db$unloadDriver()

## dctr | Driver unload RPostgres

db$verbose <- FALSE # Terminal output off
db$setupPostgreSQL()
db$unloadDriver()

db$verbose <- TRUE # Terminal output on (default)
db$setupPostgreSQL()

## Dctr | Driver load RPostgres

db$unloadDriver()

## dctr | Driver unload RPostgres
```

Structure of terminal output

```
Dctr | Driver load RSQLite
D                          = Driver     (D = loaded,    d = not set)
 c                         = Connection (C = opened,    c = closed)
  t                        = Transation (T = active,    t = no tranastion)
   r                       = Result     (R = available, r = no result)
       Driver load RSQLite = Message text
```

Optional object ID

Optionally, rocker object can be labeled with an ID. This can be helpful in case terminal output of multiple rocker objects need to be distinguished.

```
db1 <- rocker::newDB(id = "myDB 1") # New database handling object with ID

## myDB 1 | dctr | New object id myDB 1

db2 <- rocker::newDB(id = "myDB 2") # New database handling object with ID

## myDB 2 | dctr | New object id myDB 2

db1$setupPostgreSQL()

## myDB 1 | Dctr | Driver load RPostgres

db2$setupMariaDB()

## myDB 2 | Dctr | Driver load RMariaDB
```

```
db1$unloadDriver()

## myDB 1 | dctr | Driver unload RPostgres

db2$unloadDriver()

## myDB 2 | dctr | Driver unload RMariaDB

db1$id <- NULL # Remove ID
db1$setupSQLite()

## Dctr | Driver load RSQLite

db1$unloadDriver()

## dctr | Driver unload RSQLite

db1$id <- "newID 1" # Add new ID
db1$setupSQLite()

## newID 1 | Dctr | Driver load RSQLite

db1$unloadDriver()

## newID 1 | dctr | Driver unload RSQLite
```

Object properties

Object properties are stored in the info field and can be displayed by print function.

```
db <- rocker::newDB() # New database handling object

## dctr | New object

db$setupPostgreSQL()

## Dctr | Driver load RPostgres

db$info

## $package
## [1] "RPostgres"
##
## $host
## [1] "127.0.0.1"
##
## $port
## [1] "5432"
##
## $dbname
## [1] "mydb"
```

```
db
```

```
## object
##   id             null
##   verbose        true
##   validateQuery null
## database
##   package        RPostgres
##   host           127.0.0.1
##   port           5432
##   dbname         mydb
## status
##   driver         true
##   connection     false
##   transaction    false
##   result         false
```

```
db$print()
```

```
## object
##   id             null
##   verbose        true
##   validateQuery null
## database
##   package        RPostgres
##   host           127.0.0.1
##   port           5432
##   dbname         mydb
## status
##   driver         true
##   connection     false
##   transaction    false
##   result         false
```

```
print(db)
```

```
## object
##   id             null
##   verbose        true
##   validateQuery null
## database
##   package        RPostgres
##   host           127.0.0.1
##   port           5432
##   dbname         mydb
## status
##   driver         true
##   connection     false
##   transaction    false
##   result         false
```

```
db$unloadDriver()

## dctr | Driver unload RPostgres
```

Connection validation – Is the earlier opened database connection still open?

```
db <- rocker::newDB() # New database handling object

## dctr | New object

db$setupSQLite()

## Dctr | Driver load RSQLite

db$print()

## object
##   id            null
##   verbose       true
##   validateQuery null
## database
##   package       RSQLite
##   dbname        :memory:
## status
##   driver        true
##   connection    false
##   transaction   false
##   result        false
```

During connection setup, a validateQuery is looked up automatically.

```
db$connect()

## DCtr | Database connected

db$print()

## object
##   id            null
##   verbose       true
##   validateQuery SELECT 1
## database
##   package       RSQLite
##   dbname        :memory:
## status
##   driver        true
##   connection    true
##   transaction   false
##   result        false
```

Discovered validateQuery

```
db$validateQuery
```

```
## [1] "SELECT 1"
```

Validate connection

```
db$validateCon()
```

```
## DCtr | Connection valid true
```

```
## [1] TRUE
```

If required, validateQuery can be defined manually.

```
db$validateQuery <- "SELECT 2"
db$validateCon()
```

```
## DCtr | Connection valid true
```

```
## [1] TRUE
```

```
db$print()
```

```
## object
##   id            null
##   verbose       true
##   validateQuery SELECT 2
## database
##   package       RSQLite
##   dbname        :memory:
## status
##   driver        true
##   connection    true
##   transaction   false
##   result        false
```

Clean up

```
db$disconnect()
```

```
## Dctr | Database disconnected
```

```
db$validateCon()
```

```
## Dctr | Connection valid false
```

```
## [1] FALSE
```

```
db$unloadDriver()
```

```
## dctr | Driver unload RSQLite
```

Additional packages and database types

The listed packages are required for some functions of *rocker*.

RSQLite package:

RSQLite package for handling of SQLite database connections. It is required for the setupSQLite() function of *rocker* class.

```
install.packages("RSQLite")
```

Setup database

Option 1

```
db <- rocker::newDB() # New database handling object

## dctr | New object

db$setupSQLite( # Setup SQLite database
  dbname = ":memory:"
)

## Dctr | Driver load RSQLite

db$unloadDriver() # Reset database handling object

## dctr | Driver unload RSQLite
```

Option 2

```
db <- rocker::newDB() # New database handling object

## dctr | New object

db$setupDriver( # Setup SQLite database
  drv = RSQLite::SQLite(),
  dbname = ":memory:"
)

## Dctr | Driver load RSQLite

db$unloadDriver() # Reset database handling object

## dctr | Driver unload RSQLite
```

RPostgres package:

RPostgres package for handling of PostgreSQL database connections. It is required for the setupPostgreSQL() function of *rocker* class.

```
install.packages("RPostgres")
```

Setup database

Option 1

```
db <- rocker::newDB() # New database handling object
#> dctr | New object
db$setupPostgreSQL( # Setup PostgreSQL database
  host = "127.0.0.1", port = "5432", dbname = "mydb",
  user = "postgres", password = "password"
)
#> Dctr | Driver load RPostgres
db$unloadDriver() # Reset database handling object
#> dctr | Driver unload RPostgres
```

Option 2

```
db <- rocker::newDB() # New database handling object
#> dctr | New object
db$setupDriver( # Setup PostgreSQL database
  drv = RPostgres::Postgres(),
  host = "127.0.0.1", port = "5432", dbname = "mydb",
  user = "postgres", password = "password"
)
#> Dctr | Driver load RPostgres
db$unloadDriver() # Reset database handling object
#> dctr | Driver unload RPostgres
```

***RMariaDB* package:**

RMariaDB package for handling of MariaDB and MySQL database connections. It is required for the setupMariaDB() function of *rocker* class.

```
install.packages("RMariaDB")
```

Setup database

Option 1

```
db <- rocker::newDB() # New database handling object
#> dctr | New object
db$setupMariaDB( # Setup MariaDB database
  host = "127.0.0.1", port = "3306", dbname = "mydb",
  user = "root", password = "password"
)
#> Dctr | Driver load RMariaDB
db$unloadDriver() # Reset database handling object
#> dctr | Driver unload RMariaDB
```

Option 2

```
db <- rocker::newDB() # New database handling object
#> dctr | New object
db$setupDriver( # Setup MariaDB database
  drv = RMariaDB::MariaDB(),
  host = "127.0.0.1", port = "3306", dbname = "mydb",
  user = "root", password = "password"
)
#> Dctr | Driver load RMariaDB
```

```
db$unloadDriver() # Reset database handling object
#> dctr | Driver unload RMariaDB
```

***crayon* package:**

The *crayon* package is required for colored terminal output. If missing terminal output is monochrome.

```
install.packages("crayon")
```

Database connection

There are different ways to open a connection and to get data.

Prepare database with a table

```
db <- rocker::newDB() # New database handling object

## dctr | New object

db$setupSQLite(dbname = tempfile()) # Setup SQLite database

## Dctr | Driver load RSQLite

db$connect() # Open connection

## DCtr | Database connected

db$writeTable("mtcars", mtcars) # Create table for testing

## DCtr | Write table mtcars columns mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, carb rows 32

db$disconnect() # Close connection

## Dctr | Database disconnected
```

Example 1

Get query with automatic connection / disconnection

```
output <- db$getQuery("SELECT * FROM mtcars;") # Get query

## DCtr | Database connected
## DCtR | Send query 21 characters
## DCtR | Fetch rows all -> Received 32 rows, 11 columns, 4824 bytes
## DCtR | Rows fetched 32
## DCtR | Has completed yes
## DCtr | Clear result
## Dctr | Database disconnected
```

Example 2

Get query with manual connection / disconnection

```
db$connect() # Open connection

## DCtr | Database connected

output1 <- db$getQuery("SELECT * FROM mtcars;") # Get query 1

## DCtR | Send query 21 characters
## DCtR | Fetch rows all -> Received 32 rows, 11 columns, 4824 bytes
## DCtR | Rows fetched 32
## DCtR | Has completed yes
## DCtr | Clear result

output2 <- db$getQuery("SELECT * FROM mtcars;", 15) # Get query 2

## DCtR | Send query 21 characters
## DCtR | Fetch rows 15 -> Received 15 rows, 11 columns, 3416 bytes
## DCtR | Rows fetched 15
## DCtR | Has completed no
## DCtR | Fetch rows 15 -> Received 15 rows, 11 columns, 3416 bytes
## DCtR | Rows fetched 30
## DCtR | Has completed no
## DCtR | Fetch rows 15 -> Received 2 rows, 11 columns, 2184 bytes
## DCtR | Rows fetched 32
## DCtR | Has completed yes
## DCtr | Clear result

db$disconnect() # Close connection

## Dctr | Database disconnected
```

Example 3

Function getQuery() is a combination of functions sendQuery(), fetch() and clearResult().

```
db$connect() # Open connection

## DCtr | Database connected

db$sendQuery("SELECT * FROM mtcars;") # Send query

## DCtR | Send query 21 characters

output <- db$fetch() # Fetch result

## DCtR | Fetch rows all -> Received 32 rows, 11 columns, 4824 bytes

db$clearResult() # Clean up result
```

```
## DCtr | Clear result

db$disconnect() # Close connection

## Dctr | Database disconnected
```

Clean up

```
db$unloadDriver() # Reset database handling object

## dctr | Driver unload RSQLite
```

Password storage

Some efforts were undertaken to encrypt and to protect the password in the private area of the class. The class stores the password hidden and inaccessible. **Please let me know, in case you discover a way how to access the password!**

```
db <- rocker::newDB() # New database handling object
#> dctr | New object
db$setupDriver( # Setup PostgreSQL database with stored password (password and user are hidden - default
  RPostgres::Postgres(),
  host = "127.0.0.1", port = "5432", dbname = "mydb",
  user = "postgres", password = "password",
  protect = c("password", "user")
)
#> Dctr | Driver load RPostgres

db$connect() # Open connection 1; Password is stored in the class and does not need to be provided.
#> DCtr | Database connected
output1 <- db$getQuery("SELECT * FROM mtcars;") # Get query 1
#> DCtR | Send query 21 characters
#> DCtR | Fetch rows all -> Received 32 rows, 11 columns, 4824 bytes
#> DCtR | Rows fetched 32
#> DCtR | Has completed yes
#> DCtr | Clear result
db$disconnect() # Close connection 1
#> Dctr | Database disconnected

db$connect() # Open connection 2; Password is stored in the class and does not need to be provided.
#> DCtr | Database connected
output2 <- db$getQuery("SELECT * FROM mtcars;") # Get query 2
#> DCtR | Send query 21 characters
#> DCtR | Fetch rows all -> Received 32 rows, 11 columns, 4824 bytes
#> DCtR | Rows fetched 32
#> DCtR | Has completed yes
#> DCtr | Clear result
db$disconnect() # Close connection 2
#> Dctr | Database disconnected
```

```
db$unloadDriver() # Reset database handling object
#> dctr | Driver unload RPostgres
```

In case you do not want to store the password in the class, you will need to provide it each time a connection is opened.

```
db <- rocker::newDB() # New database handling object
#> dctr | New object
db$setupDriver( # Setup PostgreSQL database without stored password
  RPostgres::Postgres(),
  host = "127.0.0.1", port = "5432", dbname = "mydb",
  user = "postgres"
)
#> Dctr | Driver load RPostgres

db$connect(password = "password") # Open connection 1; Password needs to be provided.
#> DCtr | Database connected
output1 <- db$getQuery("SELECT * FROM mtcars;") # Get query 1
#> DCtR | Send query 21 characters
#> DCtR | Fetch rows all -> Received 32 rows, 11 columns, 4824 bytes
#> DCtR | Rows fetched 32
#> DCtR | Has completed yes
#> DCtr | Clear result
db$disconnect() # Close connection 1
#> Dctr | Database disconnected

db$connect(password = "password") # Open connection 2; Password needs to be provided.
#> DCtr | Database connected
output2 <- db$getQuery("SELECT * FROM mtcars;") # Get query 2
#> DCtR | Send query 21 characters
#> DCtR | Fetch rows all -> Received 32 rows, 11 columns, 4824 bytes
#> DCtR | Rows fetched 32
#> DCtR | Has completed yes
#> DCtr | Clear result
db$disconnect() # Close connection 2
#> Dctr | Database disconnected

db$unloadDriver() # Reset database handling object
#> dctr | Driver unload RPostgres
```

DBI objects

rocker class encapsulates the *DBI* objects driver, connection and result. If required, these objects can be directly used with *DBI* functions. **However, it is recommended to use this option with care!** Direct usage of *DBI* functions, may disrupt proper function of *rocker* class. Many *DBI* functions are implemented in *rocker* class. Whenever possible, use the *rocker* class functions.

Prepare object

```
db <- rocker::newDB() # New database handling object
```

```
## dctr | New object

db$.drv # Empty driver

## NULL

db$.con # Empty connection

## NULL

db$.res # Empty result

## NULL
```

DBIDriver-class:

```
db$setupSQLite() # Setup SQLite database

## Dctr | Driver load RSQLite

db$.drv # 'DBI' DBIDriver-class

## <SQLiteDriver>

db$getInfoDrv() # 'rocker' class function

## Dctr | Driver info 2.2.9 (driver.version), 3.37.0 (client.version)

## $driver.version
## [1] '2.2.9'
##
## $client.version
## [1] '3.37.0'

DBI::dbGetInfo(db$.drv) # Direct usage of 'DBI' function on 'rocker' class

## $driver.version
## [1] '2.2.9'
##
## $client.version
## [1] '3.37.0'

RSQLite::dbGetInfo(db$.drv) # Direct usage of driver package, 'RSQLite', function on 'rocker' class

## $driver.version
## [1] '2.2.9'
##
## $client.version
## [1] '3.37.0'
```

DBIConnection-class:

```
db$connect() # Open connection
```

```
## DCtr | Database connected
```

```
db$.con # 'DBI' DBIConnection-class
```

```
## <SQLiteConnection>
##   Path: :memory:
##   Extensions: TRUE
```

```
db$getInfoCon() # 'rocker' class function
```

```
## DCtr | Connection info 3.37.0 (db.version), :memory: (dbname), NA (username), NA (host), NA (port)
```

```
## $db.version
## [1] "3.37.0"
##
## $dbname
## [1] ":memory:"
##
## $username
## [1] NA
##
## $host
## [1] NA
##
## $port
## [1] NA
```

```
DBI::dbGetInfo(db$.con) # Direct usage of 'DBI' function on 'rocker' class
```

```
## $db.version
## [1] "3.37.0"
##
## $dbname
## [1] ":memory:"
##
## $username
## [1] NA
##
## $host
## [1] NA
##
## $port
## [1] NA
```

```
RSQLite::dbGetInfo(db$.con) # Direct usage of driver package, 'RSQLite', function on 'rocker' class
```

```
## $db.version
## [1] "3.37.0"
##
## $dbname
## [1] ":memory:"
##
```

```
## $username
## [1] NA
##
## $host
## [1] NA
##
## $port
## [1] NA
```

Prepare table

```
db$writeTable("mtcars", mtcars) # Create table for testing
```

```
## DCtr | Write table mtcars columns mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, carb rows 32
```

DBIResult-class:

```
db$sendQuery("SELECT * FROM mtcars;") # Send query
```

```
## DCtR | Send query 21 characters
```

```
db$.res # 'DBI' DBIResult-class
```

```
## <SQLiteResult>
##   SQL  SELECT * FROM mtcars;
##   ROWS Fetched: 0 [incomplete]
##        Changed: 0
```

```
db$getInfoRes() # 'rocker' class function
```

```
## DCtR | Result info SELECT * FROM mtcars; (statement), 0 (row.count), 0 (rows.affected), FALSE (has.c
```

```
## $statement
## [1] "SELECT * FROM mtcars;"
##
## $row.count
## [1] 0
##
## $rows.affected
## [1] 0
##
## $has.completed
## [1] FALSE
```

```
DBI::dbGetInfo(db$.res) # Direct usage of 'DBI' function on 'rocker' class
```

```
## $statement
## [1] "SELECT * FROM mtcars;"
##
## $row.count
## [1] 0
##
## $rows.affected
```

```
## [1] 0
##
## $has.completed
## [1] FALSE
```

```
RSQLite::dbGetInfo(db$.res) # Direct usage of driver package, 'RSQLite', function on 'rocker' class
```

```
## $statement
## [1] "SELECT * FROM mtcars;"
##
## $row.count
## [1] 0
##
## $rows.affected
## [1] 0
##
## $has.completed
## [1] FALSE
```

Clean up

```
db$clearResult() # Clean up result
```

```
## DCtr | Clear result
```

```
db$.res # Empty result
```

```
## NULL
```

```
db$disconnect() # Close connection
```

```
## Dctr | Database disconnected
```

```
db$.con # Empty connection
```

```
## NULL
```

```
db$unloadDriver() # Reset database handling object
```

```
## dctr | Driver unload RSQLite
```

```
db$.drv # Empty driver
```

```
## NULL
```

DBI functions in *rocker*

Generally, *rocker* function names are related to *DBI* function names. In *rocker* functions, the leading **db** is removed.

In *DBI* most functions need to be supplied with a driver (drv), connection (conn) or result (res) object. In *rocker*, functions automatically access the corresponding objects (.drv, .con and .res) stored in the class.

DBI example

```
drv <- RSQLite::SQLite() # SQLite driver
DBI::dbCanConnect( # Test parameter
  drv = drv,
  dbname = ":memory:"
)

## [1] TRUE

con <- DBI::dbConnect( # Open connection
  drv = drv,
  dbname = ":memory:"
)
DBI::dbWriteTable(con, "mtcars", mtcars) # Create table for testing
res <- DBI::dbSendQuery(con, "SELECT * FROM mtcars;") # Send query
output <- DBI::dbFetch(res) # Fetch result
DBI::dbClearResult(res) # Clean up result
DBI::dbDisconnect(con) # Close connection
DBI::dbUnloadDriver(drv) # Unload driver
```

***rocker* example**

```
db <- rocker::newDB(verbose = FALSE) # New database handling object
db$setupDriver( # Setup SQLite database
  drv = RSQLite::SQLite(),
  dbname = ":memory:"
)
db$canConnect() # Test parameter

## [1] TRUE

db$connect() # Open connection
db$writeTable("mtcars", mtcars) # Create table for testing
db$sendQuery("SELECT * FROM mtcars;") # Send query
output <- db$fetch() # Fetch result
db$clearResult() # Clean up result
db$disconnect() # Close connection
db$unloadDriver() # Reset database handling object
```

List of functions:

rocker function	Corresponding *DBI* function	*DBI* object used	Comment
initialize()	*none*	*none*	
print()	*none*	*none*	
setupDriver()	*none*	*driver from appropriate package*	Usually, par
setupPostgreSQL()	*none*	*none*	RPostgres::I
setupMariaDB()	*none*	*none*	RMariaDB::
setupSQLite()	*none*	*none*	RSQLite::S
unloadDriver()	dbUnloadDriver()	driver	
canConnect()	dbCanConnect()	driver	Usually, par

connect()	dbConnect()	driver	Usually, par
disconnect()	dbDisconnect()	connection	
sendQuery()	dbSendQuery()	connection	
getQuery()	Is **not** using dbGetQuery(), but has the same function	connection	Especially, c
sendStatement()	dbSendStatement()	connection	
execute()	Is **not** using dbExecute(), but has the same function	connection	Especially, c
fetch()	dbFetch()	result	
hasCompleted()	dbHasCompleted()	result	
getRowsAffected()	dbGetRowsAffected()	result	
getRowCount()	dbGetRowCount()	result	
columnInfo()	dbColumnInfo()	result	
getStatement()	dbGetStatement()	result	
clearResult()	dbClearResult()	result	
begin()	dbBegin()	connection	
commit()	dbCommit()	connection	
rollback()	dbRollback()	connection	
getInfoDrv()	dbGetInfo()	driver	
getInfoCon()	dbGetInfo()	connection	
getInfoRes()	dbGetInfo()	result	
isValidDrv()	dbIsValid()	driver	
isValidCon()	dbIsValid()	connection	
isValidRes()	dbIsValid()	result	
createTable()	dbCreateTable()	connection	
appendTable()	dbAppendTable()	connection	
writeTable()	dbWriteTable	connection	
readTable()	dbReadTable	connection	
removeTable()	dbRemoveTable()	connection	
existsTable()	dbExistsTable()	connection	
listFields()	dbListFields()	connection	
listObjects()	dbListObjects()	connection	
listTables()	dbListTables()	connection	

Transaction

Setup database and a table with 32 rows.

```
db <- rocker::newDB() # New database handling object
```

```
## dctr | New object
```

```
db$setupSQLite() # Setup SQLite database
```

```
## Dctr | Driver load RSQLite
```

```
db$connect() # Open connection
```

```
## DCtr | Database connected
```

```
db$writeTable("mtcars", mtcars) # Create table for testing

## DCtr | Write table mtcars columns mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, carb rows 32

output <- db$getQuery("SELECT * FROM mtcars;") # Get query -> 32 rows

## DCtR | Send query 21 characters
## DCtR | Fetch rows all -> Received 32 rows, 11 columns, 4824 bytes
## DCtR | Rows fetched 32
## DCtR | Has completed yes
## DCtr | Clear result

db$transaction # Transaction indicator

## [1] FALSE
```

Starting with a table with 32 rows, begin transaction 1. Delete 15 rows and commit transaction. Operations results in a table with 17 rows.

```
db$begin() # Start transaction 1

## DCTr | Transaction begin

db$transaction # Transaction indicator

## [1] TRUE

AFFECTED <- db$execute("DELETE FROM mtcars WHERE gear = 3;") # Modify table -> 15 rows

## DCTR | Send statement 34 characters
## DCTR | Rows affected 15
## DCTr | Clear result

db$commit() # Commit transaction 1

## DCtr | Transaction commit

db$transaction # Transaction indicator

## [1] FALSE

output <- db$getQuery("SELECT * FROM mtcars;") # Get query -> 17 rows

## DCtR | Send query 21 characters
## DCtR | Fetch rows all -> Received 17 rows, 11 columns, 3504 bytes
## DCtR | Rows fetched 17
## DCtR | Has completed yes
## DCtr | Clear result
```

Starting with a table with 17 rows, begin transaction 2. Delete 5 rows and rollback transaction. Operations results in a table with 17 rows.

```
db$begin() # Start transaction 2

## DCTr | Transaction begin

db$transaction # Transaction indicator

## [1] TRUE

AFFECTED <- db$execute("DELETE FROM mtcars WHERE gear = 5;") # Modify table -> 5 rows

## DCTR | Send statement 34 characters
## DCTR | Rows affected 5
## DCTr | Clear result

output <- db$getQuery("SELECT * FROM mtcars;") # Get query -> 12 rows

## DCTR | Send query 21 characters
## DCTR | Fetch rows all -> Received 12 rows, 11 columns, 3416 bytes
## DCTR | Rows fetched 12
## DCTR | Has completed yes
## DCTr | Clear result

db$rollback() # Rollback transaction 2

## DCtr | Transaction rollback

db$transaction # Transaction indicator

## [1] FALSE

output <- db$getQuery("SELECT * FROM mtcars;") # Get query -> 17 rows

## DCtR | Send query 21 characters
## DCtR | Fetch rows all -> Received 17 rows, 11 columns, 3504 bytes
## DCtR | Rows fetched 17
## DCtR | Has completed yes
## DCtr | Clear result
```

Clean up

```
db$disconnect() # Close connection

## Dctr | Database disconnected

db$unloadDriver() # Reset database handling object

## dctr | Driver unload RSQLite
```

Usage of S3 and R6 functions

Although *rocker* is a *R6* class, functions can be also accesses in classical S3 way.

S3 example

```
library(rocker)
db <- newDB()

## dctr | New object

setupDriver(db, drv = RSQLite::SQLite(), dbname = ":memory:")

## Dctr | Driver load RSQLite

connect(db)

## DCtr | Database connected

writeTable(db, "mtcars", mtcars)

## DCtr | Write table mtcars columns mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, carb rows 32

sendQuery(db, "SELECT * FROM mtcars;")

## DCtR | Send query 21 characters

output <- fetch(db)

## DCtR | Fetch rows all -> Received 32 rows, 11 columns, 4824 bytes

clearResult(db)

## DCtr | Clear result

disconnect(db)

## Dctr | Database disconnected

unloadDriver(db)

## dctr | Driver unload RSQLite
```

R6 example

```
db <- rocker::newDB()

## dctr | New object
```

```
db$setupDriver(drv = RSQLite::SQLite(), dbname = ":memory:")

## Dctr | Driver load RSQLite

db$connect()

## DCtr | Database connected

db$writeTable("mtcars", mtcars)

## DCtr | Write table mtcars columns mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, carb rows 32

db$sendQuery("SELECT * FROM mtcars;")

## DCtR | Send query 21 characters

output <- db$fetch()

## DCtR | Fetch rows all -> Received 32 rows, 11 columns, 4824 bytes

db$clearResult()

## DCtr | Clear result

db$disconnect()

## Dctr | Database disconnected

db$unloadDriver()

## dctr | Driver unload RSQLite
```

See Also

Other rocker: newDB(), rocker-R6-class, rocker-S3-functions, rocker-package

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

rocker-S3-functions *'rocker' Database Interface R6 Class - S3 functions*

Description

R6 class interface for handling database connections using DBI package as backend. The class allows handling of connections to e.g. PostgreSQL, MariaDB and SQLite. Although rocker is a R6 class, functions can be also accesses in classical S3 way.

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Other rocker: newDB(), rocker-R6-class, rocker-README, rocker-package

Examples

```
# New database handling object
db <- rocker::newDB()
# Setup SQLite database
rocker::setupSQLite(db)
# Open connection
rocker::connect(db)
# Write table
rocker::writeTable(db, "mtcars", mtcars)
# Get query
output <- rocker::getQuery(db, "SELECT * FROM mtcars;")
# Close connection
rocker::disconnect(db)
# Reset database handling object
rocker::unloadDriver(db)
```

rollback *Rollback transaction.*

Description

Rollback transaction.

Usage

```
rollback(db, ...)
```

Arguments

db	rocker object
...	Optional, additional suitable parameters passed to DBI::dbRollback()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::begin(db)
rocker::sendStatement(db, "DELETE FROM mtcars WHERE gear = 3;")
rocker::clearResult(db)
rocker::rollback(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

---

sendQuery *Send SQL query to database.*

Description

Send SQL query to database.

Usage

```
sendQuery(db, statement, ...)
```

Arguments

db	rocker object
statement	SQL query (SELECT)
...	Optional, additional suitable parameters passed to DBI::dbSendQuery()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::sendQuery(db, "SELECT * FROM mtcars;")
output <- rocker::fetch(db)
rocker::clearResult(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

---

sendStatement *Send SQL statement to database.*

Description

Send SQL statement to database.

Usage

```
sendStatement(db, statement, ...)
```

Arguments

db	rocker object
statement	SQL statement (UPDATE, DELETE, INSERT INTO, ...)
...	Optional, additional suitable parameters passed to DBI::dbSendStatement()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::sendStatement(db, "DELETE FROM mtcars WHERE gear = 3;")
rocker::clearResult(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

`setupDriver` *Setup database driver and define connection parameters.*

Description

Setup database driver and define connection parameters.

Usage

```
setupDriver(db, drv, protect = c("password", "user"), ...)
```

Arguments

db	rocker object
drv	Driver object from suitable package e.g. RPostgres::Postgres(), RMariaDB::MariaDB() and RSQLite::SQLite()
protect	Parameters to be hidden
...	Suitable parameters passed to DBI::dbConnect() e.g. host, port, dbname, user and password

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupDriver(
  db,
  drv = RSQLite::SQLite(),
  dbname = ":memory:"
)
rocker::unloadDriver(db)
```

`setupMariaDB` *Setup database driver and define connection parameters for MariaDB using RMariaDB package. Wrapper for setupDriver() function.*

Description

Setup database driver and define connection parameters for MariaDB using RMariaDB package. Wrapper for setupDriver() function.

Usage

```
setupMariaDB(
  db,
  host = "127.0.0.1",
  port = "3306",
  dbname = "mydb",
  user = "root",
  password = "password",
  protect = c("password", "user"),
  ...
)
```

Arguments

db	rocker object
host	Host name or IP number
port	Port number
dbname	Database name

user	User name
password	Password
protect	Parameters to be hidden
...	Optional, additional suitable parameters passed to DBI::dbConnect()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupMariaDB(
  db,
  host = "127.0.0.1", port = "3306", dbname = "mydb",
  user = "root", password = "password"
)
rocker::unloadDriver(db)
```

---

setupPostgreSQL *Setup database driver and define connection parameters for PostgreSQL using RPostgres package. Wrapper for setupDriver() function.*

Description

Setup database driver and define connection parameters for PostgreSQL using RPostgres package. Wrapper for setupDriver() function.

Usage

```
setupPostgreSQL(
  db,
  host = "127.0.0.1",
  port = "5432",
  dbname = "mydb",
  user = "postgres",
  password = "password",
```

```
  protect = c("password", "user"),
  ...
)
```

Arguments

db	rocker object
host	Host name or IP number
port	Port number
dbname	Database name
user	User name
password	Password
protect	Parameters to be hidden
...	Optional, additional suitable parameters passed to DBI::dbConnect()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupSQLite(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupPostgreSQL(
  db,
  host = "127.0.0.1", port = "5432", dbname = "mydb",
  user = "postgres", password = "password"
)
rocker::unloadDriver(db)
```

setupSQLite *Setup database driver and define connection parameters for SQLite using RSQLite package. Wrapper for setupDriver() function.*

Description

Setup database driver and define connection parameters for SQLite using RSQLite package. Wrapper for setupDriver() function.

Usage

```
setupSQLite(db, dbname = ":memory:", protect = c("password", "user"), ...)
```

Arguments

db	rocker object
dbname	Database name
protect	Parameters to be hidden
...	Optional, additional suitable parameters passed to DBI::dbConnect()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), unloadDriver(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(
  db,
  dbname = ":memory:"
)
rocker::unloadDriver(db)
```

unloadDriver *Reset database driver and connection parameters.*

Description

Reset database driver and connection parameters.

Usage

```
unloadDriver(db, ...)
```

Arguments

db	rocker object
...	Optional, suitable parameters passed to DBI::dbUnloadDriver()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), validateCon(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::unloadDriver(db)
```

validateCon *Check if an earlier opened connection is still open.*

Description

Check if an earlier opened connection is still open.

Usage

```
validateCon(db, statement = NULL, onLostNull = FALSE, ...)
```

Arguments

db	rocker object
statement	Optional SQL statement. If not set default validateQuery will be used.
onLostNull	TRUE or FALSE. If connection lost, set .con to NULL
...	Not used yet

Value

TRUE or FALSE

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), writeTable()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::validateCon(db)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

---

writeTable *Write data to table.*

Description

Write data to table.

Usage

```
writeTable(db, name, value, ...)
```

Arguments

db	rocker object
name	Table name
value	Values data.frame
...	Optional, additional suitable parameters passed to DBI::dbWriteTable()

Value

Invisible self

See Also

Other rocker-S3-functions: appendTable(), begin(), canConnect(), clearResult(), columnInfo(), commit(), connect(), createTable(), disconnect(), execute(), existsTable(), fetch(), getInfoCon(), getInfoDrv(), getInfoRes(), getQuery(), getRowCount(), getRowsAffected(), getStatement(), hasCompleted(), isValidCon(), isValidDrv(), isValidRes(), listFields(), listObjects(), listTables(), readTable(), removeTable(), rocker-README, rocker-S3-functions, rocker-package, rollback(), sendQuery(), sendStatement(), setupDriver(), setupMariaDB(), setupPostgreSQL(), setupSQLite(), unloadDriver(), validateCon()

Examples

```
db <- rocker::newDB()
rocker::setupSQLite(db)
rocker::connect(db)
rocker::writeTable(db, "mtcars", mtcars)
rocker::disconnect(db)
rocker::unloadDriver(db)
```

Index

∗ **rocker-S3-functions**
appendTable, 2
begin, 3
canConnect, 4
clearResult, 5
columnInfo, 6
commit, 7
connect, 8
createTable, 9
disconnect, 10
execute, 11
existsTable, 12
fetch, 13
getInfoCon, 14
getInfoDrv, 15
getInfoRes, 16
getQuery, 17
getRowCount, 18
getRowsAffected, 19
getStatement, 20
hasCompleted, 21
isValidCon, 22
isValidDrv, 23
isValidRes, 24
listFields, 25
listObjects, 26
listTables, 27
readTable, 28
removeTable, 29
rocker-README, 56
rocker-S3-functions, 78
rollback, 78
sendQuery, 79
sendStatement, 80
setupDriver, 81
setupMariaDB, 82
setupPostgreSQL, 83
setupSQLite, 85
unloadDriver, 86
validateCon, 86
writeTable, 87
∗ **rocker**
newDB, 28
rocker-R6-class, 30
rocker-README, 56
rocker-S3-functions, 78

appendTable, 2, *4–27*, *29, 30*, *77–88*

begin, *3*, 3, *5–27*, *29, 30*, *77–88*

canConnect, *3, 4*, 4, *5–27*, *29, 30*, *77–88*
clearResult, *3–5*, 5, *6–27*, *29, 30*, *77–88*
columnInfo, *3–5*, 6, *7–27*, *29, 30*, *77–88*
commit, *3–6*, 7, *8–27*, *29, 30*, *77–88*
connect, *3–7*, 8, *9–27*, *29, 30*, *77–88*
createTable, *3–8*, 9, *10–27*, *29, 30*, *77–88*

DBI, *28*, *30*, *78*
DBI::dbAppendTable(), *3*, *44*
DBI::dbBegin(), *4*, *40*
DBI::dbCanConnect(), *4*, *34*
DBI::dbClearResult(), *5*, *39*
DBI::dbColumnInfo(), *6*, *38*
DBI::dbCommit(), *7*, *40*
DBI::dbConnect(), *8*, *32–35*, *81*, *83–85*
DBI::dbCreateTable(), *9*, *43*
DBI::dbDisconnect(), *10*, *35*
DBI::dbExistsTable(), *12*, *45*
DBI::dbFetch(), *13*, *37*
DBI::dbGetInfo(), *14–16*, *41*
DBI::dbGetRowCount(), *18*, *38*
DBI::dbGetRowsAffected(), *19*, *38*
DBI::dbGetStatement(), *20*, *39*
DBI::dbHasCompleted(), *21*, *37*
DBI::DBIConnection-class, *30*
DBI::DBIDriver-class, *30*
DBI::DBIResult-class, *30*
DBI::dbIsValid(), *22–24*, *42, 43*

DBI::dbListFields(), *25*, *46*
DBI::dbListObjects(), *26*, *46*
DBI::dbListTables(), *27*, *46*
DBI::dbReadTable(), *28*, *44*
DBI::dbRemoveTable(), *29*, *45*
DBI::dbRollback(), *40*, *79*
DBI::dbSendQuery(), *17*, *35, 36*, *79*
DBI::dbSendStatement(), *11*, *36*, *80*
DBI::dbUnloadDriver(), *34*, *86*
DBI::dbWriteTable(), *44*, *87*
disconnect, *3–9*, 10, *11–27*, *29, 30*, *77–88*

execute, *3–10*, 11, *12–27*, *29, 30*, *77–88*
existsTable, *3–11*, 12, *13–27*, *29, 30*, *77–88*

fetch, *3–12*, 13, *14–27*, *29, 30*, *77–88*

getInfoCon, *3–13*, 14, *15–27*, *29, 30*, *77–88*
getInfoDrv, *3–14*, 15, *16–27*, *29, 30*, *77–88*
getInfoRes, *3–15*, 16, *17–27*, *29, 30*, *77–88*
getQuery, *3–16*, 17, *18–27*, *29, 30*, *77–88*
getRowCount, *3–17*, 18, *19–27*, *29, 30*, *77–88*
getRowsAffected, *3–18*, 19, *20–27*, *29, 30*, *77–88*
getStatement, *3–19*, 20, *21–27*, *29, 30*, *77–88*

hasCompleted, *3–20*, 21, *22–27*, *29, 30*, *77–88*

isValidCon, *3–21*, 22, *23–27*, *29, 30*, *77–88*
isValidDrv, *3–22*, 23, *24–27*, *29, 30*, *77–88*
isValidRes, *3–23*, 24, *25–27*, *29, 30*, *77–88*

listFields, *3–24*, 25, *26, 27*, *29, 30*, *77–88*
listObjects, *3–25*, 26, *27*, *29, 30*, *77–88*
listTables, *3–26*, 27, *29, 30*, *77–88*

newDB, 28, *46*, *77, 78*

R6, *28*, *30*, *78*
R6Class, *30*
readTable, *3–27*, 28, *30*, *77–88*
removeTable, *3–27*, *29*, 29, *77–88*
RMariaDB, *33*, *82*
RMariaDB::MariaDB(), *32*, *81*
rocker, *28*
rocker *(*rocker-R6-class*)*, 30
rocker-R6-class, 30
rocker-README, 56
rocker-S3-functions, 78
rollback, *3–27*, *29, 30*, *77, 78*, 78, *80–88*
RPostgres, *32*, *83*
RPostgres::Postgres(), *32*, *81*
RSQLite, *34*, *85*
RSQLite::SQLite(), *32*, *81*

sendQuery, *3–27*, *29, 30*, *77–79*, 79, *81–88*
sendStatement, *3–27*, *29, 30*, *77–80*, 80, *82–88*
setupDriver, *3–27*, *29, 30*, *77–81*, 81, *83–88*
setupMariaDB, *3–27*, *29, 30*, *77–82*, 82, *84–88*
setupPostgreSQL, *3–27*, *29, 30*, *77–83*, 83, *85–88*
setupSQLite, *3–27*, *29, 30*, *77–84*, 85, *86–88*

unloadDriver, *3–27*, *29, 30*, *77–85*, 86, *87, 88*

validateCon, *3–27*, *29, 30*, *77–86*, 86, *88*

writeTable, *3–27*, *29, 30*, *77–87*, 87