Package 'RMongo'

July 2, 2014

Type Package

Title MongoDB Client for R

Version 0.0.25

Date 2013-08-28

Author Tommy Chheng

Maintainer Tommy Chheng <tommy.chheng@gmail.com>

Description MongoDB Database interface for R. The interface is provided via Java calls to the mongo-java-driver.

License GPL-3

LazyLoad yes

Depends R(>= 2.14.1), rJava, methods

Suggests RUnit

SystemRequirements Java (>= 1.6), MongoDB (>= 1.6)

URL http://github.com/tc/RMongo

NeedsCompilation no

Repository CRAN

Date/Publication 2013-09-08 08:11:11

R topics documented:

RMongo-package *MongoDB client for R.*

Description

The functions in this package allow you make queries with a MongoDB database from R.

Overview

A typical workflow using RMongo:

1. Connect to a mongodb database:

```
mongo <- mongoDbConnect("test", "localhost", 27017)
```

2. Insert a document into a collection: This will insert a document of '"foo":"bar"' into the "test_data" collection.

```
output <- dbInsertDocument(mongo, "test_data", '{"foo": "bar"}')
```

3. Query a MongoDB collection: This will query for documents which have a "foo" attribute with the value of "bar". The return value will be a list of rjson objects or a converted data.frame depending on the format parameter. NOTE: nested documents and arrays may not work properly if you specify a data.frame.

```
output <- dbGetQuery(mongo, "test_data",
                             '{"foo": "bar"}')
```

or if you want only a particular set of keys:

```
output <- dbGetQueryForKeys(mongo, "test_data",
                             '{"foo": "bar"}', '{"foo": 1}')
```

4. Close the connection:

```
dbDisconnect(mongo)
```

5. Large result sets: Set your java heap space to a large number to support loading more results into memory.

```
options( java.parameters = "-Xmx2g" )
```

Author(s)

Tommy Chheng <tommy.chheng@gmail.com> http://tommy.chheng.com

---

dbAggregate-methods *Performing a MongoDB aggregate query*

Description

Send a set of aggregate pipeline commands to mongodb. The output is a data.frame object and will work properly only if the mongoDB collection contains primitive data types. It may not work properly if there are any embedded documents or arrays.

Usage

```
dbAggregate(rmongo.object, collection, query)
```

Arguments

rmongo.object	The RMongo object.
collection	The name of the collection the query is being performed upon.
query	A vector of pipeline command in JSON format. See http://docs.mongodb.org/manual/reference/aggregatic for more information on the MongoDB aggregation framework.

See Also

mongoDbConnect

Examples

```
mongo <- mongoDbConnect("test")
# insert two records
dbInsertDocument(mongo, "test_data", '{"foo": "bar", "size": 5 }')
dbInsertDocument(mongo, "test_data", '{"foo": "nl", "size": 10 }')
output <- dbAggregate(mongo, "test_data", c(' { "$project" : { "baz" : "$foo" } } ',
                                            ' { "$group" : { "_id" : "$baz" } } ',
                                            ' { "$match" : { "_id" : "bar" } } '))
print(output)
```

dbAuthenticate-methods *Authenticate with a username/password.*

Description

This is an optional authenticate method.

Usage

```
dbAuthenticate(rmongo.object, username, password)
```

Arguments

rmongo.object	RMongo object containing the database connection information.
username	Username
password	Password

See Also

dbAuthenticate

Examples

```
mongo <- mongoDbConnect("test")
username = ""
password = ""
authenticated <- dbAuthenticate(mongo, username, password)
dbDisconnect(mongo)
```

dbDisconnect-methods *Disconnect from the MongoDB database*

Description

This will close the connect from the MongoDB database.

Usage

```
dbDisconnect(rmongo.object)
```

Arguments

rmongo.object	RMongo object containing the database connection information.

See Also

mongoDbConnect

Examples

```
mongo <- mongoDbConnect("test")
dbDisconnect(mongo)
```

---

dbGetDistinct-methods *Performing a distinct MongoDB query*

Description

Send a json query to mongodb. The output is a data.frame object and will work properly only if the mongoDB collection contains primitive data types. It may not work properly if there are any embedded documents or arrays.

Usage

```
dbGetDistinct(rmongo.object, collection, key, query)
```

Arguments

rmongo.object	The RMongo object.
collection	The name of the collection the query is being performed upon.
key	A JSON string query. See http://www.mongodb.org/display/DOCS/Advanced+Queries for more information on the MongoDB query language.
query	A JSON string query. See http://www.mongodb.org/display/DOCS/Advanced+Queries for more information on the MongoDB query language.

See Also

mongoDbConnect

Examples

```
mongo <- mongoDbConnect("test")
output <- dbInsertDocument(mongo, "test_data", '{"foo": "bar"}')
output <- dbInsertDocument(mongo, "test_data", '{"foo": "bar"}')
output <- dbGetDistinct(mongo, 'test_data', '{"foo": "bar"}')
print(output)
```

dbGetQuery-methods *Performing a MongoDB query*

Description

Send a json query to mongodb. The output is a data.frame object and will work properly only if the mongoDB collection contains primitive data types. It may not work properly if there are any embedded documents or arrays.

Usage

```
dbGetQuery(rmongo.object, collection, query, skip=0, limit=1000)
```

Arguments

rmongo.object	The RMongo object.
collection	The name of the collection the query is being performed upon.
query	A JSON string query. See http://www.mongodb.org/display/DOCS/Advanced+Queries for more information on the MongoDB query language.
skip	Offset the resultset. Can be used with limit to perform pagination.
limit	Limits the resultset size.

See Also

mongoDbConnect

Examples

```
mongo <- mongoDbConnect("test")
output <- dbInsertDocument(mongo, "test_data", '{"foo": "bar"}')
output <- dbGetQuery(mongo, 'test_data', '{"foo": "bar"}')
print(output)

output <- dbInsertDocument(mongo, "test_data", '{"foo": "bar with spaces"}')
output <- dbGetQuery(mongo, 'test_data', '{"foo": { "$regex": "bar*", "$options": "i"} }')
```

dbGetQueryForKeys-methods

Performing a MongoDB query and only return back certain keys

Description

Send a json query to mongodb. The output is a data.frame object and will work properly only if the mongoDB collection contains primitive data types. It may not work properly if there are any embedded documents or arrays.

Usage

```
dbGetQueryForKeys(rmongo.object, collection, query, keys, skip=0, limit=1000)
```

Arguments

rmongo.object	The RMongo object.
collection	The name of the collection the query is being performed upon.
query	A JSON string query. See http://www.mongodb.org/display/DOCS/Advanced+Queries for more information on the MongoDB query language.
keys	A JSON string query. See http://www.mongodb.org/display/DOCS/Advanced+Queries for more information on the MongoDB query language.
skip	Offset the resultset. Can be used with limit to perform pagination.
limit	Limits the resultset size.

See Also

mongoDbConnect

Examples

```
mongo <- mongoDbConnect("test")
output <- dbInsertDocument(mongo, "test_data", '{"foo": "bar"}')
output <- dbGetQueryForKeys(mongo, 'test_data', '{"foo": "bar"}', '{"foo":1}')
print(output)
```

---

dbInsertDocument-methods

Insert a document into a MongoDB collection

Description

Insert a json character object into the mongodb collection.

Usage

```
dbInsertDocument(rmongo.object, collection, doc)
```

Arguments

rmongo.object	The RMongo object.
collection	The name of the collection the document is being inserted into.
doc	A JSON string document.

See Also

dbGetQuery-methods

Examples

```
mongo <- mongoDbConnect("test")
output <- dbInsertDocument(mongo, "test_data", '{"foo": "bar"}')
dbDisconnect(mongo)
```

---

dbRemoveQuery-methods *Performing a delete MongoDB query*

Description

Send a json query to mongodb.

Usage

```
dbRemoveQuery(rmongo.object, collection, query)
```

Arguments

rmongo.object	The RMongo object.
collection	The name of the collection the query is being performed upon.
query	A JSON string query. See http://www.mongodb.org/display/DOCS/Advanced+Queries for more information on the MongoDB query language.

See Also

mongoDbConnect

Examples

```
mongo <- mongoDbConnect("test")
output <- dbInsertDocument(mongo, "test_data", '{"foo": "bar"}')
output <- dbRemoveQuery(mongo, 'test_data', '{"foo": "bar"}')
print(output)
```

dbSetWriteConcern-methods

Set write concern.

Description

This is an optional method to set the write concern for a given Mongo DB connection. By default the Mongo DB library does not check if a write (insert, update, remove) actually succeed.

Usage

```
dbSetWriteConcern(rmongo.object, w, wtimeout, fsync, j)
```

Arguments

rmongo.object	RMongo object containing the database connection information.
w	Number of write to aknowledge. -1 Don't even report network errors. 0 Don't wait for acknowledgement from the server. 1 Wait for acknowledgement but don't wait for secondaries to replicate. 2+ Wait for one or more secondaries to also acknowledge.
wtimeout	How long to wait for slaves before failing. 0 indefinite. Greater than 0 milliseconds to wait.
j	Wait for group commit to journal.
fsync	Force fsync to disk.

Examples

```
mongo <- mongoDbConnect("test")
dbSetWriteConcern(mongo, 1, 0, FALSE, FALSE)
dbDisconnect(mongo)
```

dbShowCollections-methods

Shows a list of collections

Description

Shows a list of collections in the specified rmongo.object database.

Usage

```
dbShowCollections(rmongo.object)
```

Arguments

rmongo.object	The RMongo object.

See Also

dbShowCollections-methods

Examples

```
mongo <- mongoDbConnect("test")
dbShowCollections(mongo)
dbDisconnect(mongo)
```

---

mongoDbConnect *Connecting to a MongoDB database*

Description

Creates a RMongo object to use with querying and inserting methods.

Usage

```
mongoDbConnect(dbName, host="127.0.0.1", port=27017)
```

Arguments

dbName	Database name.
host	Host name. Default to localhost: 127.0.0.1. You can specify a remote host. optional
port	Port number. Default to mongodb's default port 27017. optional

Examples

```
mongo <- mongoDbConnect("test", "127.0.0.1", 27017)
```

mongoDbReplicaSetConnect *Connecting to a MongoDB Replica Set*

Description

Connects to a Mongo DB Replica Set and creates a RMongo object to use with querying and inserting methods.

Usage

```
mongoDbReplicaSetConnect(dbName, hosts="127.0.0.1:27017")
```

Arguments

dbName	Database name.
hosts	Comma separated list of hosts member of the replica set. Each entry in the list must have the format <hostname>[:<port>] . If <port> is omitted, 27017 is used optional

Examples

```
mongo <- mongoDbReplicaSetConnect("test", "localhost:27017")
```

RMongo-class *Class RMongo*

Description

Contains the mongodb connection information. See mongoDbConnect for creating a RMongo object.

Methods

dbAuthenticate

dbGetQuery

dbGetQueryForKeys

dbInsertDocument

dbRemoveQuery

dbGetDistinct

dbDisconnect

See Also

mongoDbConnect

Index