Package ‘RcppRedis’

April 10, 2022

Type Package

Title 'Rcpp' Bindings for 'Redis' using the 'hiredis' Library

Version 0.2.1

Date 2022-04-09

Author Dirk Eddelbuettel and Bryan W. Lewis

Maintainer Dirk Eddelbuettel <edd@debian.org>

Description Connection to the 'Redis' key/value store using the
C-language client library 'hiredis' (included as a fallback) with
'MsgPack' encoding provided via 'RcppMsgPack' headers. It now also
includes the pub/sub functions from the 'redis' package.

SystemRequirements An available hiredis library (eg via package
libhiredis-dev on Debian/Ubuntu, hiredis-devel on
Fedora/RedHat, or directly from source from
<https://github.com/redis/hiredis>) can be used but version
1.0.2 is also included and built on demand if needed. The
optional (but suggested) 'redis' package can be installed from
the 'ghrr' 'drat' repo for additional illustrations and tests.

URL https://github.com/eddelbuettel/rcppredis,
https://dirk.eddelbuettel.com/code/rcpp.redis.html

BugReports https://github.com/eddelbuettel/rcppredis/issues

License GPL (>= 2)

Imports methods, Rcpp (>= 0.11.0), RApiSerialize

LinkingTo Rcpp, RApiSerialize

Suggests RcppMsgPack, redis, tinytest

Additional_repositories https://ghrr.github.io/drat

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-04-09 23:22:32 UTC
Description

The Redis module is created using Rcpp modules and wraps a minimal class Redis around a Redis connection context object which permits bi-directional communication with a Redis in-memory database.

New instances can be created using either a default constructor (using localhost and the default port) and either host and port, or just the host.

Currently, the module has just one worker command which sends a string to the Redis instance and returns a string.

The helper functions `serializeToChar()` and `unserializeFromChar` convert R objects to/from a character representation (and internalize the conversion from raw to char representation at the compiled level). The helper functions `serializeToRaw()` and `unserializeFromRaw` convert R objects to/from raw vectors.

Details

Please consult the Redis documentation for details on the available commands. See the Rcpp-modules vignette for details on Rcpp modules.

Author(s)

Dirk Eddelbuettel <edd@debian.org>

Description

Listen for messages on subscribed Redis message channels.

Usage

    redisMonitorChannels(context, type=c("rdata", "raw", "string"))
redisMonitorChannels

Arguments

context A valid Redis context (see example).
type The expected message value type.

Details

(From the Redis.io documentation): implement the Publish/Subscribe messaging paradigm where (citing Wikipedia) senders (publishers) are not programmed to send their messages to specific receivers (subscribers). Rather, published messages are characterized into channels, without knowledge of what (if any) subscribers there may be. Subscribers express interest in one or more channels, and only receive messages that are of interest, without knowledge of what (if any) publishers there are.

The redisMonitorChannels function may be called repeatedly in an event loop to service messages on all subscribed channels. When a message is received, the redisMonitorChannels function will attempt to evaluate a callback function with same name as the channel, with the message as its single argument. If no such function can be found, the message is returned. See the help page for redisGetResponse for a description of the message format.

WARNING: The redisMonitorChannels function blocks indefinitely until a message is received. Use the lower-level listen context method to simply poll channels for messages without evaluating function callbacks.

Value

The result of an evaluated function callback message, or if no matching callback exists, the message.

Author(s)

B. W. Lewis

References

http://redis.io/commands

Examples

## Not run:
x <- new(Redis)
y <- new(Redis)

# Define a callback function to process messages from channel 1:
channel1 <- function(x) {
  cat("Message received from channel 1: ",x,"\n")
}

# Define a callback function to process messages from channel 2:
channel2 <- function(x) {
  cat("Message received from channel 2: ",x,"\n")
}

# Subscribe to the channels...
x$subscribe(c("channel1", "channel2"))
y$publish("channel2", pi)

redisMonitorChannels(x)

# Unsubscribe
x$unsubscribe(c("channel1", "channel2"))

## End(Not run)
Index

* **package**
  RcppRedis, 2

Rcpp_Redis (RcppRedis), 2
Rcpp_Redis-class (RcppRedis), 2
RcppRedis, 2
Redis (RcppRedis), 2
redisMonitorChannels, 2

serializeToChar (RcppRedis), 2
serializeToRaw (RcppRedis), 2

unserializeFromChar (RcppRedis), 2
unserializeFromRaw (RcppRedis), 2