Package ‘easydb’

March 6, 2023

Type Package

Title Easily Connect to Common Types of Databases

Version 1.1.0

Description A unified interface for connecting to databases ('SQLite', 'MySQL', 'PostgreSQL').
Just provide the database name and the package will ask you questions
to help you configure the connection and setup your credentials. Once
database configuration and connection has been set up once, you won't
have to do it ever again.

License MIT + file LICENSE

URL https://github.com/selkamand/easydb,
https://selkamand.github.io/easydb/

BugReports https://github.com/selkamand/easydb/issues

Imports askpass, assertthat, cli, DBI, keyring, rlang, utils, yaml

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

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Sam El-Kamand [aut, cre, cph] (<https://orcid.org/0000-0003-2270-8088>)

Maintainer Sam El-Kamand <sam.elkamand@gmail.com>

Repository CRAN

Date/Publication 2023-03-06 10:20:07 UTC

R topics documented:

`assert_config_file_appropriate`
Check config filepath is appropriate

Description

Ensures file is hidden, in a folder that exists, and that has read and write permissions. The file itself doesn't already have to exist

Usage

```
assert_config_file_appropriate(file)
```

Arguments

`file`	path to config file (string)

Value

invisibly returns TRUE

`easydb_available_databases`
Available Databases

Description

List databases with configuration details stored in a configuration file.

Usage

```
easydb_available_databases(config_file)
```

Arguments

config_file	path to yaml file containing configuration information about databases (port, host, etc.) (string)

Value

database names (character vector)

Examples

```
path_to_config = tempfile()
easydb_available_databases(path_to_config)
```

---

easydb_connect *Easy Database Connection*

Description

Easily connect to a database. When connecting to a database for the first time, you will be prompted for connection details. Configuration details will be stored in the user-specified 'configuration file'. The next time you run the same command, config details will be automatically retreived from the config file. Any usernames and passwords are stored in your system credential manager (not on-disk)

Usage

```
easydb_connect(dbname, config_file, from_scratch = FALSE)
```

Arguments

dbname	name of database (string)
config_file	path to yaml file containing configuration information about databases (port, host, etc.) (string)
from_scratch	should we delete any current config / credentials for databases with the supplied name and start again? (logical)

Value

connection to database (connection)

Examples

```
if(interactive()) {

  # Choose config file path
  # Do NOT use tempfile in practice.
  # Instead, choose a fixed location such as '~/.easydb'
  config <- tempfile('.example_config')

  # Initialise config file
  easydb_init(config)

  # Connect to SQLite database
  path_to_db <- system.file(package = 'easydb', 'testdbs/mtcars.sqlite')
  con <- easydb_connect(dbname = path_to_db, config_file = config)

  # Disconnect from database when finished
  easydb_disconnect(con)
}
```

`easydb_disconnect` *Disconnect from database*

Description

Simple wrapper around [DBI::dbDisconnect()]

Usage

```
easydb_disconnect(connection)
```

Arguments

`connection`	coneection generated by [easydb_connect]

Value

Invisibly returns TRUE

Examples

```
if(interactive()) {

  # Choose config file path
  # Do NOT use tempfile in practice.
  # Instead, choose a fixed location such as '~/.easydb'
  config <- tempfile('.example_config')

  # Initialise config file
  easydb_init(config)
```

```
    # Connect to SQLite database
    path_to_db <- system.file(package = 'easydb', 'testdbs/mtcars.sqlite')
    con <- easydb_connect(dbname = path_to_db, config_file = config)

    # Disconnect from database when finished
    easydb_disconnect(con)
  }
```

`easydb_init` *Initialise Configuration Store*

Description

Create a file to store your database configurations in.

Usage

```
easydb_init(config_file)
```

Arguments

`config_file`	path to create a new config file to store database configurations

Value

invisibly returns path to config file

Examples

```
# Choose config file path
# Do NOT use tempfile in practice.
# Choose a fixed location such as '~/.easydb'
config <- tempfile('.example_config')

# Initialise Configuration File
easydb_init(config)
```

`supported_drivers` *List supported drivers*

Description

List supported drivers

Usage

```
supported_drivers()
```

Value

returns a vector of supported database drivers (character). Names relate to the R packages containing the relevant driver function

`utils_database_already_in_yaml` *Database Utils*

Description

Check if a yaml contains an entry describing a given database

Usage

```
utils_database_already_in_yaml(dbname, file)
```

Arguments

`dbname`	database name (character)
`file`	file (string)

Value

true if yaml entry for dbname found, otherwise false (logical)

`utils_database_get_or_set_config`
Interactive getting/setting database configurations

Description

Interactive getting/setting database configurations

Usage

```
utils_database_get_or_set_config(dbname, file)
```

Arguments

`dbname`	nane of database (string)
`file`	yaml file to store database configuration information

Value

list describing database configuration

`utils_database_get_or_set_creds`
Create/Retrieve database credentials

Description

Retrieves credentials (username and password) from os credential store using keyring package.

Usage

```
utils_database_get_or_set_creds(dbname)
```

Arguments

`dbname`	Name of database to store/retrieve credentials for (string)

Value

Invisibly returns list of username and password

Examples

```
## Not run:
creds <- easydb::util_get_database_creds(
  service = "R-keyring-test-service",
  username = "donaldduck"
)
creds$username
creds$password

## End(Not run)
```

`utils_database_read_yaml` *Read the database config*

Description

Read the database config

Usage

```
utils_database_read_yaml(dbname, file)
```

Arguments

`dbname`	name of database whose config you want to read
`file`	path to config yaml

Value

list with config

`utils_database_remove_entry_from_yaml` *Delete database config entries from yaml*

Description

Delete database config entries from yaml

Usage

```
utils_database_remove_entry_from_yaml(dbnames, file)
```

Arguments

`dbnames`	names of databases to delete from yaml (character)
`file`	path to config yaml (string)

Value

Run for its side effects

`utils_database_write_yaml` *Write database yaml*

Description

Writes database config details into a yaml. Usernames and passwords are never saved in this yaml file

Usage

```
utils_database_write_yaml(
  dbname,
  driver,
  creds_required = FALSE,
  port = NULL,
  host = NULL,
  ssl_cert = NULL,
  ssl_key = NULL,
  ssl_ca = NULL,
  file,
  append = TRUE
)
```

Arguments

`dbname`	name of database
`driver`	name of driver
`creds_required`	are credentials (username/password) required for this database (flag)
`port`	database port
`host`	database host
`ssl_cert`	path to ssl certificate (string)
`ssl_key`	path to ssl key (string)
`ssl_ca`	path to ssl CA certificate (string)
`file`	where config file should be located (will be produced if it doesn't already exist)
`append`	should config file be appended or overwritten? Defualts to append. Don't change unless you know what you're doing

Value

path to config yaml containing the new database info (string)

---

utils_user_input_retrieve_free_text

get user input

Description

get user input

Usage

```
utils_user_input_retrieve_free_text(
  message,
  default = NULL,
  type = c("string", "number")
)
```

Arguments

`message`	message - tell user what information to input
`default`	if user enters no data, this value will be returned (NULL)
`type`	return type

Value

user input formatted according to type. If user does not enter anything will return value of `default`

Index