Package ‘nodbi’

November 21, 2021

Title 'NoSQL' Database Connector

Description Simplified document database access and manipulation, providing a common API across supported 'NoSQL' databases 'Elasticsearch', 'CouchDB', 'MongoDB' as well as 'SQLite/JSON1'.

Version 0.5.0

License MIT + file LICENSE

LazyData true

URL https://docs.ropensci.org/nodbi/, https://github.com/ropensci/nodbi

BugReports https://github.com/ropensci/nodbi/issues

Depends R (>= 2.10)

Encoding UTF-8

Language en-US

Imports stringi, jsonlite, jsonify, uuid, jqr, sofa (>= 0.3.0), elastic (>= 1.0.0), mongolite (>= 1.6), RSQLite (>= 2.2.4), DBI

Suggests testthat

RoxygenNote 7.1.2

X-schema.org-applicationCategory Databases

X-schema.org-keywords database, MongoDB, Elasticsearch, CouchDB, SQLite, NoSQL, JSON, documents

X-schema.org-isPartOf https://ropensci.org

NeedsCompilation no

Author Ralf Herold [aut, cre] (<https://orcid.org/0000-0002-8148-6748>), Scott Chamberlain [aut] (<https://orcid.org/0000-0003-1444-9135>), Rich FitzJohn [aut].

Jeroen Ooms [aut]

Maintainer Ralf Herold <ralf.herold@mailbox.org>

Repository CRAN

Date/Publication 2021-11-21 15:00:02 UTC
nodbi-package

Description

Simplified document database access and manipulation, providing a common API across supported 'NoSQL' databases 'Elasticsearch', 'CouchDB', 'MongoDB' as well as 'SQLite/JSON1'.

Author(s)

Scott Chamberlain <sckott@protonmail.com>
Rich FitzJohn <rich.fitzjohn@gmail.com>
Jeroen Ooms <jeroen.ooms@stat.ucla.edu>
Ralf Herold <ralf.herold@mailbox.org>
contacts

contacts JSON data set

Description
contacts JSON data set

Usage
contacts

Format
A JSON string with ragged, nested contact details

diamonds

diamonds data set

Description
diamonds data set

Format
A data frame with 53940 rows and 10 variables:

- price price in US dollars ($326-$18,823)
- carat weight of the diamond (0.2-5.01)
- cut quality of the cut (Fair, Good, Very Good, Premium, Ideal)
- color diamond colour, from J (worst) to D (best)
- clarity a measurement of how clear the diamond is (I1 (worst), SI1, SI2, VS1, VS2, VVS1, VVS2, IF (best))
- x length in mm (0-10.74)
- y width in mm (0-58.9)
- z depth in mm (0-31.8)
- depth total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43-79)
- table width of top of diamond relative to widest point (43-95)

Source
from ggplot2
Description

A message is emitted if the container key already exists.

Usage

docdb_create(src, key, value, ...)

Arguments

src Source object, result of call to any of functions src_mongo(), src_sqlite(), src_elastic() or src_couchdb()
key (character) A key as name of the container (corresponds to parameter collection for MongoDB, dbname for CouchDB, index for Elasticsearch and to a table name for SQLite)
value The data to be created in the database: a single data.frame, a JSON string or a list
... Passed to functions:
• CouchDB: sofa::db_bulk_create()
• Elasticsearch: elastic::docs_bulk
• MongoDB: mongolite::mongo()
• RSQLite: ignored

Value

(integer) Number of successfully created documents

Identifiers

Any _id’s in value will be used as _id’s and primary index in the database. If there are no _id’s in value, row names (if any exist) will be used as _id’s, or random _id’s will be created (using uuid::UUIDgenerate() with use.time = TRUE).

A warning is emitted if a document(s) with _id’s already exist in value and that document in value is not newly created in the database; use docdb_update() to update such document(s).

Examples

```r
## Not run:
src <- src_sqlite()
docdb_create(src, key = "diamonds_small",
              value = as.data.frame(diamonds[1:3000L,]))
head(docdb_get(src, "diamonds_small"))
docdb_create(src, key = "contacts", value = contacts)
```
**docdb_delete**

```
docdb_get(src, "contacts")["friends"]

## End(Not run)
```

---

**Delete documents or container**

**Description**

Delete documents or container

**Usage**

```
docdb_delete(src, key, ...)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>src</td>
<td>Source object, result of call to any of functions <code>src_mongo()</code>, <code>src_sqlite()</code>, <code>src_elastic()</code> or <code>src_couchdb()</code></td>
</tr>
<tr>
<td>key</td>
<td>(character) A key as name of the container (corresponds to parameter collection for MongoDB, dbname for CouchDB, index for Elasticsearch and to a table name for SQLite)</td>
</tr>
<tr>
<td>...</td>
<td>optional query parameter with a JSON query as per <code>mongolite::mongo()</code> and as working in <code>docdb_query()</code> to identify documents to be deleted. The default is to delete the container key. Other parameters passed on to database backends.</td>
</tr>
</tbody>
</table>

**Value**

(logical) success of operation. Typically TRUE if document or collection existed and FALSE is document did not exist or collection did not exist or delete was not successful.

**Examples**

```
## Not run:
src <- src_sqlite()
docdb_create(src, "iris", iris)
docdb_delete(src, "iris", query = '{"Species": {"$regex": "a$"}}')
docdb_delete(src, "iris")

## End(Not run)
```
**docdb_exists**  
*Check if container exists in database*

**Description**
Check if container exists in database

**Usage**
```
docdb_exists(src, key, ...)
```

**Arguments**
- `src` Source object, result of call to any of functions `src_mongo()`, `src_sqlite()`, `src_elastic()` or `src_couchdb()`
- `key` (character) A key as name of the container (corresponds to parameter `collection` for MongoDB, `dbname` for CouchDB, `index` for Elasticsearch and to a table name for SQLite)
- `...` Passed to functions:
  - MongoDB: `find()` in `mongolite::mongo()`
  - RSQLite: `DBI::dbListTables()`
  - Elasticsearch: `elastic::index_exists()`
  - CouchDB: `sofa::db_info()`

**Value**
(logical) TRUE or FALSE to indicate existence of container key in database

**Examples**
```r
## Not run:
src <- src_sqlite()
docdb_exists(src, "nonexistingcontainer")
docdb_create(src, "mtcars", mtcars)
docdb_exists(src, "mtcars")
## End(Not run)
```
**Description**

Get all documents from container in database

**Usage**

docdb_get(src, key, limit = NULL, ...)

**Arguments**

- **src**
  Source object, result of call to any of functions `src_mongo()`, `src_sqlite()`, `src_elastic()` or `src_couchdb()`

- **key**
  (character) A key as name of the container (corresponds to parameter `collection` for MongoDB, `dbname` for CouchDB, `index` for Elasticsearch and to a table name for SQLite)

- **limit**
  (integer) Maximum number of documents to return (defaults to all for MongoDB, all for SQLite, 10,000 for Elasticsearch and all for CouchDB)

- **...**
  Passed on to functions:
  - MongoDB: `find()` in `mongolite::mongo()`
  - SQLite: ignored
  - Elasticsearch: `elastic::Search()`
  - CouchDB: `sofa::db_alldocs()`

**Value**

Document(s) in a data frame

**Examples**

```r
## Not run:
src <- src_sqlite()
docdb_create(src, "mtcars", mtcars)
docdb_get(src, "mtcars", limit = 10L)
## End(Not run)
```
**docdb_list**

*List containers in database*

**Description**

List containers in database

**Usage**

docdb_list(src, ...)

**Arguments**

- **src**
  - Source object, result of call to any of functions `src_mongo()`, `src_sqlite()`, `src_elastic()` or `src_couchdb()`

- **...**
  - Passed to functions:
    - CouchDB: `sofa::db_info()`
    - Elasticsearch: `elastic::aliases_get()`
    - MongoDB: ignored
    - RSQLite: ignored

**Value**

(vector) of names of containers that can be used as parameter key with other functions such as `docdb_create()`. Parameter key corresponds to collection for MongoDB, dbname for CouchDB, index for Elasticsearch and a table name for RSQLite)

**Examples**

```r
## Not run:
src <- src_sqlite()
docdb_create(src, "iris", iris)
docdb_list(src)
## End(Not run)
```

---

**docdb_query**

*Get documents with a filtering query*

**Description**

Get documents with a filtering query

**Usage**

docdb_query(src, key, query, ...)

Arguments

src
Source object, result of call to any of functions `src_mongo()`, `src_sqlite()`, `src_elastic()` or `src_couchdb()`

key
(character) A key as name of the container (corresponds to parameter `collection` for MongoDB, `dbname` for CouchDB, `index` for Elasticsearch and to a table name for SQLite)

query
(character) A JSON query string, see examples

... Optionally, `fields` a JSON string of fields to be returned from anywhere in the tree (dot paths notation).

Other parameters passed on to functions:
- MongoDB: `find()` in `mongolite::mongo()`
- RSQLite: for mangling into SQL query
- Elasticsearch: `elastic::Search()`
- CouchDB: `sofa::db_query()`

Value

Data frame with requested data, may have nested lists in columns

Examples

```
## Not run:
src <- src_sqlite()
docdb_create(src, "mtcars", mtcars)
docdb_query(src, "mtcars", query = '{"mpg":21}')
docdb_query(src, "mtcars", query = '{"mpg":21}', fields = '{"mpg":1, "cyl":1}')
docdb_query(src, "mtcars", query = '{"_id": {"$regex": "^.+0.*$"}}', fields = '{"gear": 1}')
# complex query, not supported for Elasticsearch and CouchDB backends at this time:
docdb_query(src, "mtcars", query = '{"$and": [{"mpg": {"$lte": 18}}, {"gear": {"$gt": 3}}]}')
```

## End(Not run)

---

**docdb_update**

*Update documents*

**Description**

Documents identified by the query are updated by patching their JSON with `value`. This is native with MongoDB and SQLite and is emulated for Elasticsearch and CouchDB using SQLite/JSON1.

**Usage**

```
docdb_update(src, key, value, query, ...)
```
Arguments

src  
Source object, result of call to any of functions src_mongo(), src_sqlite(), src_elastic() or src_couchdb()

key  
(character) A key as name of the container (corresponds to parameter collection for MongoDB, dbname for CouchDB, index for Elasticsearch and to a table name for SQLite)

value  
The data to be created in the database: a single data.frame, a JSON string or a list

query  
(character) A JSON query string, see examples

Passed on to functions:

- CouchDB: sofa::db_bulk_create()
- Elasticsearch: elastic::docs_bulk_update
- MongoDB: mongolite::mongo()
- RSQLite: ignored

Value

(int) Number of successfully updated documents

Examples

## Not run:
src <- src_sqlite()
docdb_create(src, "mtcars", mtcars)
docdb_update(src, "mtcars", value = mtcars[3, 4:5], query = '{"gear": 3}')
docdb_update(src, "mtcars", value = '{"carb":999}', query = '{"gear": 5}')
docdb_get(src, "mtcars")

## End(Not run)

mapdata

mapdata JSON data set

Description

mapdata JSON data set

Usage

mapdata

Format

A JSON string with ragged, nested map
Description

• src_etcd: etcd removed, with all its S3 methods for docdb_*
• src_redis: redis removed, with all its S3 methods for docdb_*

src

Setup database connections

Description

Setup database connections

Details

There is a src_*() function to setup a connection to each of the database backends. Each has their own unique set of parameters.

• MongoDB - src_mongo()
• SQLite - src_sqlite()
• Elasticsearch - src_elastic()
• CouchDB - src_couchdb()

Documentation details for each database:

• MongoDB - https://docs.mongodb.com/
• SQLite/JSON1 - https://www.sqlite.org/json1.html
• CouchDB - http://docs.couchdb.org/

Documentation for R packages used by nodbi for the databases:

• mongolite - https://CRAN.R-project.org/package=mongolite
• RSQLite - https://CRAN.R-project.org/package=RSQLite
• sofa - https://CRAN.R-project.org/package=sofa
• elastic - https://CRAN.R-project.org/package=elastic
src_couchdb  Setup a CouchDB database connection

Description

Setup a CouchDB database connection

Usage

```r
src_couchdb(
  host = "127.0.0.1",
  port = 5984,
  path = NULL,
  transport = "http",
  user = NULL,
  pwd = NULL,
  headers = NULL
)
```

Arguments

- **host** (character) host value, default: 127.0.0.1
- **port** (integer/numeric) Port. Remember that if you don’t want a port set, set this parameter to NULL. Default: 5984
- **path** (character) context path that is appended to the end of the url, e.g., bar in http://foo.com/bar. Default: NULL, ignored
- **transport** (character) http or https. Default: http
- **user** (character) Username, if any
- **pwd** (character) Password, if any
- **headers** (list) list of named headers

Details

uses **sofa** under the hood; uses **sofa::Cushion()** for connecting

Examples

```r
## Not run:
src_couchdb()

## End(Not run)
```
**Description**

Setup an Elasticsearch database connection

**Usage**

```r
src_elastic(
  host = "127.0.0.1",
  port = 9200,
  path = NULL,
  transport_schema = "http",
  user = NULL,
  pwd = NULL,
  force = FALSE,
  ...
)
```

**Arguments**

- **host** (character) the base url, defaults to localhost (http://127.0.0.1)
- **port** (character) port to connect to, defaults to 9200 (optional)
- **path** (character) context path that is appended to the end of the url. Default: NULL, ignored
- **transport_schema** (character) http or https. Default: http
- **user** (character) User name, if required for the connection. You can specify, but ignored for now.
- **pwd** (character) Password, if required for the connection. You can specify, but ignored for now.
- **force** (logical) Force re-load of connection details
- **...** Further args passed on to `elastic::connect()`

**Details**

uses `elastic` under the hood; uses `elastic::connect()` for connecting

**Examples**

```r
## Not run:
src_elastic()

## End(Not run)
```
**src_mongo**  
Setup a MongoDB database connection

**Description**  
Setup a MongoDB database connection

**Usage**  
```
src_mongo(collection = "test", db = "test", url = "mongodb://localhost", ...)  
```

**Arguments**

- **collection**  
  (character) Name of collection

- **db**  
  (character) Name of database

- **url**  
  (character) Address of the MongoDB server in Mongo connection string URI format, see `mongolite::mongo()`

- **...**  
  Additional named parameters passed on to `mongolite::mongo()`

**Details**

Uses **monoglite** under the hood; uses `mongolite::mongo()` for connecting

**Examples**

````
## Not run:
con <- src_mongo()
print(con)
## End(Not run)
```

**src_sqlite**  
Setup a RSQLite database connection

**Description**

Setup a RSQLite database connection

**Usage**

```
src_sqlite(dbname = "~/:memory:", ...)  
```

**Arguments**

- **dbname**  
  (character) name of database file, defaults to "~/:memory:" for an in-memory database, see `RSQLite::SQLite()`

- **...**  
  additional named parameters passed on to `RSQLite::SQLite()`
src_sqlite

Details

uses **RSQLite** under the hood

Examples

```r
## Not run:
con <- src_sqlite()
print(con)

## End(Not run)
```
Index

* data
  - contacts, 3
  - diamonds, 3
  - mapdata, 10

* package
  - nodbi-package, 2

contacts, 3

DBI::dbListTables(), 6
- diamonds, 3
docdb_create, 4
docdb_create(), 8
docdb_delete, 5
docdb_exists, 6
docdb_get, 7
docdb_list, 8
docdb_query, 8
docdb_query(), 5
docdb_update, 9
docdb_update(), 4

elastic::aliases_get(), 8
elastic::connect(), 13
elastic::docs_bulk, 4
elastic::docs_bulk_update, 10
elastic::index_exists(), 6
elastic::Search(), 7, 9

mapdata, 10
mongolite::mongo(), 4–7, 9, 10, 14

nodbi (nodbi-package), 2
nodbi-defunct, 11
nodbi-package, 2

RSQLite::SQLite(), 14

sofa::Cushion(), 12
sofa::db_alldocs(), 7
sofa::db_bulk_create(), 4, 10

sofa::db_info(), 6, 8
sofa::db_query(), 9
src, 11
src_couchdb, 12
src_couchdb(), 4–11
src_elastic, 13
src_elastic(), 4–11
src_etcd, 11
src_mongo, 14
src_mongo(), 4–11
src_redis, 11
src_sqlite, 14
src_sqlite(), 4–11

uuid::UUIDgenerate(), 4