
couchdb-requests Documentation

Release 0.0.1

Adam Lofts

March 01, 2017

1	Server: Connecting to CouchDB	3
2	Database: Saving and querying data	5
3	View: Map reduce queries	9
4	Indices and tables	11

[Getting Started](#)

[API Docs:](#)

Server: Connecting to CouchDB

class couchdbreq.**Server** (*uri='http://127.0.0.1:5984', session=None, timeout=5*)

A Server object represents the connection to the CouchDB database.

Parameters

- **uri** – URI of the server
- **session** – A *couchdbreq.Session* object. Use this to configure the connection parameters such as timeout, connection pool size and authentication.

active_tasks ()

return active tasks

create_db (*dbname*)

Create a database on the server

Parameters **dbname** – unicode name of the db

Returns *couchdbreq.Database*

Raise *couchdbreq.exceptions.DatabaseExistsException* If the database already exists

get_db (*dbname, is_verify_existence=True*)

Get a *couchdbreq.Database* object for an existing database

If the database does not exist and *is_verify_existence=True* then *couchdbreq.exceptions.ResourceNotFound* will be raised.

Parameters

- **dbname** – unicode name of the db
- **is_verify_existence** – bool Set to false to avoid a HEAD request checking that the database exists

Returns *couchdbreq.Database*

Raise *couchdbreq.exceptions.ResourceNotFound* If the database does not exist

get_db_names ()

Get all database names on the server

Returns List of unicode database names

get_dbs ()

Iterate over all databases

Returns List of *couchdbreq.Database* objects

get_info()

Get server info

get_or_create_db(dbname)

Get a database or create new database if missing.

Parameters **dbname** – unicode name of the db

Returns couchdbreq.Database

replicate(source, target, cancel=False, continuous=False, create_target=False, doc_ids=None, filter=None, proxy=None, query_params=None)

Replicate a database

More info about replication here: <http://wiki.apache.org/couchdb/Replication>

Parameters

- **source** – URI or dbname of the source
- **target** – URI or dbname of the target

class couchdbreq.Session

Bases: requests.sessions.Session

The http session for a server to use.

See <http://docs.python-requests.org/en/latest/api/#sessionapi>.

Database: Saving and querying data

class couchdbreq.database.**Database** (*server*, *name*, *create=False*, *get_or_create=False*, *is_verify_existance=True*)

Provides access to a CouchDB database

Do not construct directly. Use `couchdbreq.Server.get_db()` or `couchdbreq.Server.create_db()`.

all_docs (*schema=None*, *startkey=<object object>*, *endkey=<object object>*, *keys=None*, *key=<object object>*, *startkey_docid=<object object>*, *endkey_docid=<object object>*, *stale=None*, *descending=False*, *skip=0*, *limit=None*, *group=<object object>*, *group_level=<object object>*, *reduce=<object object>*, *include_docs=False*, *inclusive_end=True*, *update_seq=False*)

Get all documents from a database

You can use `all()`, `one()`, `first()` on the returned View object just like a View from `couchdbreq.database.view()`.

Returns couchreq.view.View

changes (*since=None*, *limit=None*, *descending=False*, *filter=None*, *include_docs=False*, *style='main_only'*)

Get changes from the db

Only feed=normal is supported because other feed types involve integration with a mainloop

compact ()

Compact database

compact_view (*view_name*)

Compact a view

Parameters *view_name* – The name of the view _design/<view_name>

copy_doc (*doc*, *dest=None*)

Copy an existing document to a new id. If *dest* is None, a new uuid will be requested

Parameters

- **doc** – dict or string, document or document id
- **dest** – basestring or dict. if *_rev* is specified in dict it will override the doc

delete_attachment (*doc*, *name*)

Delete attachment on the document

Parameters

- **doc** – dict

- **name** – name of attachment (unicode or str)

Returns dict, with member ok set to True if delete was ok.

delete_doc (*doc*)

Delete a document

The document will have a `_deleted` field set to true.

Parameters **doc** – The doc

Returns dict like: `{"ok":true,"rev":"2839830636"}`

delete_docs (*docs, all_or_nothing=False*)

Delete many docs at once.

It adds `'_deleted'` member to doc then uses `bulk_save` to save them.

Parameters **all_or_nothing** – In the case of a power failure, when the database

restarts either all the changes will have been saved or none of them. However, it does not do conflict checking, so the documents will

See also:

HTTP Bulk Document API <http://wiki.apache.org/couchdb/HTTP_Bulk_Document_API>

ensure_full_commit ()

Commit all docs in memory

This can be used after saving docs with `batch=True` to ensure they are all saved to disk.

fetch_attachment (*id_or_doc, name, stream=False, stream_chunk_size=16384*)

Get an attachment in a document

Note: If you have stored text e.g. utf8 in the attachment you will need to decode the response to this call using `.decode('utf8')`.

Parameters

- **id_or_doc** – str or dict, doc id or document dict
- **name** – name of attachment (unicode or str)
- **stream** – boolean, if True return a file object
- **stream_chunk_size** – Size in bytes to return per stream chunk (default 16 * 1024)

Returns Bytestring or file like iterable if `stream=True`

get_doc (*docid, rev=None, schema=None*)

Get document from database

Parameters

- **docid** – str, document id to retrieve
- **rev** – Get a specific revision of a document
- **schema** – A schema to pass. This is an object with a function `wrap_doc(doc)`

which will be used to map the response.

Returns dict, representation of CouchDB document as a dict.

Raise `couchdbreq.exceptions.InvalidDocNameError` if the docid is invalid

get_fragmentation()

Get the fragmentation on this database 0 <= fragmentation <= 100 See <http://wiki.apache.org/couchdb/Compaction>

get_info()

Get database information

Returns dict

get_rev(docid)

Get last revision from docid (the ‘_rev’ member)

Parameters **docid** – str, undecoded document id.

Return **rev** str, the last revision of document.

get_server()

Get the couchdbreq.server.Server object hosting this database :return: couchdbreq.server.Server

get_view_group_info(view_group)

Get the info of a view group

Parameters **view_group** – ‘designname’

Returns info

length()

Count the number of documents in the database.

This is implemented with a HEAD request to the database.

put_attachment(doc, content, name=None, content_type=None, content_length=None)

Add attachment to a document

If you are storing unicode text then you must encode before passing it to this function. e.g. `db.put_attachment(doc, u"Some unicode £".encode("utf8"), "My unicode attachment", "text/plain")`

Parameters

- **doc** – dict
- **content** – str or file like object.
- **name** – name of attachment (unicode or str) encoded as utf8
- **content_type** – string, mimetype of attachment. If you don’t set it, it will be autodetected.
- **content_length** – int, size of attachment in bytes

Returns bool, True if everything was ok.

save_doc(doc=None, encode_attachments=True, batch=False)

Save a document. It will use the `_id` member of the document or request a new uuid from CouchDB. IDs are attached to documents on the client side because POST has the curious property of being automatically retried by proxies in the event of network segmentation and lost responses.

Parameters

- **doc** – dict. doc is updated with doc ‘_id’ and ‘_rev’ properties returned by CouchDB server when you save.
- **batch** – If true then use reduced guarantee that the document has been saved. The `_rev` field will not be updated.

Returns doc updated with ‘_id’ and ‘_rev’

Raise `couchdbreq.exceptions.ResourceConflict` if the save generated a conflict

save_docs (*docs*, *use_uuids=True*, *all_or_nothing=False*)

Save multiple docs at once

Parameters

- **docs** – list of docs
- **use_uuids** – add _id in doc who don’t have it already set.
- **all_or_nothing** – In the case of a power failure, when the database

restarts either all the changes will have been saved or none of them. However, it does not do conflict checking.

See also:

HTTP Bulk Document API <http://wiki.apache.org/couchdb/HTTP_Bulk_Document_API>

view (*view_name*, *schema=None*, *startkey=<object object>*, *endkey=<object object>*, *keys=None*, *key=<object object>*, *startkey_docid=<object object>*, *endkey_docid=<object object>*, *stale=None*, *descending=False*, *skip=0*, *limit=None*, *group=<object object>*, *group_level=<object object>*, *reduce=<object object>*, *include_docs=False*, *inclusive_end=True*, *update_seq=False*)

Query for view results

Parameters

- **view_name** – ‘designname/viewname’
- **schema** – A schema to pass. This is an object with a function `wrap_row(row)`

which will be used to map the response.

Returns `couchreq.view.View`

view_cleanup ()

Clean up views

View: Map reduce queries

class `couchdbreq.view.View` (*db*, *view_path*, *schema=None*, *params=None*)

An iterable object representing a query.

Do not construct directly. Use `couchdbreq.Database.view()`, `couchdbreq.Database.all_docs()` or `couchdbreq.view.View.filter()`.

all ()

Get a list of all rows

Returns `list`

count ()

Return the number of results

Returns `int`

filter (*startkey=<object object>*, *endkey=<object object>*, *keys=<object object>*, *key=<object object>*, *startkey_docid=<object object>*, *endkey_docid=<object object>*, *skip=<object object>*, *limit=<object object>*, *inclusive_end=<object object>*)

Return a new View object with updated query parameters. The original View object remains unchanged.

Returns A new `couchdbreq.view.View` object

first (*is_null_exception=False*)

Return the first result of this query or None if the result doesn't contain any rows.

Parameters **is_null_exception** – If True then raise `couchdbreq.exceptions.NoResultFound` if no results are found.

Returns A dict representing the row result or None

one (*is_null_exception=False*)

Return exactly one result or raise an exception if multiple results are found.

Parameters **is_null_exception** – If True then raise `couchdbreq.exceptions.NoResultFound` if no results are found.

Returns A dict representing the row result or None

Indices and tables

- `genindex`
- `modindex`
- `search`

A

active_tasks() (couchdbreq.Server method), 3
all() (couchdbreq.view.View method), 9
all_docs() (couchdbreq.database.Database method), 5

C

changes() (couchdbreq.database.Database method), 5
compact() (couchdbreq.database.Database method), 5
compact_view() (couchdbreq.database.Database method), 5
copy_doc() (couchdbreq.database.Database method), 5
count() (couchdbreq.view.View method), 9
create_db() (couchdbreq.Server method), 3

D

Database (class in couchdbreq.database), 5
delete_attachment() (couchdbreq.database.Database method), 5
delete_doc() (couchdbreq.database.Database method), 6
delete_docs() (couchdbreq.database.Database method), 6

E

ensure_full_commit() (couchdbreq.database.Database method), 6

F

fetch_attachment() (couchdbreq.database.Database method), 6
filter() (couchdbreq.view.View method), 9
first() (couchdbreq.view.View method), 9

G

get_db() (couchdbreq.Server method), 3
get_db_names() (couchdbreq.Server method), 3
get_dbs() (couchdbreq.Server method), 3
get_doc() (couchdbreq.database.Database method), 6
get_fragmentation() (couchdbreq.database.Database method), 6
get_info() (couchdbreq.database.Database method), 7
get_info() (couchdbreq.Server method), 3

get_or_create_db() (couchdbreq.Server method), 4
get_rev() (couchdbreq.database.Database method), 7
get_server() (couchdbreq.database.Database method), 7
get_view_group_info() (couchdbreq.database.Database method), 7

L

length() (couchdbreq.database.Database method), 7

O

one() (couchdbreq.view.View method), 9

P

put_attachment() (couchdbreq.database.Database method), 7

R

replicate() (couchdbreq.Server method), 4

S

save_doc() (couchdbreq.database.Database method), 7
save_docs() (couchdbreq.database.Database method), 8
Server (class in couchdbreq), 3
Session (class in couchdbreq), 4

V

View (class in couchdbreq.view), 9
view() (couchdbreq.database.Database method), 8
view_cleanup() (couchdbreq.database.Database method), 8