madpink

couchdb4livecode

madpink edited this page on Jul 13, 2017 · 9 revisions

Daybed - A Library for Apache CouchDB
There are 3 methods to use the Daybed library:
1. The Main functions which give access to all CouchDB commands.
2. The Quick functions which allow for quick access to essential functions.
3. The Sync functions which synchronizes local documents with a database designated for the user.

Note: FastJSON has been appended to the library, and is used by default for JSON encoding and decoding.
- The primary commands have "fast." prefixed to the commands to differentiate them from the built in commands (mergJSON).
- I've modified the handling of numeric arrays so that order is maintained.
- I've added code to "fill in the blanks" with null if a numeric array has numeric gaps. (Optional)
- I've added local variables for "keep numeric" and "fill in the blanks" so they do not need to get passed back in as an argument.
- The original is maintained by Bob Hall here: https://github.com/bhall2001/fastjson

I recommend using FastJSON over mergJSON because:

- MergJSON only creates JSON arrays with numeric keys that go from 1 to N with no gaps. I'm experimenting with some commands to improve this, but for now it requires a few extra steps to work around.
- I've had difficulties with MergJSON in Ubuntu standalones. (Sorry, I REALLY need to fill out bug report for this)
- FastJSON is easier to use with Livecode-server. (No need to compile a stack and include the external)

Gregory Miller edited this page on Jul 8, 2019 · 8 revisions

Main Functions:

`couch.get(pFunk,pURL,pDB,pDocID,pParams,pOptions,pFormat)`

- The couch.get function retrives a document, multiple documents, or information.

`couch.put(pFunk,pURL,pDB,pDoc,pParams,pOptions,pFormat)`

- The couch.put function inserts a value or a document into a database.

`couch.post(pFunk,pURL,pDB,pDoc,pParams,pOptions,pFormat)`

- The couch.post function inserts multiple documents or values into a database.

`couch.delete(pFunk,pURL,pDB,pDocID,pParams,pOptions,pFormat)`

- The couch.delete function deletes a document or database, or a config key.
    - Note: the database is really deleted, however the document can still be retrieved if the id and rev are known.

Function Parameters

- **pFunk** (always required) Couch function being called; the leading underscore can be omitted, for example "all_docs"
    - a list is included below of functions that have been tested
    - there are four categories of functions: system, database, document, design document
- **pURL** (always required) the URL of the CouchDB installation, including "http://" and the port number if applicable
    - For example: "http://192.168.23.42:5984/"
    - With username/password: "http://admin:trustno1@192.168.23.42:5984/"
- **pDB** (required when acting on or retrieving from a database) - the name of the database being accessed
    - should be blank for system functions, must be included for database, document, and design document
- **pDocID** (for get and delete functions) the document "_id" being retrieved
    - should be blank for system and database functions, must be included for document and design document
    - use the pDocID param to specify the name of a design document being called
- **pDoc** - (for put and post functions) array containing the data being converted into a document for the database
    - should be blank for system functions
    - required for document and design document (also used in a db function)

- **pParams** - an array with any OPTIONAL parameters, with the parameter as a key.
    - These parameters are specified in the CouchDB API
    - For example: to download documents when running the _all_docs function, and limit the list to only 10 records:
        - `put true into pParams["include\_docs"]`
        - `put 10 into pParams["limit"]`
    - Example, to include the revision number for a document
        - `put "13-8j4f9438jf3498j98fy39d23d" into pParams["rev"]`
- **pOptions** - (optional) header options, including authentication, config values and return format
    - For Design Documents, use the following options:
        - pOptions["ddoc"]["func"] for the function being called (info, view, show, list, update)
        - pOptions["ddoc"]["name"] for the name of the specific function programmed in the ddoc
        - pOptions["ddoc"]["otherid"] and pOptions["ddoc"]["otherfunc"] for further extended URLs
    - For Authentication, use the following options:
        - pOptions["authtype"] for the type of authentication being used, valid values:
            - "login" - default value if blank, must be in the format username:password
            - "encoded" - base64 encoded version of username:password
            - "cookie" - cookie-based login
        - pOptions["authval"] with the login, encoded login or cookie value
    - To set the return format:
        - pOptions["format"] with a valid format value (array, rawjson, prettyjson)
    - When setting _config values:
        - pOptions["key"] with the key to be set

        - pOptions["value"] with the value to set it to
    - When PUTting an attachment (using the "attach" function):
        - pOptions["attachname"] with the file name to be used in CouchDB
        - pOptions["attachpath"] with the full path to the file being uploaded
        - pOptions["attachmode"] with "bin" or "text", "bin" will be used if left blank
        - pOptions["attachtype"] with the MIME content-type, for example "image/jpg" or "application/pdf"
            - If left blank, the script will attempt to fill in an applicable type
    - When PUTting an attachment (inline):
        - Repeat for each file to be included
        - `pOptions["attachments"][filename]["path"]` with the full file path of the file to be uploaded
        - `pOptions["attachments"][filename]["mime"]` with the MIME content-type of the file
        - `pOptions["attachments"][filename]["mode"]` with "bin" or "text", with "bin" used as default
    - When GETting an attachment:
        - pOptions["attachment"] with the file name in DB
        - pOptions["destination"] with the destination location (include filename)

Other Functions/Parameters

`couch.getrev(pURL,pDB,pDocID,pOptions)`

This function returns the most recent revision number for the specified document.

`couch.securedb(pFunk,pURL,pDB,pOptions,pAdminNames,pAdminRoles,pMemberNames,pMemberRoles)`

The couch.securedb function sets the "_security" document for the specified database.

- pFunk - "set" (replace current security), "add" adds user/roles to existing security, "delete" removes user/roles from existing
- pAdminNames: sets the given names up with admin rights (read,write,delete all)
- pAdminRoles: sets the given user roles up with admin rights (read,write,delete all)
- pMemberNames: sets the given names up with member rights (read, write documents/read design documents)
- pMemberRoles: sets the given user roles up with member rights (read, write documents/read design documents)

`couch.adduser(pURL,pUser,pPass,pOptions,pAddDB)`
The couch.adduser function inserts a new record into the "_users" database, optionally creates a database for the user, and sets the new user as the admin and member (which makes that user the only one who can access it).

- pUsername: Username of the person signing up.
- pPassword: Password for the account
- pOptions["roles"]: can be used to assign the user to roles, must be in a numbered array
- pAddDB: if true, adds a database with the user's name and secures it
- Note: only an admin can create a user

`couch.peruserDB(pUsername)`
Requires "couchperuser" to be installed (https://github.com/etrepum/couchperuser) The couch.peruserDB function returns the database name associated with the username specified.

Authentication:

if the CouchDB URL requires authentication, it can be achieved in one of two ways:

1. Include the username and password as part of the URL, for

example:
    - "http://admin:passw0rd@192.168.0.42:5984/"

2. Use the pOptions parameter, and the script will encode the username and password into the httpheaders:
    - `put "admin:passw0rd" into pOptions["authval"]`
    - `put "login" into pOptions["authtype"]`
3. Use the pOptions parameter with a base 64 encoded username:password, which will be added to the httpheaders:
    - for example, put base64encode("username:password") will yield the string dXNlcm5hbWU6cGFzc3dvcmQ=
    - `put "dXNlcm5hbWU6cGFzc3dvcmQ=" into pOptions["authval"]`
    - `put "encoded" into pOptions["authtype"]`
4. Use cookies/sessions...
    - First get cookie by posting username and password to sessions
        - `put "admin" into pDoc["name"]`
        - `put "passw0rd" into pDoc["password"]`
    - `put couch.post("session",tURL,,pDoc) into theCookie`
    - Store it somewhere. For each subsequent call, send theCookie in pOptions
    - `put theCookie into pOptions["authval"]`
        - `put "cookie" into pOptions["authtype"]`

Adding Attachments

```
* Files can be attached to documents inline or through
the standalone API
* In order to properly view a document online (e.g. -
through a CouchApp) the proper MIME-Type needs to be
specified
* To use the standalone API:
    * Must specify the revision number in
pParams["rev"]
    * Each attachment needs to be added seperately
    * Use these options:
        * pOptions["attachname"] - name of the file as
```

```
saved to the document
        * pOptions["attachpath"] - path to the file
        * pOptions["attachmode"] - binary or text
        * pOptions["attachtype"] - MIME type (will try
to generate if blank)
* To add documents inline:
    * Use the options" pOptions["attachments"] with a
seperate key for each file
    * If the document exists, the entire document needs
to be reuploaded
```

Return Format

```
* include pOptions["format"] with "array", "rawjson" or
"prettyjson" for the return format
* the stack can have a customProperty called
"preferredFormat" which can be one of those three
values,
* if pOptions["format"] is blank, then
"preferredFormat" will be used
* if neither has a value, then "array" will be used
```

To-Do List (Extra functions)

- Couch based message queueing system
- Daybed Toolbox
- More backendian services

Main Functions: Examples

madpink edited this page on Jun 30, 2017 · 2 revisions

```
/
GET /    Returns the welcome message and version
information
     put "http://127.0.0.1:5984/" into tURL
     put couch.get(slash,tURL)
```

```
/_active_tasks
GET /_active_tasks    Obtains a list of the tasks
running in the server
     put "http://127.0.0.1:5984/" into tURL
     put couch.get(active_tasks,tURL)

/_all_dbs
GET /_all_dbs    Returns a list of all the databases
     put "http://127.0.0.1:5984/" into tURL
     put couch.get(all_dbs,tURL)

/_config
GET /_config Obtains a list of the entire server
configuration
   put "http://127.0.0.1:5984/" into tURL
   put couch.get(config,tURL)

/_config/{section}
GET /_config/{section}    Returns all the configuration
values for the specified section
   put "http://127.0.0.1:5984/" into tURL
   put "uuids" into tOptions["section"]
   put couch.get(config,tURL,,,,tOptions)

/_config/{section}/{key}
GET /_config/{section}/{key}  Returns a specific
section/configuration value
   put "http://127.0.0.1:5984/" into tURL
   put "uuids" into tOptions["section"]
   put "algorithm" into tOptions["key"]
   put couch.get(config,tURL,,,,tOptions)

PUT /_config/{section}/{key}  Sets the specified
configuration value
   put "http://127.0.0.1:5984/" into tURL
   put "uuids" into tOptions["section"]
   put "algorithm" into tOptions["key"]
   put "random" into tOptions["value"]
   put couch.put(config,tURL,,,,tOptions)

DELETE /_config/{section}/{key}   Removes the current
```

```
setting
   put "http://127.0.0.1:5984/" into tURL
   put "uuids" into tOptions["section"]
   put "algorithm" into tOptions["key"]
   put couch.delete(config,tURL,,,,tOptions)


/_db_updates
GET /_db_updates Return the server changes of databases
     put "http://127.0.0.1:5984/" into tURL
     put couch.get("db_updates",tURL)

/_log
GET /_log    Returns the server log file
     put "http://127.0.0.1:5984/" into tURL
     put couch.get(log,tURL)

/_replicate
POST /_replicate Starts or cancels the replication
     put "http://127.0.0.1:5984/" into tURL
     put "test1" into tDoc["source"]
     put "http://192.168.42.23:5984/test2" into
tDoc["target"]
     put couch.post(restart,tURL,,tDoc)

/_restart
POST /_restart   Restarts the server
     put "http://127.0.0.1:5984/" into tURL
     put couch.post(restart,tURL)

/_session
GET /_session    Returns Cookie-based login user
information
     put "http://127.0.0.1:5984/" into tURL
     put couch.get("session",tURL) into tInfo

POST /_session   Authenticates user by Cookie-based
user login
     put "http://127.0.0.1:5984/" into tURL
     put "admin" into pDoc["name"]
     put "passw0rd" into pDoc["password"]
     put couch.post("session",tURL,,pDoc) into
theCookie
          ---returns a string with the cookie in it

DELETE /_session Logout Cookie-based user
     put "http://127.0.0.1:5984/" into tURL
     put couch.delete("session",tURL) into tInfo

/_stats
GET /_stats  Returns server statistics
     put "http://127.0.0.1:5984/" into tURL
     put couch.get(stats,tURL)

/_uuids
GET /_uuids  Generates a list of UUIDs from the server
     put "http://127.0.0.1:5984/" into tURL
     put couch.get(uuids,tURL)


DATABASE API


/{db}
GET /{db}    Returns the database information
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put couch.get(db,tURL,tDB)
POST /{db}   Creates a new document with generic ID if
he had not specified
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "somedata" into tDoc["somekey"]
     put "otherdata" into tDoc["otherkey"]
     put couch.post(db,tURL,tDB,tDoc)
PUT /{db}    Creates a new database
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put couch.put(db,tURL,tDB)
DELETE /{db} Deletes an existing database
     put "http://127.0.0.1:5984/" into tURL
     put "testdb2" into tDB
     put couch.delete(db,tURL,tDB)

/{db}/_all_docs
GET /{db}/_all_docs   Returns a built-in view of all
documents in this database
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put couch.get(all_docs,tURL,tDB)
POST /{db}/_all_docs Returns certain rows from the
built-in view of all documents
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "c3f2c12bc8c242826e1849097900091d" into
tDocIDs["keys"][1]
     put "c3f2c12bc8c242826e1849097900358d" into
tDocIDs["keys"][2]
     put couch.post(all_docs,tURL,tDB,tDocIDs)

/{db}/_bulk_docs
POST /{db}/_bulk_docs Inserts or updates multiple
documents in to the database in a single request
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "somedata" into tDoc[1]["somekey"]
     put "otherdata" into tDoc[1]["otherkey"]
     put "somedata2" into tDoc[2]["somekey"]
     put "otherdata2" into tDoc[2]["otherkey"]
     put couch.post(bulk_docs,tURL,tDB,tDoc) into tZZZ

/{db}/_changes
GET /{db}/_changes    Returns changes for the given
database
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put couch.get(changes,tURL,tDB)
POST /{db}/_changes   Returns changes for the given
database for certain document IDs
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "c3f2c12bc8c242826e1849097900091d" into
tDocIDs["doc_ids"][1]
```

```
     put "c3f2c12bc8c242826e1849097900358d" into
tDocIDs["doc_ids"][2]
     put couch.post(changes,tURL,tDB,tDocIDs)

/{db}/_compact
POST /{db}/_compact   Starts a compaction for the
database
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put couch.post(compact,tURL,tDB)

/{db}/_compact/{ddoc}
POST /{db}/_compact/{ddoc}    Starts a compaction for
all the views in the selected design document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "testdoc" into tDocID
     put couch.post(compactdesign,tURL,tDB,tDocID)

/{db}/_ensure_full_commit
POST /{db}/_ensure_full_commit    Makes sure all
uncommitted changes are written and synchronized to the
disk
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put couch.post(ensure_full_commit,tURL,tDB)

/{db}/_local/{docid}
GET /{db}/_local/{docid}  Returns the latest revision
of the non-replicated document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "0dc77cd3-389b-43c1-aef4-e2a5a31eef72" into
tDocID
     put couch.get("localdoc",tURL,tDB,tDocID)

PUT /{db}/_local/{docid}  Inserts a new version of the
non-replicated document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "somedata" into tDoc["somekey"]
     put "otherdata" into tDoc["otherkey"]
     put couch.put("localdoc",tURL,tDB,tDoc)

DELETE /{db}/_local/{docid}   Deletes the non-
replicated document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "0dc77cd3-389b-43c1-aef4-e2a5a31eef72" into
tDocID
     put "1-e0af530ae2775044d7b2db4c91cc18a1" into
tParams["rev"]
     put
couch.delete("localdoc",tURL,tDB,tDocID,tParams)

/{db}/_missing_revs
POST /{db}/_missing_revs  By given list of document
revisions, returns the document revisions that do not
exist in the database
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "c3f2c12bc8c242826e1849097900091d" into tID
     put "3-b06fcd1c1c9e0ec7c480ee8aa467bf3b" into
tDoc[tID][1]
     put "4-0e871ef78849b0c206091f1a7af6ec41" into
tDoc[tID][1]
     put couch.post("missings_revs",tURL,tDB,tDocIDs)

/{db}/_purge
POST /{db}/_purgePurges some historical documents
entirely from database history
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "c3f2c12bc8c242826e1849097900091d" into tID
     put "4-dc8088c3be9d44b41f87ba1470064672" into
tDoc[tID][1]
     put couch.post(purge,tURL,tDB,tDoc)

/{db}/_revs_diff
POST /{db}/_revs_diff By given list of document
revisions, returns differences between the given
revisions and ones that are in the database
```

```
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "c3f2c12bc8c242826e1849097900091d" into tID
     put "3-b06fcd1c1c9e0ec7c480ee8aa467bf3b" into
tDoc[tID][1]
     put "4-0e871ef78849b0c206091f1a7af6ec41" into
tDoc[tID][1]
     put couch.post("revs_diff",tURL,tDB,tDocIDs)

/{db}/_revs_limit
GET /{db}/_revs_limit Returns the limit of historical
revisions to store for a single document in the
database
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put couch.get(revs_limit,tURL,tDB)
PUT /{db}/_revs_limit Sets the limit of historical
revisions to store for a single document in the
database
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put couch.put(revs_limit,tURL,tDB,500)

/{db}/_security
GET /{db}/_security  Returns the special security
object for the database
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put couch.get(security,tURL,tDB)
PUT /{db}/_security  Sets the special security object
for the database
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "uber" into tDoc["admins"]["names"][1]
     put "admins" into tDoc["admins"]["roles"][1]
     put "user" into tDoc["members"]["names"][1]
     put "developer" into tDoc["members"]["roles"][1]
     put couch.put(security,tURL,tDB,tDoc)

/{db}/_temp_view
POST /{db}/_temp_view Executes a given view function for
```

```
all documents and returns the result
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
      put "function(doc) { if (doc.value)
{ emit(doc.value, null); } }" into tDoc["map"]
     put "_count" into tDoc["_reduce"]
     put couch.post("temp_view",tURL,tDB,tDoc)

/{db}/_view_cleanup
POST /{db}/_view_cleanup  Removes view files that are
not used by any design document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put couch.post("view_cleanup",tURL,tDB)

DOCUMENT API

/{db}/{docid}
GET /{db}/{docid}Returns the documentrr
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "0dc77cd3-389b-43c1-aef4-e2a5a31eef72" into
tDocID
     put couch.get(doc,tURL,tDB,tDocID)
PUT /{db}/{docid}Creates a new document, or new version
of an existing document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "somedata" into tDoc["somekey"]
     put "otherdata" into tDoc["otherkey"]
     put couch.put(doc,tURL,tDB,tDoc)
DELETE /{db}/{docid} Deletes the document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "0dc77cd3-389b-43c1-aef4-e2a5a31eef72" into
tDocID
     put "1-e0af530ae2775044d7b2db4c91cc18a1" into
tParams["rev"]
     put couch.delete(doc,tURL,tDB,tDocID,tParams)

/{db}/{docid}/{attname}
```

```
GET /{db}/{docid}/{attname}   Gets the attachment of a
document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "thisdoc" into tDocID
     put "my_cat.jpg" into tOptions["attachment"]
     put couch.get(attach,tURL,tDB,tDocID)

PUT /{db}/{docid}/{attname}   Adds an attachment of a
document
     put "http://127.0.0.1:5984/" into tURL
     put "test_suite_db2" into tDB
     put "zxsd3324d" into tOptions["docid"]
     put "my_cat.jpg" into tOptions["attachname"]
     put "1-465aa47fc9a072e79d5898a3257acb4c" intp
tParams["rev"]
     put "/User/pink/testdata/my_cat.jpg" into
tOptions["attachpath"]
     put "image/jpg" into tOptions["attachtype"]
     put "bin" into tOptions["attachmode"]
     put couch.get(attach,tURL,tDB)

DELETE /{db}/{docid}/{attname}    Deletes an attachment
of a document
     put "http://127.0.0.1:5984/" into tURL
     put "test_suite_db2" into tDB
     put "zxsd3324d" into tDocID
     put "my_cat.jpg" into tOptions["attachment"]
     put "1-465aa47fc9a072e79d5898a3257acb4c" intp
tParams["rev"]
     put couch.delete(attach,tURL,tDB,tDocID)

DESIGN DOCUMENT API

/{db}/_design/{ddoc}
GET /{db}/_design/{ddoc}  Returns the design document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "testdoc" into tDocID
     put couch.get(design,tURL,tDB,tDocID)
PUT /{db}/_design/{ddoc}  Creates a new design
```

```
document, or new version of an existing one
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "testdoc2" into pDoc["_id"]
     put "javascript" into pDoc["language"]
     put "function(doc) {"&cr&"  emit(null,
doc);"&cr&"}" into pDoc["views"]["myquery"]["map"]
     put couch.put(design,tURL,tDB,pDoc)
DELETE /{db}/_design/{ddoc}   Deletes the design
document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "myviewdoc" into tDocID
     put "1-e0af530ae2775044d7b2db4c91cc18a1" into
tParams["rev"]
     put couch.delete(design,tURL,tDB,tDocID,tParams)

/{db}/_design/{ddoc}/_info
GET /{db}/_design/{ddoc}/_info    Returns view index
information for the specified design document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "testdoc" into tDocID
     put "info" into tOptions["ddoc"]["func"]
     put couch.get(design,tURL,tDB,tDocID,tOptions)
into zzz

/{db}/_design/{ddoc}/_list/{func}/{other-ddoc}/{view}
GET /{db}/_design/{ddoc}/_list/{func}/{other-ddoc}/
{view}   Executes a list function against the view from
other design document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "testdoc" into tDocID
     put "list" into tOptions["ddoc"]["func"]
     put "myspeciallist" into tOptions["ddoc"]["name"]
     put "otherdoc" into tOptions["ddoc"]["otherdoc"]
     put "view" into tOptions["ddoc"]["otherfunc"]
     put couch.get(design,tURL,tDB,tDocID,tOptions)
into zzz
```

```
POST /{db}/_design/{ddoc}/_list/{func}/{other-ddoc}/
{view}
     The same as GET

/{db}/_design/{ddoc}/_list/{func}/{view}
GET /{db}/_design/{ddoc}/_list/{func}/{view}   Executes
a list function against the view from the same design
document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "testdoc" into tDocID
     put "list" into tOptions["ddoc"]["func"]
     put "myspeciallist" into tOptions["ddoc"]["name"]
     put "view" into tOptions["ddoc"]["otherdoc"]
     put couch.get(design,tURL,tDB,tDocID,tOptions)
into zzz
POST /{db}/_design/{ddoc}/_list/{func}/{view}
     The same as GET

/{db}/_design/{ddoc}/_show/{func}
GET /{db}/_design/{ddoc}/_show/{func}  Executes a show
function against null document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "testdoc" into tDocID
     put "show" into tOptions["ddoc"]["func"]
     put "myshowfunc" into tOptions["ddoc"]["name"]
     put couch.get(design,tURL,tDB,tDocID,tOptions)
into zzz
POST /{db}/_design/{ddoc}/_show/{func}
     The same as GET

/{db}/_design/{ddoc}/_show/{func}/{docid}
GET /{db}/_design/{ddoc}/_show/{func}/{docid}  Executes
a show function against the specified document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "testdoc" into tDocID
     put "show" into tOptions["ddoc"]["func"]
     put "myshowfunc" into tOptions["ddoc"]["name"]
     put "47438242384239472349849" into tOptions["ddoc"]
```

```
["othername"]
     put couch.get(design,tURL,tDB,tDocID,tOptions)
into zzz
POST /{db}/_design/{ddoc}/_show/{func}/{docid}
     The same as GET

/{db}/_design/{ddoc}/_update/{func}
POST /{db}/_design/{ddoc}/_update/{func}   Executes an
update function
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "updatedoc" into tOptions["docid"]
     put "update" into tOptions["ddoc"]["func"]
     put "myupdatefunc" into tOptions["ddoc"]["name"]
     put couch.post(design,tURL,tDB,,tOptions) into zzz

/{db}/_design/{ddoc}/_update/{func}/{docid}
PUT /{db}/_design/{ddoc}/_update/{func}/{docid}
Executes an update function against the specified
document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "testdoc" into tDocID
     put "update" into tOptions["ddoc"]["func"]
     put "myview" into tOptions["ddoc"]["name"]
     put "ssdd98r98t443e" into tOptions["ddoc"]
["otherdoc"]
     put couch.put(design,tURL,tDB,tDocID,tOptions)
into zzz

/{db}/_design/{ddoc}/_view/{view}
GET /{db}/_design/{ddoc}/_view/{view}  Returns results
for the specified stored view
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "testdoc" into tDocID
     put "view" into tOptions["ddoc"]["func"]
     put "myview" into tOptions["ddoc"]["name"]
     put couch.get(design,tURL,tDB,tDocID,tOptions)
into zzz
```

```
POST /{db}/_design/{ddoc}/_view/{view} Returns certain
rows for the specified stored view
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "testdoc" into tOptions["docid"]
     put "view" into tOptions["ddoc"]["func"]
     put "myview" into tOptions["ddoc"]["name"]
     put "case0001" into tDoc["keys"][1]
     put "case0002" into tDoc["keys"][2]
     put couch.post(design,tURL,tDB,tDoc,tOptions) into
zzz


/{db}/_design/{ddoc}/{attname}
GET /{db}/_design/{ddoc}/{attname}Gets the attachment
of a design document
     put "http://127.0.0.1:5984/" into tURL
     put "testdb" into tDB
     put "thisddoc" into tDocID
     put "my_cat.jpg" into tOptions["attachment"]
     put couch.get(design,tURL,tDB,tDocID)

PUT /{db}/_design/{ddoc}/{attname}Adds an attachment of
a design document
     put "http://127.0.0.1:5984/" into tURL
     put "test_suite_db2" into tDB
     put "_design/mydesigndoc" into tOptions["docid"]
     put "my_cat.jpg" into tOptions["attachname"]
     put "1-465aa47fc9a072e79d5898a3257acb4c" intp
tParams["rev"]
     put "/User/pink/testdata/my_cat.jpg" into
tOptions["attachpath"]
     put "image/jpg" into tOptions["attachtype"]
     put "bin" into tOptions["attachmode"]
     put couch.get(attach,tURL,tDB)

DELETE /{db}/_design/{ddoc}/{attname} Deletes an
attachment of a design document
     put "http://127.0.0.1:5984/" into tURL
     put "test_suite_db2" into tDB
     put "myddoc" into tDocID
     put "my_cat.jpg" into tOptions["attachment"]
     put "1-465aa47fc9a072e79d5898a3257acb4c" intp
tParams["rev"]
     put couch.delete(design,tURL,tDB,tDocID)
```

Sync functions

madpink edited this page on Aug 12, 2017 · 16 revisions

The sync functions save all the data to a property set of an external/invisible stack for storge, and keeps the data in a property set on the main stack while the app is running.

The Sync commands:

dbdb.local.newuser *pURL pUser pPass pAuth pPrefix pOther pConflict pIDPrefix pApp*

This is the first command that needs to be run. It creates the save stack, saves all the necessary settings and creates the CouchDB user, user's database, installs the necessary design documents, and/or if there is already existing documents they are downloaded.

- pURL - the URL including port number of the CouchDB installation, e.g. http://127.0.0.1:5984
- pUser - User's username
- pPass - User's password
- pAuth - base64 encoded string for an administrator's username:password
- pPrefix - (optional) prefix of the database name, default: "user"
- pOther - (optional) array of other data to be saved in the user record (e.g. email address, secret question)
- pConflict - (optional) how to handle conflicting updated: server, local, or duplicate, default: "server"
    - server: server's copy wins conflict
    - local: local copy wins conflict

    - duplicate: both copies are saved, one with a modified name
- pIDPrefix - (optional) prefix for the id number of each document, default: "user"
- pApp - (optional) application name for searches, default: the same as pIDPrefix

dbdb.stack.load *pFileName pStackName pMainFolder pSubFolder pPropSet*

Should be put in the "openStack" handler.
Opens save stack in the background and copies its contents to the main stack.

- pFileName - file name to use for the save stack
- pStackName - name of the stack (for saving)
- pMainFolder - (optional) the main folder or specialFolder that the save stack will be saved in. Defaults:
    - Mac and Windows: defaults to "support"
    - iOS and Android: defaults to "document"
    - Linux and HTML5: defaults to "home"
- pSubFolder - (optional) a folder to be created in the main folder for the save file, defaults to the short name of the stack
- pPropSet - (optional) name for the property set to save the data (useful for using the same data file for multiple apps), default: "daybedDB"

dbdb.stack.save *pStackName pPropSet*

Should be put at least in the "closeStack".
Copies the user data from the main stack to the save stack and saves it.

- pStackName - name of the save stack
- pPropSet - the same as the one used in the load command (if applicable)

dbdb.local.sync

Begins the sync process between the data saved in the local stack and the user's CouchDB database.

dbdb.local.setdoc *pDoc*

Saves a document to the main property set and flags it as either new or changed. pDoc - array containing the full document
dbdb.local.deletedoc *pDocID*
Loads and saves a document, flagging it as deleted. pDocID - ID number of the document
dbdb.local.getdoc *pDocID*
Loads a document from the main property set. pDocID - ID number of the document