COUCH4MAT: CouchDB Matlab tbox

Table of Contents

Jonas Almeida, April 2010

Updated by Gustavo Delfino on Nov 2016

This manual was generated automatically by running pub.m .

The couch tbox revolves around a data structure and a single command. The data structure can be created by just pointing the command to a a URL where CouchDB is installed. If no URL is provided then the command assumes you are targetting a local deployment, and will use localhost:5984.

All CouchDB functionalities are coded under the couch command with one exception - the JSON parsers. These two parsers, json2mat and mat2json, collectively designated as json4mat.

Old versions of couch4mat and json4mat may still be available at http://couch4mat.googlecode.com and http://json4mat.googlecode.com The versions on https://github.com/mathbiol/couch4mat are newer.

Installation

You can download CouchDB for the different OSs from http://couchdb.apache.org/

In addition to Matlab this tutorial also assumes you have CURL installed. If you are running on Windows you can get it from http://curl.haxx.se/. If you are running this on a Mac or Linux you probably already have it.

couch()

Lets start by calling the local deployment, you can also target and arbitrary deployment by using the URL as the input argument, i.e. couch('url goes here')

```
c=couch
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
  Current
                                 Dload  Upload   Total   Spent    Left
  Speed
    0     0    0     0    0     0      0      0 --:--:--
 --:--:-- --:--:--     0 100   151  100   151
 0     0  10066      0 --:--:-- --:--:-- --:--:--
 10066{"couchdb":"Welcome","uuid":"96739cb96002ee7eb1ef6500c0b61b36","version":"1.
{"version":"1.6.1","name":"The Apache Software Foundation"}}


c =

     url: 'http://localhost:5984'
     opt: 'welcome'
    curl: [1x1 struct]
    info: [1x1 struct]
```

this structure, c, will now be used as the input argument of couch() to which we will add more arguments.

Create database

The name of the database you want to create can be passed directly in the c structure at c.db.name or can be sent as a third input argument, which we'll do here:

```
c=couch(c,'create db','mydb')
```

```
c =

     url: 'http://localhost:5984'
     opt: 'create db'
    curl: [1x1 struct]
    info: [1x1 struct]
      db: [1x1 struct]
```

List all databases

As with other usages of couch we'll use arguments that are as close as possible to CouchDB's own syntax. For example, to get a list of all databases one would call [c.url,'/_all_dbs'] so the second argument, and field in the updated c structure, is 'all_dbs':

```
c=couch(c,'all_dbs')
```

```
c =

        url: 'http://localhost:5984'
        opt: 'all_dbs'
       curl: [1x1 struct]
```

```
       info: [1x1 struct]
         db: [1x1 struct]
    all_dbs: {1x12 cell}
```

Insert a document

the document can be provided already as a JSON string

```
[c,doc]=couch(c,'insert doc','{"hello":"world"}');
```

or as a structured variable, which can include additional fields

```
lala.hello='world';
lala.randM=randi([0 99],5,5)
[c,doc]=couch(c,'insert doc',lala)


lala =

    hello: 'world'
    randM: [5x5 double]


c =

        url: 'http://localhost:5984'
        opt: 'insert doc'
       curl: [1x1 struct]
       info: [1x1 struct]
         db: [1x1 struct]
    all_dbs: {1x12 cell}
        doc: [1x1 struct]


doc =

     ok: 'true'
     id: '1bac260bb68846a01d573c2d4f03c431'
    rev: '1-b4c2a7cf48fe6abad2c599458820b0e1'
```

note that the response in c.curl has all the details of the exchange,

```
c.curl


ans =

       msg:
 '{"ok":true,"id":"1bac260bb68846a01d573c2d4f03c431","rev":"1-b...'
    system: 2
       ans: 0
      call: 'curl -kX POST http://localhost:5984/mydb -H "Content-
Type: ap...'
```

```
      json: [1x1 struct]
      more: [1x1 struct]
```

including the _id and _rev of the document

```
c.curl.json
```

```
ans =

     ok: 'true'
     id: '1bac260bb68846a01d573c2d4f03c431'
    rev: '1-b4c2a7cf48fe6abad2c599458820b0e1'
```

which is evertheless returned as a second output argument, doc. If only one output argument is specified that will be doc

```
doc=couch(c,'insert doc',lala);
```

see "Doc output argument" section for more on this

List all docs

as with 'all_dbs' but now for documents of a database

```
c=couch(c,'all_docs');
c.db.all_docs
```

```
ans =

    total_rows: 3
        offset: 0
          rows: [1x3 struct]
```

and each row will contain the key/value returned by that minimalist view

```
c.db.all_docs.rows(:)
```

```
ans =

3x1 struct array with fields:

    id
    key
    value
```

Get one doc

Lets get the last document inserted using its _id

```
id = c.db.all_docs.rows(end).id


id =

1bac260bb68846a01d573c2d4f03d179
```

and now retrieve the entry

```
[c,doc]=couch(c,'doc',id)


c =

        url: 'http://localhost:5984'
        opt: 'doc'
       curl: [1x1 struct]
       info: [1x1 struct]
         db: [1x1 struct]
    all_dbs: {1x12 cell}
        doc: [1x1 struct]


doc =

       id: '1bac260bb68846a01d573c2d4f03d179'
      rev: '1-b4c2a7cf48fe6abad2c599458820b0e1'
    hello: 'world'
    randM: [5x5 double]
```

doc comes out as the second output argument of couch(). If you only provide one output argument couch will assume it is doc such that you can conveniently do doc=couch(c,'doc',id).

Doc output argument

In the previous example two output arguments were used, the c structure and usual and the target answer to the URL call, the document. So now is a good time to look at the use of output arguments. Because c is used as couch db document model it is expected as input and output argument of couch(). However, it may be convenient in functions such as the previous one to simply return the document that was requested. To acomodate that in functions such as these the number of output arguments is used to decide what is returned. If there is only one output argument doc is returned. This pattern will be used for all doc centric commands, such as 'delete doc' further ahead in this tutorial.

```
doc=couch(c,'doc',id)


doc =

       id: '1bac260bb68846a01d573c2d4f03d179'
      rev: '1-b4c2a7cf48fe6abad2c599458820b0e1'
    hello: 'world'
    randM: [5x5 double]
```

JSON4MAT

This is also a good time to recognize what the JSON4MAT parsers are doing to accomodate Matlab data own formats. Notice that doc.randM is a 2D matrix but JSON has no convention to represent dimensionality even if it is reasonable to imagine it as an array of arrays (see below). JSON4MAT is doing that and more - following Matlab's loose management of datatypes it is also attempting to create numeric matrices from JSON arrays and only when that fails will it follow a the conservative route of generating an array of cells. For more information see http://json4mat.googlecode.com

```
json=urlread([c.url,'/',c.db.name,'/',id])
doc=json2mat(json)
randM=doc.randM
```

```
json =

{"_id":"1bac260bb68846a01d573c2d4f03d179","_rev":"1-
b4c2a7cf48fe6abad2c599458820b0e1","hello":"world","randM":
[[80,54,64,84,66],[89,72,52,37,20],[59,57,37,59,65],[88,2,93,87,7],
[94,44,82,93,40]]}
```

```
doc =

       id: '1bac260bb68846a01d573c2d4f03d179'
      rev: '1-b4c2a7cf48fe6abad2c599458820b0e1'
    hello: 'world'
    randM: [5x5 double]
```

```
randM =

    80    54    64    84    66
    89    72    52    37    20
    59    57    37    59    65
    88     2    93    87     7
    94    44    82    93    40
```

Delete one document

CouchDB will want to make sure you want to delete and entry by asking for both the document id and revision id. CouchDB will retrieve the rev id for you so if you don't want to make sure you are deleting a version you know then couch4mat makes it dangerously easy by letting you do it by asking only for the document id. As for 'get doc' if there is only one output argument then the deleted document will be returned so you can use doc=couch(c,'delete doc',id). If you use two output arguments note that c.db.all_docs will be updated.

```
[c,doc]=couch(c,'delete doc',id);
c.db.all_docs
```

```
ans =

    total_rows: 3
        offset: 0
          rows: [1x3 struct]
```

Delete database

After the previous examples this one will be self-explanatory. Remember that the database name is the one specified at c.db.name. The reason not to allow database deleting with the same immediate syntax as deleting a document (with the database as the third input argument) is to make it a little harder to do in order to prevent unwanted deletion of a lot of data.

```
c=couch(c,'delete db')


c =

        url: 'http://localhost:5984'
        opt: 'delete db'
       curl: [1x1 struct]
       info: [1x1 struct]
         db: [1x1 struct]
    all_dbs: {1x12 cell}
        doc: [1x1 struct]
```

Authentication

All previous examples assume an open couch deployment which is not likely to be the case when you are hosting data or applications for others. The syntax for providing autentication information is as follows:

```
c=couch('url','auth', ...
        '{username:usernamehere,password:passwordhere}')
```

The use of "auth" is similar to that of "welcome" and returns the same statistics. This command was also tested with hosted couch deployment at cloudant.com which is configured to support use of SSL. For example:

```
c=couch('https://youraccount.cloudant.com','auth', ...
        '{username:usernamehere, password:passwordhere}');
```

Note the third input argument can either be a structure or a JSON string. The latter was used in the example above which is converted automatically into a structure. You may have noted the JSON syntax employed here is not entirely legal ("" are missing bouding the username and password strings). This loose typing is supported alongside regular JSON typing by json2mat as described in the JSON4MAT manual at http://json4mat.googlecode.com/hg/html/json4mat_pub.html#6 .

FreeMat

FreeMat is a free open source clone of Matlab which is mostly compatible with regular m-code. "most" is not "all" so a few additional functions were written to deal with the differences, such as json2freemat and

cell2freemat (equivalent to json2mat and cell2mat in regular Matlab). The good news is that couch() was written to detect which m-interpreter you are using and use the right functions so you don't have to worry about this: couch() will run equally well in Matlab and FreeMat :-).

Create Admin user

Creating and admin user will finish the "admin party" and will create the _users database. Note the output argument is not c, i.e. no authentication information is included in c yet.

```
c=couch; % start from scratch
a=couch(c,'insert admin','{username:lala,password:lele}');
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
  Current
                                 Dload  Upload   Total   Spent    Left
  Speed
   0     0    0     0    0     0      0      0 --:--:--
 --:--:-- --:--:--     0 100   151  100   151
 0     0   9437      0 --:--:-- --:--:-- --:--:--
 9437{"couchdb":"Welcome","uuid":"96739cb96002ee7eb1ef6500c0b61b36","version":"1.6
{"version":"1.6.1","name":"The Apache Software Foundation"}}
```

First add authenticaion information and then check one's own entry in _users database.

```
c.auth.username='lala';
c.auth.password='lele';
c=couch(c,'db','_users');
c.db.info
```

```
ans =

                db_name: '_users'
              doc_count: 1
          doc_del_count: 0
             update_seq: 1
              purge_seq: 0
         compact_running: 'false'
               disk_size: 4194
               data_size: 2141
    instance_start_time: '1478526734612000'
    disk_format_version: 6
   committed_update_seq: 1
```

Session

The session will now contain information about the authenticated user:

```
s=couch(c,'session')
s.info
```

```
s =

         ok: 'true'
    userCtx: [1x1 struct]
       info: [1x1 struct]


ans =

          authentication_db: '_users'
    authentication_handlers: {'oauth'  'cookie'  'default'}
              authenticated: 'default'
```

Admins

for some reason I don't understand one can't get the admin user information back from the _users database. Instead one has to go to the _config/admins database

```
c=couch(c,'db','_config/admins');
c.db.info
```

```
ans =

    lala: '-
pbkdf2-94a35818d2f5e87aff098d3e19e814fc7d60548e,08cecfbf13e366...'
```

Delete Admin

Any admin can delete any admin. Since we have created only one admin, by deleting it we are left without admins and the admin party in the localhost resumes. Note removing admin is a regular DELETE with no worries about versioning.

```
a=couch(c,'delete admin','lala')
a.curl.call
```

```
a =

     url: 'http://localhost:5984'
     opt: 'delete admin'
    curl: [1x1 struct]
    info: [1x1 struct]
    auth: [1x1 struct]
      db: [1x1 struct]


ans =

curl -kX DELETE http://lala:lele@localhost:5984/_config/admins/lala
```

Published with MATLAB® R2016a