COUCH4MAT: CouchDB Matlab tbox

Table of Contents

Jonas Almeida, April 2010

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 (http://json4mat.googlecode.com).

Installation

You can download couch for the different OSs from http://couchdb.apache.org/ or from http://www.couch.io/downloads.

In addition to Matlab this tutorial also assumes you have CURL installed.

If you are running on Windows you can get it from 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
% this structure, c, will now be used as teh input argument of coucDB() to
% which we will add more arguments.
%
```

```
c =
```

```
        url: 'http://127.0.0.1:5984'
        opt: 'welcome'
    welcome: [1x1 struct]
```

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://127.0.0.1:5984'
        opt: 'create db'
    welcome: [1x1 struct]
         db: [1x1 struct]
       cURL: [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://127.0.0.1:5984'
        opt: 'all_dbs'
    welcome: [1x1 struct]
         db: [1x1 struct]
       cURL: [1x1 struct]
    all_dbs: {'lixo'  'mydb'  'testing_again'  'test_suite_db/with_slashes'  'test
```

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=rand(5);[c,doc]=couch(c,'insert doc',lala)
% note that the response in c.cURL has all the details of the exchange,
c.cURL
% including the _id and _rev of the document
c.cURL.json
% which is evertheless returned as a seccond 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
```

```
c =

        url: 'http://127.0.0.1:5984'
        opt: 'insert doc'
    welcome: [1x1 struct]
         db: [1x1 struct]
      cURL: [1x1 struct]
    all_dbs: {'lixo'  'mydb'  'testing_again'  'test_suite_db/with_slashes'  'test
```

```
doc =

     ok: 'true'
     id: 'c25d4cac6c7f04f10ffb295bef006d57'
    rev: '1-596be8da112f40ef8cd491b21ba2c053'
```

```
ans =

    call: '-kX PUT http://127.0.0.1:5984/mydb/c25d4cac6c7f04f10ffb295bef006d57 -d
     ans: 0
     msg: [1x412 char]
    json: [1x1 struct]
```

```
ans =

     ok: 'true'
     id: 'c25d4cac6c7f04f10ffb295bef006d57'
    rev: '1-596be8da112f40ef8cd491b21ba2c053'
```

List all docs

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

```
c=couch(c,'all_docs');c.db.all_docs
%and each row will contain the key/value returned by that minimalist view
c.db.all_docs.rows(:)
```

```
ans =

    total_rows: 3
        offset: 0
          rows: [3x1 struct]
```

```
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
% and now restrive the entry
[c,doc]=couch(c,'get doc',id)
% doc comes out as the second output argument of couch(). If you only
% provide one output argument coucj will assume it is doc such that you can
% convinently do doc=couch(c,'get doc',id).
```

```
id =

c25d4cac6c7f04f10ffb295bef007970


c =

        url: 'http://127.0.0.1:5984'
        opt: 'get doc'
    welcome: [1x1 struct]
         db: [1x1 struct]
       cURL: [1x1 struct]
    all_dbs: {'lixo'  'mydb'  'testing_again'  'test_suite_db/with_slashes'  'test


doc =

       id: 'c25d4cac6c7f04f10ffb295bef007970'
      rev: '1-596be8da112f40ef8cd491b21ba2c053'
    hello: 'world'
    randM: [5x5 double]
```

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 use as couch db document model it is expected as 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 fucntions 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,'get doc',id)
```

```
doc =

       id: 'c25d4cac6c7f04f10ffb295bef007970'
      rev: '1-596be8da112f40ef8cd491b21ba2c053'
    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":"c25d4cac6c7f04f10ffb295bef007970","_rev":"1-596be8da112f40ef8cd491b21ba2c0
```

```
doc =

       id: 'c25d4cac6c7f04f10ffb295bef007970'
      rev: '1-596be8da112f40ef8cd491b21ba2c053'
    hello: 'world'
    randM: [5x5 double]
```

```
randM =

    0.1622    0.6020    0.4505    0.8258    0.1066
    0.7943    0.2630    0.0838    0.5383    0.9619
    0.3112    0.6541    0.2290    0.9961    0.0046
    0.5285    0.6892    0.9133    0.0782    0.7749
    0.1656    0.7481    0.1524    0.4427    0.8173
```

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: 2
        offset: 0
          rows: [2x1 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 teh 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://127.0.0.1:5984'
        opt: 'delete db'
    welcome: [1x1 struct]
         db: [1x1 struct]
       cURL: [1x1 struct]
    all_dbs: {'lixo'  'mydb'  'testing_again'  'test_suite_db/with_slashes'  'test
```

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 .

Published with MATLAB® 7.10