

## Node.js with couch db ,Nano

nano

minimalistic couchdb driver for node.js

nano features:

- minimalistic - there is only a minimum of abstraction between you and couchdb
- pipes - proxy requests from couchdb directly to your end user
- errors - errors are proxied directly from couchdb: if you know couchdb you already know nano

[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

installation

1. install npm [<http://npmjs.org/>]
2. npm install nano

[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

getting started

to use nano you need to connect it to your couchdb install, to do that:

---

```
var nano = require('nano')('http://localhost:5984');
```

to create a new database:

---

```
nano.db.create('alice');
```

and to use it:

---

```
var alice = nano.db.use('alice');
```

in this examples we didn't specify a callback function, the absence of a callback means "do this, ignore what happens". in nano the callback function receives always three arguments:

- err - the error, if any
- body - the http response body from couchdb, if no error. json parsed body, binary for non json responses
- header - the http response header from couchdb, if no error

a simple but complete example using callbacks is:

---

```
var nano = require('nano')('http://localhost:5984');  
  
// clean up the database we created previously
```

```
nano.db.destroy('alice', function() {
  // create a new database
  nano.db.create('alice', function() {
    // specify the database we are going to use
    var alice = nano.use('alice');
    // and insert a document in it
    alice.insert({ crazy: true }, 'rabbit', function(err, body, header) {
      if (err) {
        console.log('[alice.insert] ', err.message);
        return;
      }
      console.log('you have inserted the rabbit.')
      console.log(body);
    });
  });
});
```

if you run this example(after starting couchdb) you will see:

```
you have inserted the rabbit.
{ ok: true,
  id: 'rabbit',
  rev: '1-6e4cb465d49c0368ac3946506d26335d' }
```

you can also see your document in futon [[http://localhost:5984/\\_utils](http://localhost:5984/_utils)] .

[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

database functions




[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

```
nano.db.create(name, [callback])
```

Node Tutorial With E...

search

Classic Flipcard Magazine Mosaic Sidebar Snapshot Timeslide

Node.js Intoduction	} });	
Installing Node.js on wi...	[ <a href="http://www.blogger.com/blogger.g?blogID=3141604930394944344">http://www.blogger.com/blogger.g?blogID=3141604930394944344</a> ]	
Simple server using No...	nano.db.get(name, [callback])	
Include other files and c...	get informations about name .	
Node modules	nano.db.get('alice', function(err, body) { if (!err) { console.log(body); } });	
A basic HTTP server		
Node code to connect t...	[ <a href="http://www.blogger.com/blogger.g?blogID=3141604930394944344">http://www.blogger.com/blogger.g?blogID=3141604930394944344</a> ]	
Creating a simple Test ...	nano.db.destroy(name, [callback])	
A simple web service in...	destroys name .	
CouchDB, Express and...	nano.db.destroy('alice');	

even though this examples looks sync it is an async function.

[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

## CouchDB's RESTful API

`nano.db.list([callback])`

lists all the databases in couchdb

---

```
nano.db.list(function(err, body) {
  // body is an array
  body.forEach(function(db) {
    console.log(db);
  });
});
```

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`nano.db.compact(name, [designname], [callback])`

compacts `name` , if `designname` [Send feedback](#) o compacts its views.

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`nano.db.replicate(source, target, [opts], [callback])`

replicates `source` on `target` with options `opts` . `target` has to exist, add `create_target:true` to `opts` to create it prior to replication.

---

```
nano.db.replicate('alice', 'http://admin:password@otherhost.com:5984/alice',
  { create_target:true }, function(err, body) {
    if (!err)
      console.log(body);
  });
```

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`nano.db.changes(name, [params], [callback])`

asks for the changes feed of `name` , `params` contains additions to the querystring.

---

```
nano.db.changes('alice', function(err, body) {
  if (!err)
    console.log(body);
});
```

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`nano.use(name)`

creates a scope where you operate inside `name` .

---

```
var alice = nano.use('alice');
alice.insert({ crazy: true }, 'rabbit', function(err, body) {
  // do something
});
```

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`nano.db.use(name)`

alias for `nano.use`

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`nano.db.scope(name)`

alias for `nano.use`

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

Node.js with couch db ,Nano

nano.scope(name)

alias for nano.use

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

nano.request(opts, [callback])

makes a request to couchdb, the available opts are:

- opts.db - the database name
- opts.method - the http method, defaults to get
- opts.path - the full path of the request, overrides opts.doc and opts.att
- opts.doc - the document name
- opts.att - the attachment name
- opts.content\_type - the content type of the request, default to json
- opts.body - the document or attachment body
- opts.encoding - the encoding for attachments

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

nano.relax(opts, [callback])

alias for nano.request

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

nano.dinosaur(opts, [callback])

alias for nano.request

---

```

      / _ ) roar! i'm a vegan!
    .-^--^--/ /
   /         /
  /___\___\ /___\___\

```

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

nano.config

an object containing the nano configurations, possible keys are:

- url - the couchdb url
- db - the database name

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

document functions

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

db.insert(doc, [docname], [callback])

inserts doc in the database with an optional docname .

---

```

var alice = nano.use('alice');
alice.insert({ crazy: true }, 'rabbit', function(err, body) {
  if (!err)
    console.log(body);
});

```

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`db.destroy(docname, rev, [callback])`

removes revision `rev` of `docname` from couchdb.

---

```
alice.destroy('alice', '3-66c01cdf99e84c83a9b3fe65b88db8c0', function(err, body) {
  if (!err)
    console.log(body);
});
```

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`db.get(docname, [params], [callback])`

gets `docname` from the database with optional querystring additions `params` .

---

```
alice.get('rabbit', { revs_info: true }, function(err, body) {
  if (!err)
    console.log(body);
});
```

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`db.bulk(docs, [params], [callback])`

bulk operations(update/delete/insert) on the database, refer to the [couchdb doc](http://wiki.apache.org/couchdb/http_bulk_document_api) [\[http://wiki.apache.org/couchdb/http\\_bulk\\_document\\_api\]](http://wiki.apache.org/couchdb/http_bulk_document_api) .

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`db.list([params], [callback])`

list all the docs in the database with optional querystring additions `params` .

---

```
alice.list(function(err, body) {
  if (!err) {
    body.rows.forEach(function(doc) {
      console.log(doc);
    });
  }
});
```

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`db.fetch(docnames, [params], [callback])`

bulk fetch of the database documents, `docnames` are specified as per [couchdb doc](http://wiki.apache.org/couchdb/http_bulk_document_api) [\[http://wiki.apache.org/couchdb/http\\_bulk\\_document\\_api\]](http://wiki.apache.org/couchdb/http_bulk_document_api) . additional querystring `params` can be specified, `include_doc` is always set to `true` .

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

attachments functions

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`db.attachment.insert(docname, attname, att, contenttype, [params], [callback])`

inserts an attachment `attname` to `docname` , in most cases `params.rev` is required. refer to the [doc](http://wiki.apache.org/couchdb/http_document_api) [\[http://wiki.apache.org/couchdb/http\\_document\\_api\]](http://wiki.apache.org/couchdb/http_document_api) for

more details.

---

```
var fs = require('fs');

fs.readFile('rabbit.png', function(err, data) {
  if (!err) {
    alice.attachment.insert('rabbit', 'rabbit.png', data, 'image/png',
      { rev: '12-150985a725ec88be471921a54ce91452' }, function(err, body) {
        if (!err)
          console.log(body);
      });
  }
});
```

or using pipe :

---

```
var fs = require('fs');

fs.createReadStream('rabbit.png').pipe(
  alice.attachment.insert('new', 'rab.png', {}, 'image/png')
);
```

[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

db.attachment.get(docname, attname, [params], [callback])

get docname 's attachment attname with optional querystring  
additions params .

---

```
var fs = require('fs');

alice.attachment.get('rabbit', 'rabbit.png', function(err, body) {
  if (!err) {
    fs.writeFile('rabbit.png', body);
  }
});
```

or using pipe :

---

```
var fs = require('fs');

alice.attachment.get('rabbit', 'rabbit.png').pipe(fs.createWriteStream('rabbit.png'))
```



[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

db.attachment.destroy(docname, attname, rev, [callback])

destroy attachment attname of docname 's revision rev .

---

```
alice.attachment.destroy('rabbit', 'rabbit.png',
  '1-4701d73a08ce5c2f2983bf7c9ffd3320', function(err, body) {
    if (!err)
      console.log(body);
  });
```

[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

views and design functions

[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

db.view(designname, viewname, [params], [callback])

calls a view of the specified design with optional querystring additions `params` .

---

```
alice.view('characters', 'crazy_ones', function(err, body) {
  if (!err) {
    body.rows.forEach(function(doc) {
      console.log(doc.value);
    });
  }
});
```

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

`db.updateWithHandler(designname, updatename, docname, [body], [callback])`

calls the design's update function with the specified doc in input.

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

advanced features

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

extending nano

nano is minimalistic but you can add your own features

with `nano.request(opts, callback)`

for example, to create a function to retrieve a specific revision of the `rabbit` document:

---

```
function getrabbitrev(rev, callback) {
  nano.request({ db: 'alice',
    doc: 'rabbit',
    method: 'get',
    params: { rev: rev }
  }, callback);
}

getrabbitrev('4-2e6cdc4c7e26b745c2881a24e0eece2', function(err, body) {
  if (!err) {
    console.log(body);
  }
});
```

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

pipes

you can pipe in nano like in any other stream.

for example if our `rabbit` document has an attachment with

name `picture.png` (with a picture of our white rabbit, of course!) you can pipe it to a `writable` stream

---

```
var fs = require('fs'),
    nano = require('nano')('http://127.0.0.1:5984/');
var alice = nano.use('alice');
alice.attachment.get('rabbit', 'picture.png').pipe(fs.createWriteStream('/tmp/rab
```



then open `/tmp/rabbit.png` and you will see the rabbit picture.

[\[http://www.blogger.com/blogger.g?blogID=3141604930394944344\]](http://www.blogger.com/blogger.g?blogID=3141604930394944344)

tutorials & screencasts

- screencast: couchdb and nano [<http://nodetuts.com/tutorials/30-couchdb-and-nano.html#video>]
- article: nano - a minimalistic couchdb client for nodejs [<http://writings.nunojob.com/2011/08/nano-minimalistic-couchdb-client-for-nodejs.html>]
- article: getting started with node.js and couchdb [<http://writings.nunojob.com/2011/09/getting-started-with-nodejs-and-couchdb.html>]
- article: document update handler support [<http://jackhq.tumblr.com/post/16035106690/nano-v1-2-x-document-update-handler-support-v1-2-x>]

[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

roadmap

check issues [<http://github.com/dscape/nano/issues>]

[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

tests

to run (and configure) the test suite simply:

---

```
cd nano
vi cfg/tests.js
npm install # should install ensure and async, if it doesn't install manually
npm test
```

after adding a new test you can run it individually (with verbose output) using:

---

```
nano_env=testing node tests/doc/list.js list_doc_params
```

where `list_doc_params` is the test name.

[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

contribute

everyone is welcome to contribute with patches, bugfixes and new features

1. create an issue [<http://github.com/dscape/nano/issues>] on github so the community can comment on your idea
2. fork `nano` in github
3. create a new branch `git checkout -b my_branch`
4. create tests for the changes you made
5. make sure you pass both existing and newly inserted tests
6. commit your changes
7. push to your branch `git push origin my_branch`
8. create a pull request

[<http://www.blogger.com/blogger.g?blogID=3141604930394944344>]

meta

---

—



/ \_ ) roar! i'm a vegan!  
\_ ^ ^ / /  
\_ / /  
/\_ . | | - | | cannes est superb

- code: `git clone git://github.com/dscape/nano.git`
- home: <http://github.com/dscape/nano> [<http://github.com/dscape/nano>]
- bugs: <http://github.com/dscape/nano/issues>  
[<http://github.com/dscape/nano/issues>]
- build: build status passing [<http://travis-ci.org/dscape/nano>]

Source is from <https://github.com/dscape/nano> [<https://github.com/dscape/nano>]

Posted 27th April by [screw](#)

Labels: [Node.js](#)

0 Add a comment

Comment as: Google Account

Publish

Preview