Couch::DB

Mark Overmeer (markov)
German Perl Workshop
13 May 2025, München DE

A story about

FAT

library interfaces

SQL databases vs Object Stores

SQL

- Normalization of your data into tables.
- Join table data back into the structure you need.
- Good query optimizations.
- Separate value fields.

Object Stores

- Store your data-structures (objects) "as is"
- Retrieve the data as stored, extract, but no join.
- Limited query optimizations.
- JSON (not Perl)

Objects

```
package DBObject;

sub _keep() { $_[0]->{O_keep} }
sub save()  { $DB->save($_[0]->_keep) }


package Person;
use parent 'DBObject';

sub name() { $_[0]->_keep->{name} }
sub birth(){ $_[0]->_keep->{birth} }

sub age()  { $_[0]->{P_age} ||= now – $_[0]->birth }
```

MongoDB

"Open Console" uses MongoDB clusters.

- Perl support for MongoDB stopped at v4.4 (2020)
- Current version of MongoDB is v8.0

MongoDB®

- BSON
- NASDAQ:MDB

Apache CouchDB

- Pure JSON
- Pure REST
- Objects have attachments

- Views via "design documents"
- View generating daemons in any language.

- MongoDB: DB contains collections (= tables)
- CouchDB: DB = collection

- DB in shards, which are replicated >=3 times over Nodes in a Cluster
- Graphical Cluster manager

Existing modules

```
my $db = DB::CouchDB->new(host => $host, db => $dbname);

my $doc = $db->get_doc($docname);
my $docid = $doc->{_id};

my $bar = $db->view('foo/bar', \%view_query_opts);
while(my $result = $bar->next) { … }
```

Existing modules

- AnyEvent::CouchDB - a non-blocking CouchDB client
- Couch::DB - CouchDB database client
- CouchDB::Client - Simple, correct client for CouchDB
- DB::CouchDB - A low level perl module for CouchDB
- Data::CouchDB - CouchDB document management
- Mojo::CouchDB::DB
- POE::Component::Client::CouchDB - Asynchronous CouchDB server interaction
- Store::CouchDB - a simple CouchDB driver

Existing modules

- AnyEvent::CouchDB - a non-blocking CouchDB client
- Couch::DB - CouchDB database client
- CouchDB::Client - **Simple,** correct client for CouchDB
- DB::CouchDB - A **low level** perl module for CouchDB
- Data::CouchDB - CouchDB document management
- Mojo::CouchDB::DB
- POE::Component::Client::CouchDB - Asynchronous CouchDB server interaction
- Store::CouchDB - a **simple** CouchDB driver

simple == incomplete

Existing modules

- AnyEvent::CouchDB - a **non-blocking** CouchDB client
- Couch::DB - CouchDB database client
- CouchDB::Client - Simple, correct client for CouchDB
- DB::CouchDB - A low level perl module for CouchDB
- Data::CouchDB - CouchDB document management
- Mojo::CouchDB::DB
- POE::Component::Client::CouchDB - **Asynchronous** CouchDB server interaction
- Store::CouchDB - a simple CouchDB driver

SH*T Networking

- Retries
- Redundant connections
- Time tracing
- Threading
- Paging

SH*T Networking

- Retries → **of course!**
- Redundant connections → **nice to have**
- Time tracing → **professional use**
- Threading → **better avoid**, use event driven
- Paging → **db size/app size decoupling**

SH*T JSON

- Booleans
- DateTime
- Perl Objects

- **The library user's task or library supported?**

SH*T foreign server

Expected problems:

- Protocol changes
- Backwards compatibility
- Forwards compatibility
- Remote documentation

SH*T foreign server

Expected problems:

- Protocol changes
- Backwards compatibility
- Forwards compatibility
- Remote documentation

Usable?

- Stability of developer community?
- Size of community?
- Growth potential for your application?
- Plans for mature changes?

SH*T naming

```
GET /{db}/{docid}
```

```
$conn->db($dbname)->document($docid);
```

SH*T naming

```
GET /{db}/_design/{ddoc}/_search_info/{index}
```

```
$conn->db($dbname)->design($ddoc)
   ->indexDetails($index);
```

LOVE the coding part

```
=method reshardJobState $jobid, %options
 [CouchDB API "GET /_reshard/jobs/{jobid}/state", since 2.4]

Show the resharding job status.
=cut

sub reshardJobState($%)
{   my ($self, $jobid, %args) = @_;

    $self->couch->call(GET => "/_reshard/job/$jobid/state",
        introduced => '2.4.0',
        $self->couch->_resultsConfig(\%args),
    );
}
```

SH*T the coding part

- **Completeness**
- CouchDB version 3.3.3
  - REST 137 calls
  - Some legacy naming and documentation
  - Documentation bugs
  - Incomplete/inconsistent documentation

SH*T simplifying

```
GET  /{db}/_all_docs
GET  /{db}/_local_docs
GET  /{db}/_partition/{partition}/_all_docs
POST /{db}/_all_docs
POST /{db}/_all_docs/queries
POST /{db}/_local_docs
POST /{db}/_local_docs/queries
GET  /{db}/_design/{ddoc}/_view/{view}
POST /{db}/_design/{ddoc}/_view/{view}
POST /{db}/_design/{ddoc}/_view/{view}/queries
GET  /{db}/_partition/{partition}/_design/{ddoc}/_view/{view}
```

SH*T simplifying

```
GET  /{db}/_all_docs
GET  /{db}/_local_docs
GET  /{db}/_partition/{partition}/_all_d
POST /{db}/_all_docs
POST /{db}/_all_docs/queries
POST /{db}/_local_docs
POST /{db}/_local_docs/queries
GET  /{db}/_design/{ddoc}/_view/{view}
POST /{db}/_design/{ddoc}/_view/{view}
POST /{db}/_design/{ddoc}/_view/{view}/queries
GET  /{db}/_partition/{partition}/_design/{ddoc}/_view/{view}
```

```
$db->search(\%search, %opt);
  design => $design
  local => false
  partition => $partname
  view => $view
```

SH*T documentation

Document in the database

All methods below are inherited from standard documents. Their call URI differs, but their implementation is the same. On the other hand: they add interpretation on fields which do not start with '_'.

Extends "Document in the database" in Couch::DB::Document.

$obj->appendTo($doc, %options)

```
[CouchDB API "COPY /{db}/_design/{ddoc}"]
```

$obj->cloneInto($doc, %options)

```
[CouchDB API "COPY /{db}/_design/{ddoc}"]
```

$obj->create(\%data, %options)

Create a new design document. Design documents do not use generated ids, so: you have to have specified one with new(id). Therefore, this method is equivalent to update().

```
-Option--Defined in         --Default
 batch   Couch::DB::Document  new(batch)
```

batch => BOOLEAN

$obj->delete(%options)

```
[CouchDB API "DELETE /{db}/_design/{ddoc}"]
```

$obj->details(%options)

```
[CouchDB API "GET /{db}/_design/{ddoc}/_info", UNTESTED]
```

SH*T documentation

`GET /{db}/_shards/{docid}` Returns the specific shard in which a document is stored	DONE	`$cluster->shardsForDoc($doc, %options)`
`POST /{db}/_sync_shards` Trigger a synchronization of all shard replicas in the database	DONE	`$cluster->syncShards($db, %options)`
`POST /{db}/_view_cleanup` Removes view files that are not used by any design document	UNTESTED	`$db->purgeUnusedViews(%options)`
`COPY /{db}/{docid}` Copies the document within the same database	PARTIAL PARTIAL	`$doc->cloneInto($doc, %options)` `$doc->appendTo($doc, %options)`
`DELETE /{db}/{docid}` Deletes the document	DONE	`$doc->delete(%options)`
`GET /{db}/{docid}` Returns the document	DONE	`$doc->get([\%flags, %options])`
`HEAD /{db}/{docid}` Returns bare information in the HTTP Headers for the document	DONE	`$doc->exists(%option)`
`PUT /{db}/{docid}` Creates a new document or new version of an existing document	DONE	`$doc->update(\%data, %options)`
`DELETE /{db}/{docid}/{attname}` Deletes an attachment of a document	UNTESTED	`$doc->attDelete($name, %options)`
`GET /{db}/{docid}/{attname}` Gets the attachment of a document	UNTESTED	`$doc->attLoad($name, %options)`
`HEAD /{db}/{docid}/{attname}` Returns bare information in the HTTP Headers for the attachment	UNTESTED	`$doc->attExists($name, %options)`
`PUT /{db}/{docid}/{attname}` Adds an attachment of a document	UNTESTED	`$doc->attSave($name, $data, %options)`

SH*T documentation

Couch::DB use	impl status	CouchDB API "stable"
$couch->requestUUIDs($count, %options)	DONE	GET /_uuids
$couch->searchAnalyze(%options)	UNTESTED	POST /_search_analyze
$client->activeTasks(%options)	DONE	GET /_active_tasks
$client->clusterNodes(%options)	UNTESTED	GET /_membership
$client->databaseInfo(%options)	DONE	GET /_dbs_info
	DONE	POST /_dbs_info
$client->databaseNames(%options)	DONE	GET /_all_dbs
$client->dbUpdates(\%feed, %options)	UNTESTED	GET /_db_updates
$client->login(%options)	UNTESTED	POST /_session
$client->logout(%options)	UNTESTED	DELETE /_session
$client->nodeName($name, %options)	UNTESTED	GET /_node/{node-name}
$client->replicate(\%rules, %options)	UNTESTED	POST /_replicate
$client->replicationDoc($doc|$docid, %options)	UNTESTED	GET /_scheduler/docs/{replicator_db}/{docid}
$client->replicationDocs(%options)	UNTESTED	GET /_scheduler/docs
	UNTESTED	GET /_scheduler/docs/{replicator_db}
$client->replicationJobs(%options)	UNTESTED	GET /_scheduler/jobs
$client->serverInfo(%options)	DONE	GET /
$client->serverStatus(%options)	UNTESTED	GET /_up
$client->session(%options)	UNTESTED	GET /_session
$cluster->clusterSetup($config, %options)	UNTESTED	POST /_cluster_setup
$cluster->clusterState(%options)	DONE	GET /_cluster_setup
$cluster->reshardJob($jobid, %options)	UNTESTED	GET /_reshard/jobs/{jobid}
$cluster->reshardJobChange($jobid, %options)	UNTESTED	PUT /_reshard/jobs/{jobid}/state
$cluster->reshardJobRemove($jobid, %options)	UNTESTED	DELETE /_reshard/jobs/{jobid}
$cluster->reshardJobState($jobid, %options)	UNTESTED	GET /_reshard/jobs/{jobid}/state
$cluster->reshardJobs(%options)	DONE	GET /_reshard/jobs
$cluster->reshardStart(\%create, %options)	UNTESTED	POST /_reshard/jobs
$cluster->reshardStatus(%options)	DONE	GET /_reshard
	DONE	GET /_reshard/state

Now I want to use it!

(and you too)