SHORTCUT

CouchDB

Johan Sørensen

This is CouchDB

A wonderful idea

This is CouchDB

A wonderful idea
… Based on an awful thing

This is CouchDB

Lotus Notes

This is CouchDB

Damien Katz wanted to recreate Lotus Notes

This is CouchDB

Damien Katz wanted to recreate Lotus Notes
– But with good API

This is CouchDB

Damien Katz wanted to recreate Lotus Notes
– But with good API
– A good data format

This is CouchDB

Damien Katz wanted to recreate Lotus Notes

– But with good API

– A good data format

– Open source

This is CouchDB

Damien Katz wanted to recreate Lotus Notes

– But with good API

– A good data format

– Open source

– Without wasting effort on making it a silly groupware/email/calendaring application

This is CouchDB

– Apache licensed
– Currently under Apache Incubation
– Self-funded for two years by Damien Katz
– Now sponsored by IBM

A document database

With a RESTful API

With a RESTful API

GET, POST, PUT & DELETE

With a RESTful API

GET, POST, PUT & DELETE

– Allows us to use existing infrastructure

With a RESTful API

GET, POST, PUT & DELETE

– Allows us to use existing infrastructure
Load balencers

With a RESTful API

GET, POST, PUT & DELETE

– Allows us to use existing infrastructure
Load balencers, proxies etc

With versioning

With replication

In a sensible format

In a sensible format

(That means “not XML”)

In a sensible format

JSON

In a sensible format

JSON

– Human readable

In a sensible format

JSON

– Human readable

– Machine readable

In a sensible format

JSON

– Human readable
– Machine readable

```
{ “foo”: “bar” }
```

With querying

With querying

In Javascript

With querying

In Javascript

```
if (doc.title == “couchdb”)
```

```
{ “title”: “couchdb” }
```

Document oriented applications

Document oriented applications

In real life:

- A document is a piece of paper

Document oriented applications

In real life:

– A document is a piece of paper

– With “stuff” on it

Document oriented applications

In real life:

- A document is a piece of paper
- With “stuff” on it
- Everyone can read and understand “stuff”

Who deals with “stuff”?

Most applications

Email

Email, notes

Email, notes, todos

Email, notes, todos, wikis

Email, notes, todos, wikis, forums

Email, notes, todos, wikis, forums, blogs

Email, notes, todos, wikis, forums, blogs, bug reports

Email, notes, todos, wikis, forums, blogs, bug reports, CRM

Email, notes, todos, wikis, forums, blogs, bug reports, CRM, CMS, expense tracking

Email, notes, todos, wikis, forums, blogs, bug reports,
CRM, CMS, expense tracking, project management

Email, notes, todos, wikis, forums, blogs, bug reports, CRM, CMS, expense tracking, project management, asset tracking

Email, notes, todos, wikis, forums, blogs, bug reports, CRM, CMS, expense tracking, project management, asset tracking, TPS reports

Email, notes, todos, wikis, forums, blogs, bug reports, CRM, CMS, expense tracking, project management, asset tracking,TPS reports,TPS report coversheets

Email, notes, to blogs, bug reports,
CRM, CMS, exp ject management,
asset tracking, TPS reports, TPS report coversheets

Most applications!

A CouchDB Document

A CouchDB Document

```
{
  “_id”: “4B46AFFF0233871EAFC9EA5680E4C90B”,
  “_rev”: 1778502017,
  “title”: “My blog post”,
  “body”: “so last night...”
}
```

Every document

– Has a _rev attribute

Every document

– Has a _rev attribute
– And an _id attribute

Every document

– Has a _rev attribute
– And an _id attribute (GUUID, or custom)

Every document

- – Has a _rev attribute
- – And an _id attribute (GUUID, or custom)
- – n number of arbitrary key+value JSON data

Creating a document

POST http://localhost:5984/blog-posts

```
{
  “title”: “My blog post”,
  “body”: “so last night...”
}
```

Creating a document

POST http://localhost:5984/blog-posts

```
{
  “title”: “My blog post”,
  “body”: “so last night...”
}
```

201 Created

Creating a document with a custom ID

PUT http://localhost:5984/blog-posts/mycustomid

```
{
  “title”: “My blog post”,
  “body”: “so last night...”
}
```

Updating a document

PUT http://localhost:5984/blog-posts/mycustomid

```
{
  “_rev”: 1778502017,
  “title”: “My updated blog post”,
  “body”: “so last night, and it was like you know...”
}
```

GET a document

GET http://localhost:5984/blog-posts/docid

GET a specific revision

GET http://localhost:5984/blog-posts/docid?rev=9467819C

GET all revisions

GET http://localhost:5984/blog-posts/docid?revs=true

Attachments

```
{
  _id:“attachment_doc”,
  _attachments:
  {
    “foo.txt”:
    {
      “type”:“base64”,
      “data”: “VGhpcyBpcyBhIGJhc2U2NCBlbmNvZGVkIHRleHQ=”
    }
  }
}
```

Attachments

GET /database/attachment_doc

```
{“_id”:“attachment_doc”,
  “_rev”:1589456116,
  “_attachments”:
  {
    “foo.txt”:
    {
      “stub”:true,
      “type”:“base64”,
      “length”:29
    }
}}
```

GET /database/attachment_doc/foo.txt

Attachments

… Still has a fluctuating API

Javascript views

Javascript views

Essentially "stored procedures" in javascript

Javascript views

Essentially “stored procedures” in javascript

– Give me all docs with a “title” attribute of “bacon”

Javascript views

Essentially “stored procedures” in javascript

– Give me all docs with a “title” attribute of “bacon”
– Give me all docs where the “title” atribute matches /(chunky|crispy)bacon.+/

Javascript views

An anonymous function, that map()’s the results

```
function(doc) {
  if (doc.foo == “baz”) {
    map(null, doc);
  }
}
```

Javascript views

A temporary, on-the-fly, view

POST /mydatabase/_temp_view

```
function(doc) {
  if (doc.foo == “baz”) {
    map(null, doc);
  }
}
```

Javascript views

Persistant views

```
{
  “_id”:“_design/company”,
  “_rev”:“12345”,
  “language”: “text/javascript”,
  “views”:
  {
    “all”: “function(doc) { if (doc.Type == ‘customer’)  map(null, doc) }”
    “by_lastname”: “function(doc) { if (doc.Type == ‘customer’)
map(doc.LastName, doc) }”
  }
}
```

Javascript views

Persistant views

GET /somedatabase/_view/company/all

Javascript views

Persistant views

Updates a view index everytime a document is created or updated

Bonus features

Bonus features

– Fulltext indexing + search

Bonus features

– Fulltext indexing + search
– Lucene

Bonus features

– Fulltext indexing + search
– Lucene

WIP:
– Sphinx

Bonus features

– Fulltext indexing + search
– Lucene

WIP:
– Sphinx
– Xapian (by me - http://gitorious.org/projects/couchdb-xapian)

CouchDB & Ruby

CouchDB & Ruby

– Net::HTTP + JSON gem

CouchDB & Ruby

– Net::HTTP + JSON gem

– CouchObject (gitorious.org/projects/couchobject)

CouchObject

```
>> db = CouchObject::Database.open(“http://localhost:5984/test“)
# => #<CouchObject::Database:0x116b09c @server=#<CouchObject::Server:
0x116b038 ...>
>> response = db.get(“someid”)
# => #<CouchObject::Response:0x1167820 @response=#<Net::HTTPOK 200 OK
readbody=true>, ...>
>> doc = response.to_document
# <CouchObject::Document:0x1165700
#  @attributes={“title”=>”a document”, “description”=>”it’s schema-free”},
#  @id=”someid”,
#  @revision=”2501615157”>
#
>> doc.id
# => “someid”
>> doc.title
# => “a document”
>> doc.title = “a new title”
# => “a new title”
>> doc.save(db)
```

CouchObject::Persistable

CouchObject::Persistable

```
class Bike
  include CouchObject::Persistable

  def initialize
    @wheels = 2
  end
  attr_accessor :wheels

  def to_couch
    {:wheels => @wheels}
  end

  def self.from_couch(attributes)
    bike = new
    bike.wheels = attributes[“wheels”]
    bike
  end
end
```

CouchObject::Persistable

– Adds magic methods to objects (#find, #save etc)

CouchObject::Persistable

- Adds magic methods to objects (#get_by_id, #save etc)
- Serialization methods optional, if following conventions

CouchObject::Persistable

- Adds magic methods to objects (#get_by_id, #save etc)
- Serialization methods optional, if following conventions
- Automagic timestamping

CouchObject::Persistable

- Adds magic methods to objects (#get_by_id, #save etc)
- Serialization methods optional, if following conventions
- Automagic timestamping
- has_many, has_one & belongs_to style relations

CouchObject::Persistable

– Adds magic methods to objects (#get_by_id, #save etc)

– Serialization methods optional, if following conventions

– Automagic timestamping

– has_many, has_one & belongs_to style relations

Where the related part is either stored in the same document, or in another document

CouchObject

http://gitorious.org/projects/couchobject

Up-to-date gem Coming Soon™

CouchDB Future

– Live compaction (committed yesterday)

CouchDB Future

– Live compaction (committed yesterday)

– Working reduce implementation

Reduce part of map/reduce

mapped docs

tag	count
ruby	1
rails	1
ruby	1
java	0

reduced

tag	reduced
ruby	2
rails	1
java	0

CouchDB Future

- Live compaction (committed today)
- Working reduce implementation
- Access control

CouchDB Future

– Live compaction (committed today)
– Working reduce implementation
– Access control
– Easier sharding/partioning

CouchDB demo

Questions?

Johan Sørensen
johan@shortcut.no

Thank you

Johan Sørensen
johan@shortcut.no