

# Breizhcamp - Nosql - Ruby

Nicolas Ledez  
from-breizhcamp@ledez.net

17 Juin 2011



# Nicolas Ledez

<me>

# Nicolas Ledez



Depuis bientôt 5 ans

# Chef de projet technique

# Nicolas Ledez



[http ://www.breizhcamp.org/](http://www.breizhcamp.org/)  
Cloud et NoSQL

# Nicolas Ledez



**Rennes on Rails**

Happy coding !



[http ://www.rennesonrails.com/](http://www.rennesonrails.com/)  
Coding Dojo & Confs

# Nicolas Ledez



[http ://www.granit.org/](http://www.granit.org/)  
Forum Graphotech Cloud

# Nicolas Ledez

`</me>`



# Plan

- 1 Introduction
- 2 Ruby On Rails
- 3 NoSQL
- 4 Conclusion

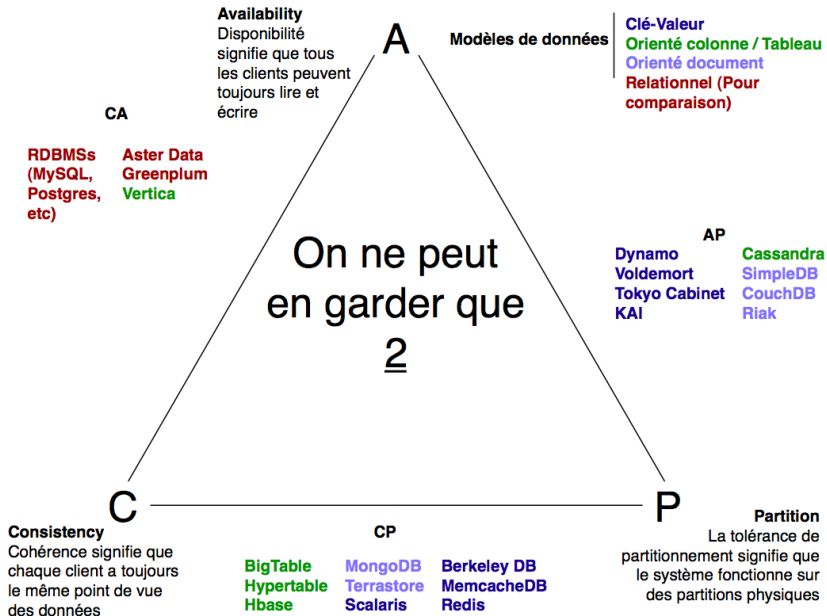
# Qui connaît ?

- NoSQL
  - SQL
  - Clé/valeur
  - Document
- Ruby On Rails
  - Ruby
  - Rails

# NoSQL

## Not Only SQL

# Choix du moteur



- Ruby
  - Plusieurs VM Ruby (MRI, JRuby, MacRuby, Rubinius)
  - 24,898 gems (en juin 2011)
  - +185 000 projets Ruby sur Github
- Rails
  - DRY
  - Convention over Configuration
  - MVC, migrations, Active Record, etc

```
$ cat config/locales/fr.yml
```

```
fr:
```

```
  listing_tickets: "Liste des tickets"
```

```
  name: "Nom"
```

```
  unknown: "Inconnu"
```

```
  back: "Retour"
```

```
  are_you_sure: "Etes-vous sur ?"
```

```
  editing_ticket: "Edition du ticket"
```

```
$ grep listing_tickets app/views/tickets/index.html
```

```
<h1><%= t('listing_tickets') %></h1>
```

# Ruby On Rails - generators

```
$ rails generate model ticket name:string  
description:text  
$ cat db/migrate/20110602142024_create_tickets.rb  
class CreateTickets < ActiveRecord::Migration  
  def change  
    create_table :tickets do |t|  
      t.string :name  
      t.text :description  
  
      t.timestamps  
    end  
  end  
end
```

# Ruby On Rails - Active Record

```
$ cat app/models/ticket.rb
```

```
class Ticket < ActiveRecord::Base
  validates_presence_of :name
  validates_presence_of :status
end
```

```
$ rake db:migrate
```

```
-- create_table("tickets", {:force=>true})
-> 0.0696s
```

DB -> Model -> Controller -> View



# Redis

- Manipulation de clés / valeurs
- Commandes simples
- Valeurs : string, list, set, sorted set, hashes
- Expiration des clés possible
- Utilisation : cache, session, compteurs, queue

# Redis exemples

## Clé/valeur

**SET** key "value"

**GET** key

**DEL** key

**INCR** key

**EXPIRE** key 5

**EXPIREAT** key

<timestp>

**TTL** key

## List

**LPUSH** key 1

**LRANGE** key 0 -1

**RPOP** key

**LLEN** key

**LTRIM** key 0 1

## Sets

**SADD** key 1

**SMEMBERS** key

**SISMEMBER** key

2

**SRANDMEMBER**

key

**SREM** key 3

# Redis avec Ruby On Rails - backend

## Ajout de "gem 'redis'" dans le Gemfile

```
$ cat config/initializers/i18n_backend.rb
```

```
I18n.backend = I18n::Backend::KeyValue.new(  
  Redis.new)
```

```
$ cat config/initializers/i18n_backend.rb
```

```
TRANSLATION_STORE = Redis.new
```

```
I18n.backend = I18n::Backend::Chain.new(  
  I18n::Backend::KeyValue.new(TRANSLATION_STORE),  
  I18n.backend)
```

# Redis avec Ruby On Rails - frontend

```
$ cat app/controllers/translations_controller.rb

class TranslationsController < ApplicationController
  def index
    @translations = TRANSLATION_STORE
  end

  def create
    I18n.backend.store_translations(params[:locale],
    {params[:key] => params[:value]}, :escape => false)
    redirect_to translations_url,
    :notice => "Added translations"
  end
end
```

# Redis avec Ruby On Rails - hack ROR 3.1 rc1 1/3

```
$ cat config/environment.rb
```

```
# Load the rails application
```

```
require File.expand_path('../application', __FILE__)
```

```
module I18n
```

```
  module Backend
```

```
    class KeyValue
```

```
      module Implementation
```

```
        def store_translations(locale, data, options = {})
```

```
          #@store[key] = ActiveSupport::JSON.encode(value) unless
```

```
            @store[key] = ActiveSupport::JSON.encode([value]) unless
```

```
        end
```

```
      end
```

```
    end
```

```
  end
```

```
end
```

```
end
```

# Redis avec Ruby On Rails - hack ROR 3.1 rc1 2/3

```
module I18n
  module Backend
    class KeyValue
      module Implementation
        protected
        def lookup(locale, key, scope = [], options = {})
          #value = ActiveSupport::JSON.decode(value) if value
          value = ActiveSupport::JSON.decode(value)[0] if value
        end
      end
    end
  end
end

# Initialize the rails application
Tickets::Application.initialize!
```

# Redis avec Ruby On Rails - hack ROR 3.1 rc1 3/3

```
module I18n
  module Backend
    class KeyValue
      module Implementation
        def store_translations(locale, data, options = {})
          escape = options.fetch(:escape, true)
          flatten_translations(locale, data, escape, @subtrees).each do |key, value|
            key = "#{locale}.#{key}"

            case value
            when Hash
              if @subtrees && (old_value = @store[key])
                old_value = ActiveSupport::JSON.decode(old_value)
                value = old_value.deep_symbolize_keys.deep_merge!(value) if old_value.is_a?(Hash)
              end
            when Proc
              raise "Key-value stores cannot handle procs"
            end

            @@store[key] = ActiveSupport::JSON.encode(value) unless value.is_a?(Symbol)
            @store[key] = ActiveSupport::JSON.encode([value]) unless value.is_a?(Symbol)
          end
        end

        protected

        def lookup(locale, key, scope = [], options = {})
          key = normalize_flat_keys(locale, key, scope, options[:separator])
          value = @store["#{locale}.#{key}"]
          #value = ActiveSupport::JSON.decode(value) if value
          value = ActiveSupport::JSON.decode(value)[0] if value
          value.is_a?(Hash) ? value.deep_symbolize_keys : value
        end
      end
    end
  end
end
```

# MongoDB

- Base orientée documents (BSON)
- Un système de requêtes évolué
- Scalable
- Hautes performances
- Sans schéma



## MongoDB exemples 1/2

```
show dbs  
use db_name  
show collections
```

```
db.foo.find();
```

```
db.foo.save({ name : "sara" });
```

```
person = db.people.findOne( { name : "sara" } );  
person.city = "New York";  
db.people.save( person );
```

## MongoDB exemples 2/2

```
db.foo.drop()
```

```
db.foo.remove()
```

```
db.foo.remove( { name : "sara" } )
```

```
db.foo.getIndexKeys()
```

```
db.foo.ensureIndex({ _field_ : 1 })
```

## Avec Rails - config

### config/initializers/mongo.rb

```
MongoMapper.connection = Mongo::Connection.new('localhost', 27
MongoMapper.database = "#myapp-#{Rails.env}"

if defined?(PhusionPassenger)
  PhusionPassenger.on_event(:starting_worker_process) do |for
    MongoMapper.connection.connect if forked
  end
end
```

# Avec Rails - model

app/models/note.rb

```
class Note
  include Mongomapper::Document

  key :title, String, :required => true
  key :body, String
end
```

# Avec Rails - à la place d'Active Record

- Pas de migration du schéma de la BDD
- Tolérance du modèle

# Conclusion

Conclusion

# Questions

Questions ?

# Sources

- <http://blog.nahurst.com/visual-guide-to-nosql-systems>
- <http://nosql-database.org/>
- <http://www.camilleroux.com/2010/08/16/pourquoi-ruby-on-rails-est-genial-1-sur-2/>
- <http://railscasts.com/episodes/256-i18n-backends>
- <http://www.slideshare.net/karmi/redis-the-ak47-of-postrelational-databases>



# Licence



CC BY-NC-SA