DECASTORE

TECHNICAL ROAD MAP

by Serguei Cambour
February 2018

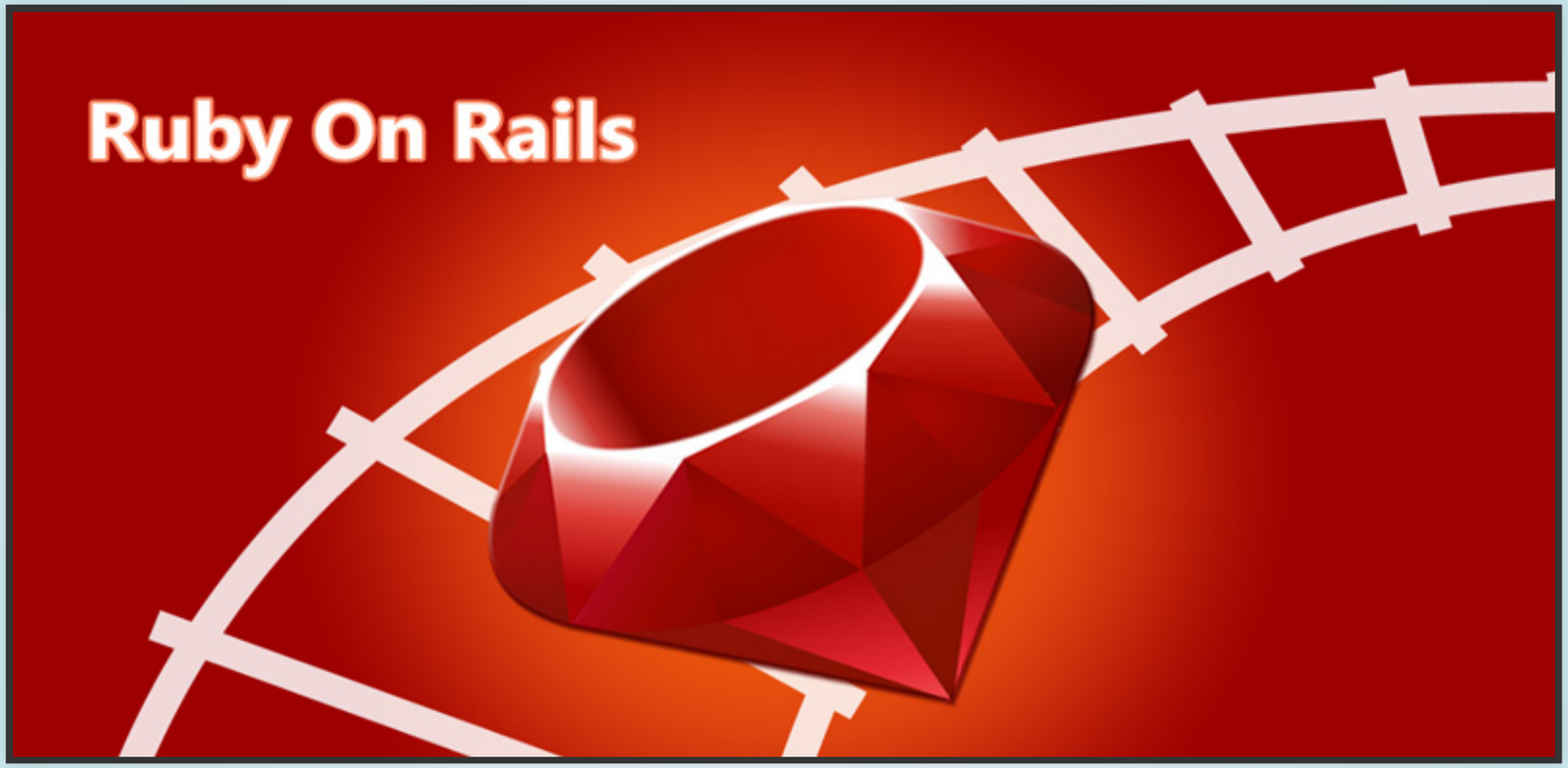

FRONT END
BACK END
comic.browserling.com

COMMENT CHOISIR PARMI LES FRAMEWORKS EXISTANTS ?


2017 TOP Backend
Frameworks
Express.js
node
laravel
python
django
RAILS

LE CHOIX EST VRAIMENT TROOOOOP VASTE ...

THE BACK-END WINNER IS

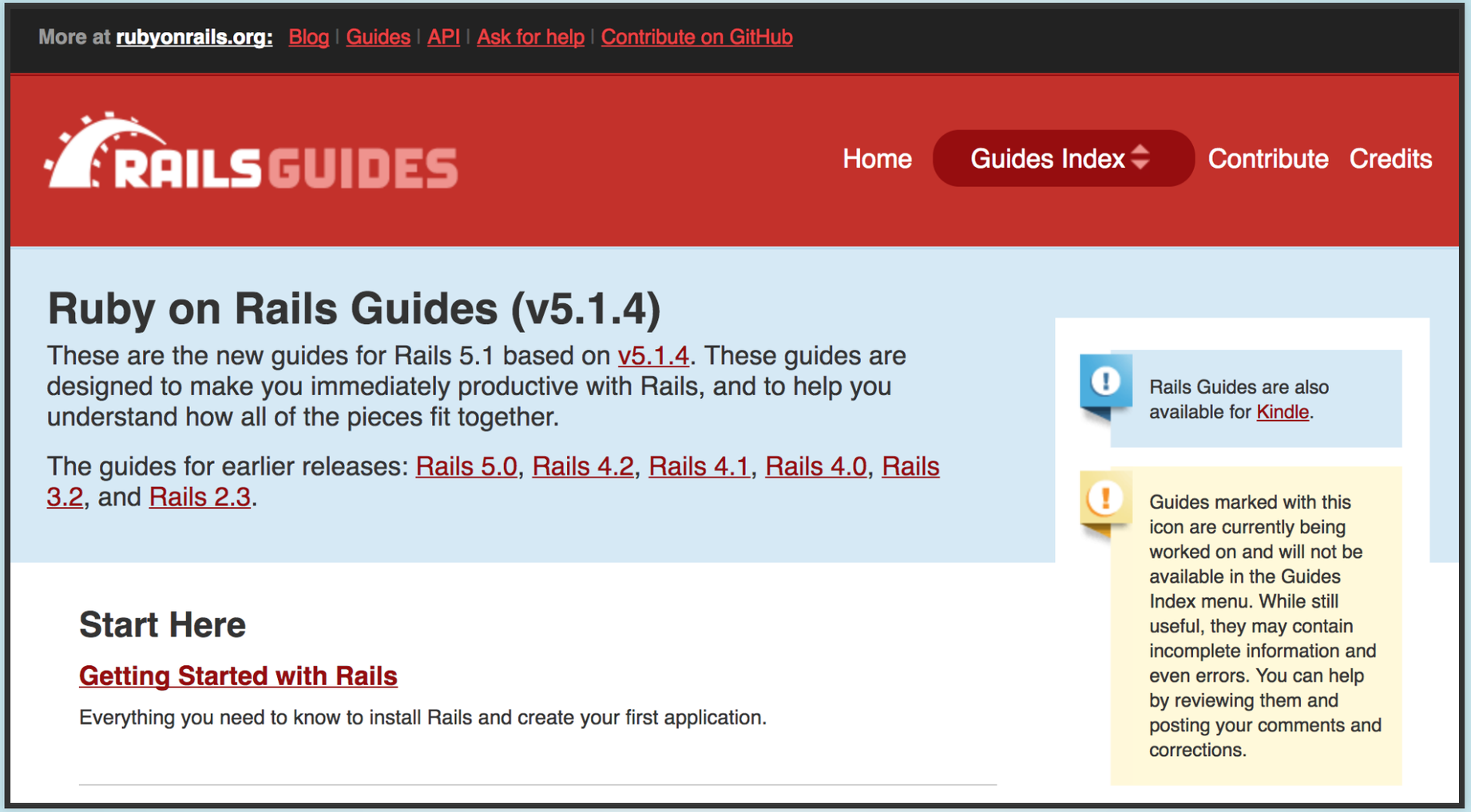


Plus de 4,500 personnes ont déjà contribué au code de Ruby on Rails.

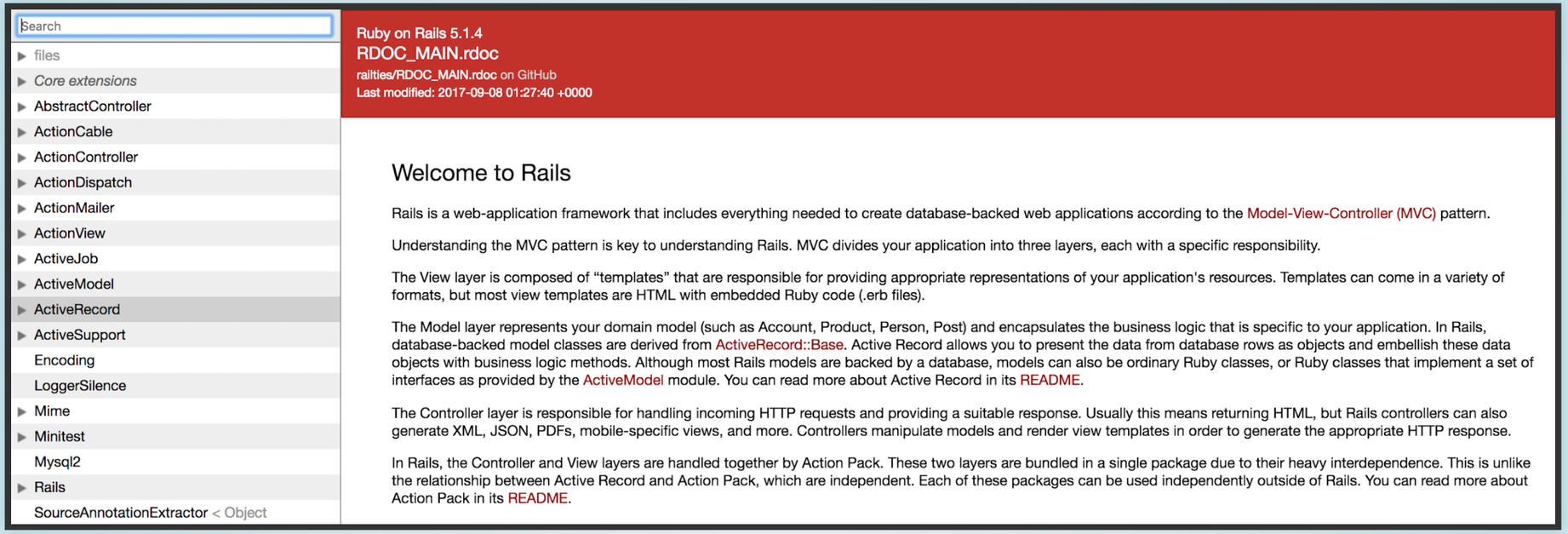
- Server-side web application framework écrit en **Ruby**
- Crée par **David Heinemeier Hansson** (aka *DHH*)
- Première version: le **13 décembre 2005**
- Dernière version: **5.1.5**, sortie le **14 février 2018**

Ruby is simple in appearance, but is very complex inside, just like our human body.

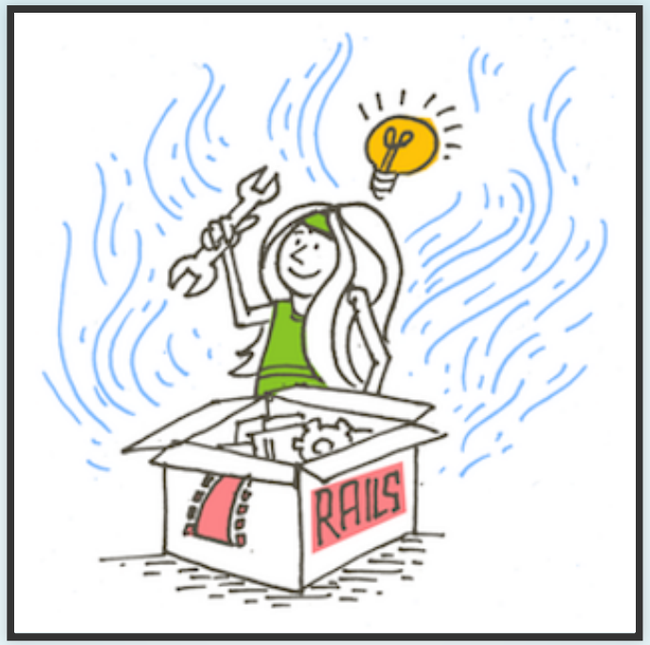
- une très bonne documentation
- pas de XML(!)
- convention over configuration
- utilise Ruby
- énorme répo de gems (petites librairies) sur RubyGems -a package management framework pour Ruby.



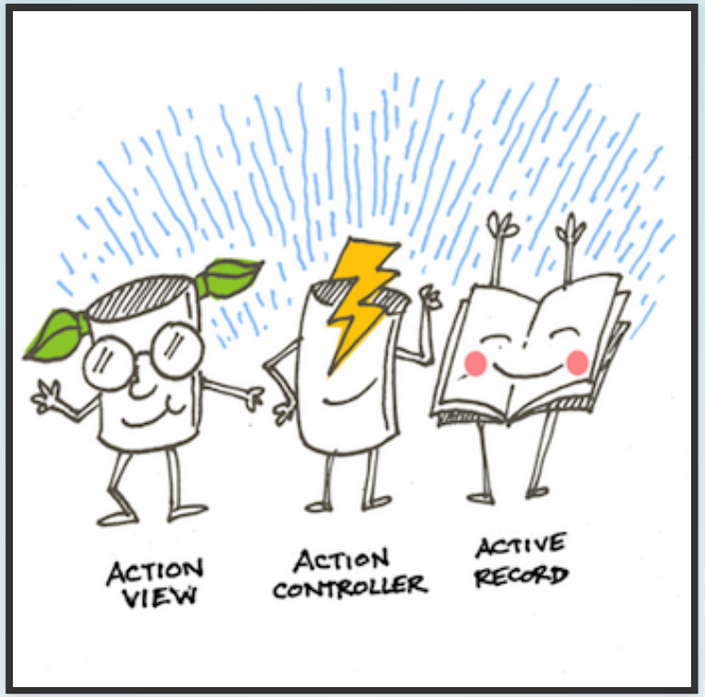
Rails Guides Guides RoR pour dévenir vite productif avec Rails



Rails API documentation pour plus d’infos sur modules, classes, méthodes.



La colonne vertébrale d’une application Rails est le pattern **Model-View-Controller** qui organize l’application en 3 couches principales:



- Le **Model** est géré principalement par Active Record
- Le **Controller** est géré par Action Controller
- La **View** par Action View, présenté ensemble comme Action Pack

A part ces 3 *amigos*, Rails est encore livré avec:

- **Action Mailer**, pour envoyer des mails en HTML ou plein text.
- **Action Cable** pour fournir “live updates” des pages via WebSockets
- **Active Support** améliore Ruby standard library avec méthodes et classes supplémentaires.

Mais il y a encore:

- **IRB** (Interactive Ruby shell) pour exécuter Ruby code dans le Terminal
- **Rails Console** pour ‘jouer’ avec code Rails, requêtes SQL dans le Terminal
- **Database Migrations**: pour mettre à jour, modifier votre schema et tables BDD en Ruby

Pas de **XML**, votre configuration vit dans des fichiers **YAML**:

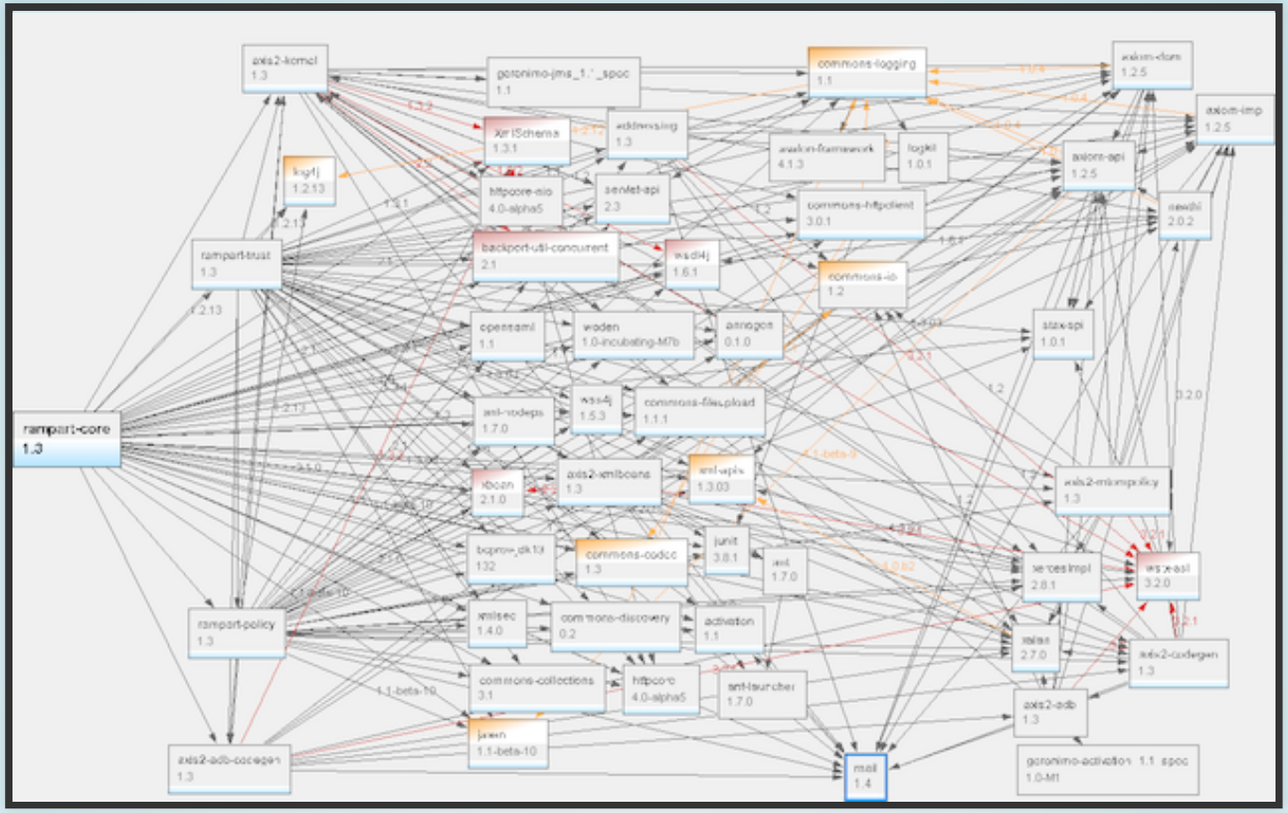
```yaml
# database.yml
default: &default
  adapter: postgresql
  encoding: unicode
  user: postgres
  password:
  pool: 5

development:
  <<: *default
  database: decastore_development
test:
  <<: *default
  database: decastore_test
production:
  <<: *default
```

Les paramètres de votre application sont dans le fichier `application.yml`

```
# application.yml provided by Figaro gem
development:
  oauth_site: 'https://preprod.idpdecathlon.oxylane.com/as/author
  client_id: 'decastore'
  client_secret: '2e53Wu3PLmUqW1CfzStdsJUHM2LltvwFPU6gogDPe5w19eT
  proxy_url: 'http://gateway.zscaler.net'
  proxy_port: '80'

test:
  oauth_site: 'https://preprod.idpdecathlon.oxylane.com/as/author
  client_id: 'decastore'
  client_secret: '2e53Wu3PLmUqW1CfzStdsJUHM2LltvwFPU6gogDPe5w19eT
  proxy_url: 'http://gateway.zscaler.net'
  proxy_port: '80'
```



COMMENT SONT GÉRÉES LES DÉPENDANCES ?


STAY
COOL
JOE
COOL

Bundler s'en occupe! Toutes les dépendances sont définiés dans le fichier `Gemfile`:

```
# Gemfile
ruby '2.4.0'

gem 'rails', '~> 5.1.4'
gem 'pg',                           '~> 0.21.0'
gem 'figaro',                       '~> 1.1.1'
gem 'jwt',                          '~> 2.1'
gem 'geocoder',                     '~> 1.4', '>= 1.4.4'
gem 'active_model_serializers',     '~> 0.10.6'
gem 'rest-client',                  '~> 2.0', '>= 2.0.2'
.....
```

Chaque gem est declaré dans son groupe d’environnement:

```
# Gemfile
group :development, :test do
  gem 'ffaker',      '~> 2.1.0'
end

group :test do
  gem 'database_cleaner'
end

group :production do
  gem 'rails_12factor'
```

Vous pouvez avoir autant de versions Ruby installées que vous voulez sur votre poste. Ruby Version Manager (RVM) le détecte via `Gemfile` automatiquement.

RAILS CONSOLE À VOTRE SERVICE:

```
➜  todo_app git:(master) ✗ rails c
Loading development environment (Rails 5.1.4)
2.4.0 :001 > Article.count
   (0.1ms)  SELECT COUNT(*) FROM "articles"
 => 10
2.4.0 :002 > Article.first
  Article Load (0.2ms)  SELECT  "articles".* FROM "articles" ORDER BY "articles"."id" ASC LIMIT ?  [["LIMIT", 1]]
 => #<Article id: 1, name: "Synergistic Rubber Bag", created_at: "2018-02-12 20:30:37", updated_at: "2018-02-12 20:30:37
">
2.4.0 :003 > article = Article.first
  Article Load (0.2ms)  SELECT  "articles".* FROM "articles" ORDER BY "articles"."id" ASC LIMIT ?  [["LIMIT", 1]]
 => #<Article id: 1, name: "Synergistic Rubber Bag", created_at: "2018-02-12 20:30:37", updated_at: "2018-02-12 20:30:37
">
2.4.0 :004 > article.name = "Awesome item"
 => "Awesome item"
2.4.0 :005 > article.save
   (0.1ms)  begin transaction
  SQL (0.5ms)  UPDATE "articles" SET "name" = ?, "updated_at" = ? WHERE "articles"."id" = ?  [["name", "Awesome item"],
["updated_at", "2018-02-12 20:33:00.826497"], ["id", 1]]
   (1.8ms)  commit transaction
 => true
2.4.0 :006 > article
 => #<Article id: 1, name: "Awesome item", created_at: "2018-02-12 20:30:37", updated_at: "2018-02-12 20:33:00">
2.4.0 :007 >
```

Avec `rails c --sandbox` toute modification dans la BDD sera annulée à la sortie.

Rails est installé avec son propre serveur d’application - **Puma**.

```
➜  todo_app git:(master) ✗ rails s
=> Booting Puma
=> Rails 5.1.4 application starting in development
=> Run `rails server -h` for more startup options
Puma starting in single mode...
* Version 3.11.0 (ruby 2.4.0-p0), codename: Love Song
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://0.0.0.0:3000
Use Ctrl-C to stop
Started GET "/" for 127.0.0.1 at 2018-02-12 21:40:21 +0100
   (0.2ms)  SELECT "schema_migrations"."version" FROM "schema_migrations" ORDER BY "schema_migrations"."version" ASC
Processing by ArticlesController#index as HTML
  Rendering articles/index.html.erb within layouts/application
  Article Load (0.4ms)  SELECT "articles".* FROM "articles"
  Rendered articles/index.html.erb within layouts/application (14.1ms)
Completed 200 OK in 243ms (Views: 233.4ms | ActiveRecord: 0.9ms)


Started GET "/articles/1" for 127.0.0.1 at 2018-02-12 21:40:28 +0100
Processing by ArticlesController#show as HTML
  Parameters: {"id"=>"1"}
  Article Load (0.2ms)  SELECT  "articles".* FROM "articles" WHERE "articles"."id" = ? LIMIT ?  [["id", 1], ["LIMIT", 1]
]
  Rendering articles/show.html.erb within layouts/application
  Rendered articles/show.html.erb within layouts/application (0.9ms)
Completed 200 OK in 29ms (Views: 21.4ms | ActiveRecord: 0.2ms)
```

QUI UTILISE RUBY ON RAILS ?

- GitHub
- Airbnb
- Hulu
- Twitch
- Basecamp

et beaucoup d’autres… Ceux-ci ne sont que des grands.

The front-end winner is

A framework for creating ambitious web applications

- basé sur le pattern Model–view–viewmodel (MVVM)
- Créé par **Yehuda Katz**
- Première release: le 8 décember 2011
- Dernière version: **3.0**, sortie le **14 février 2018**

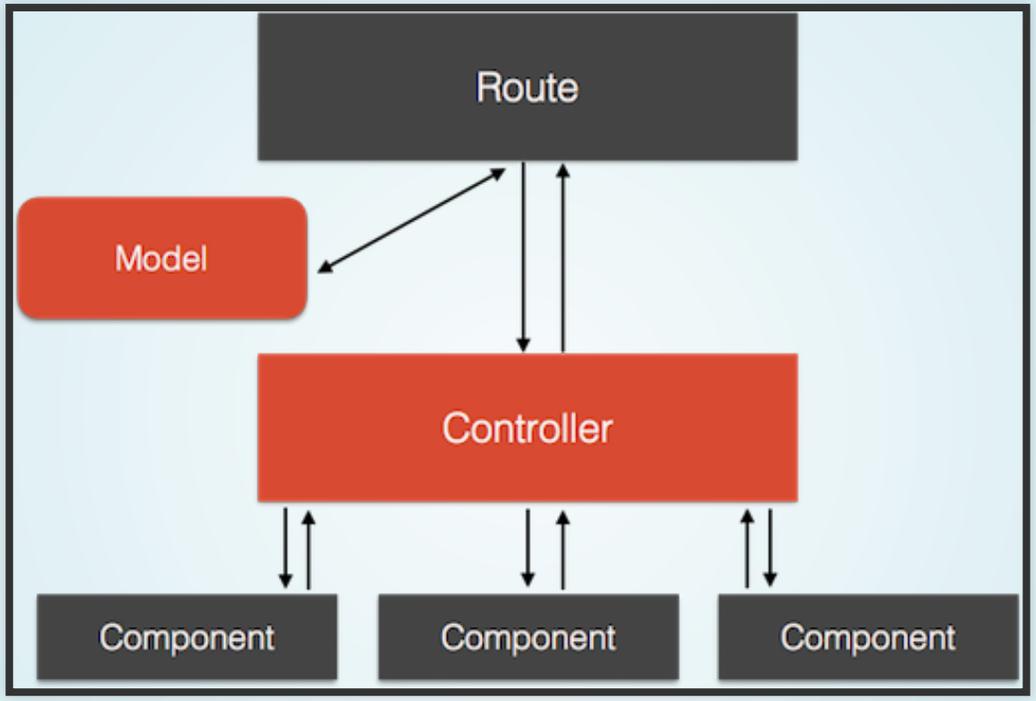
Il fournit quelques petites libraries JavaScript comme **router.js** pour routing, **rsvp.js** pour gérer les promises, **backburner** pour gérér le cycle d’exécution, et **Handlebars** pour le rendu de pages HTML.

Utilisé par beaucoup de sites-web comme Discourse, Groupon, LinkedIn, Vine, Live Nation, Nordstrom, Twitch.tv,

Considéré principalement comme framework Web,
il est aussi possible de créer des applications
Desktop et mobiles.
L'exemple le plus connu d'une app desktop est
Apple Music, une fonctionnalité de **iTunes**.

LES BRIQUES PRINCIPALES DE EMBER JS

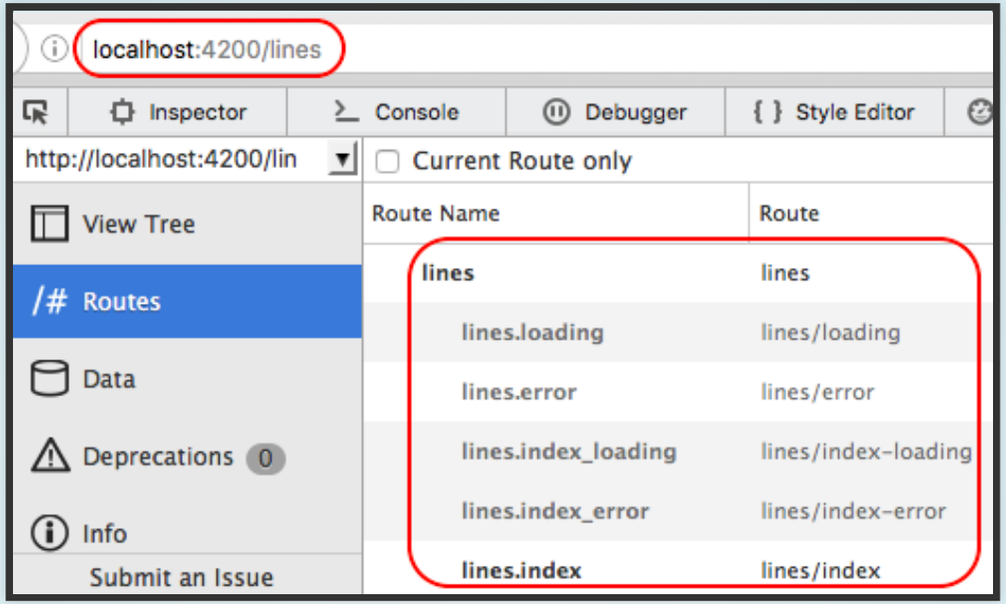
Ember CLI: Un kit d’outils pour créer, développer et compiler applications Ember.



Routing: La partie centrale Ember.

Templating engine - Utilise la syntaxe **Handlebars** pour construire les templates HTML.

Data layer - utilise Ember Data et permet de communiquer avec APIs externes et gérer l'état de l'application.



Ember Inspector - Extension de navigateur pour observer le comportement l'application.

Un gestionnaire de dépendances rapide, fiable et sécurisé

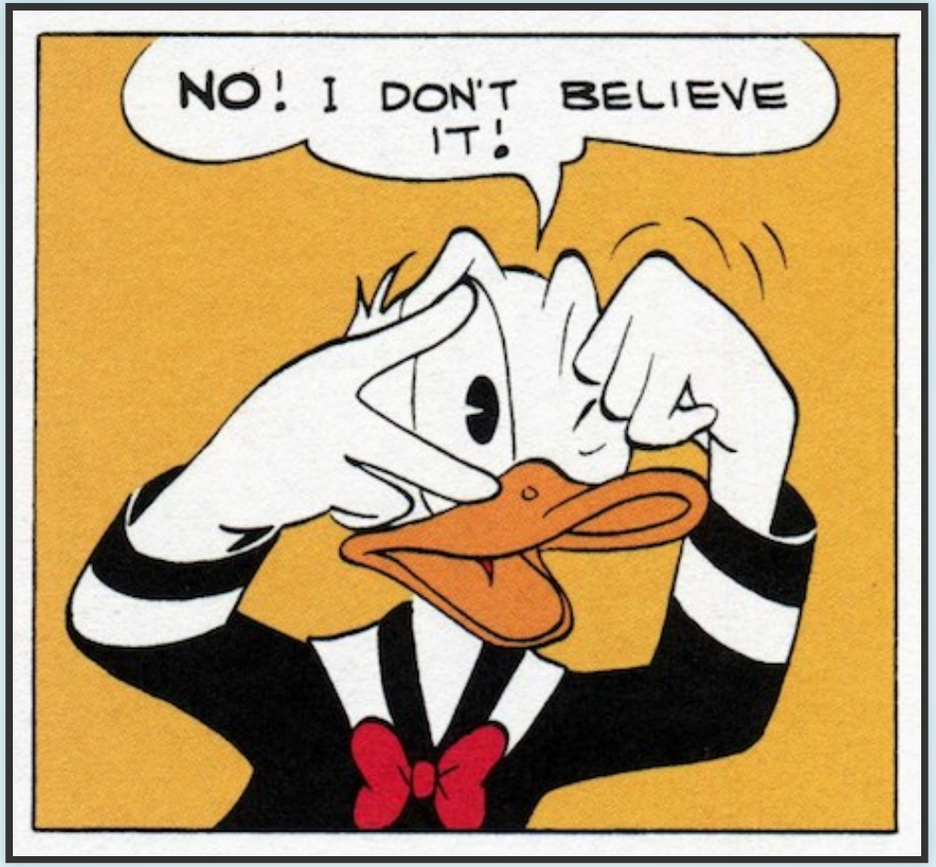
LE CHOIX TECHNIQUE

- **Rails API** comme JSON API back-end
- **JWT** authentication (Implicit grant flow)
- **Ember Simple Auth** add-on (lightweight library for implementing authentication/authorization with Ember apps)
- **Bootstrap 4** (CSS framework)


EMBER
SIMPLE
AUTH

EMBER SIMPLE AUTH:

- maintient la session et synchronise son état à travers de multiples onglets et fenêtres de l'application.
- authentifie la session contre propres serveurs d'applications, mas aussi services externes comme Facebook, Twitter etc.
- authorize les requests vers serveurs backend.
- facile à personnaliser et extensier



On peut développer une app Ember sans avoir besoin de back-end!

Juste utilisez
add-on EMBER-CLI MIRAGE

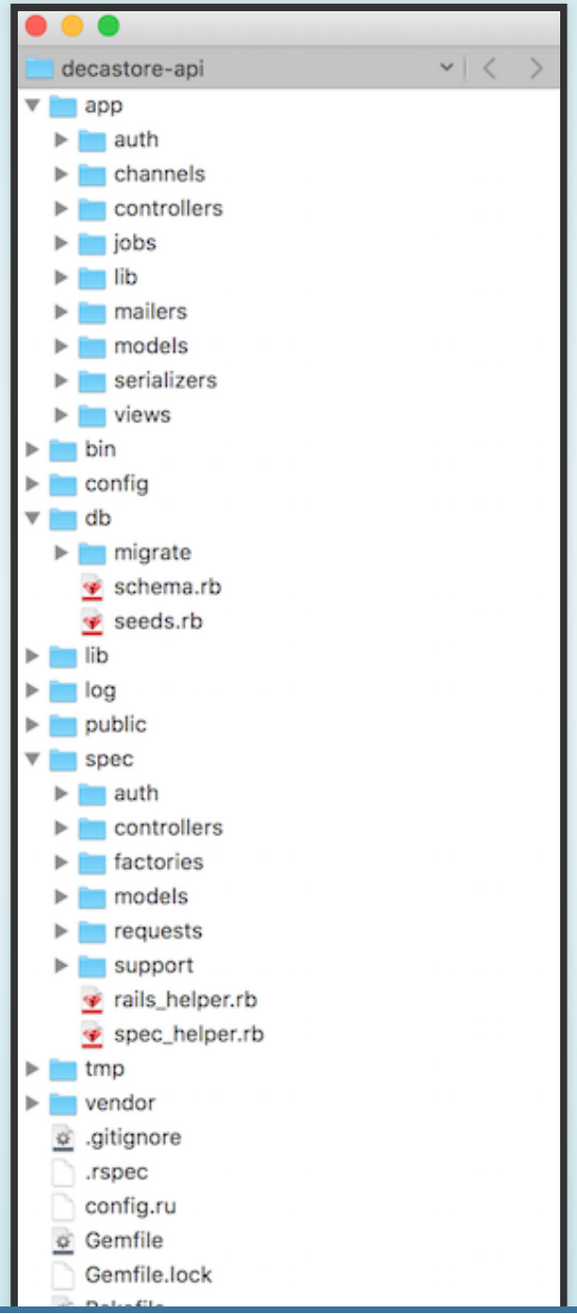
Mirage est un outil pour simuler un serveur API. Vous définissez vos `route handlers` pour répondre aux requests AJAX de votre application.

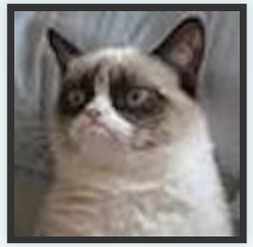
```
// mirage/config.js
export default function() {
  this.namespace = 'api';

  this.get('/authors', () => {
    return {
      authors: [
        {id: 1, name: 'Zelda'},
        {id: 2, name: 'Link'},
        {id: 3, name: 'Epona'},
      ]
    };
  });
}
```

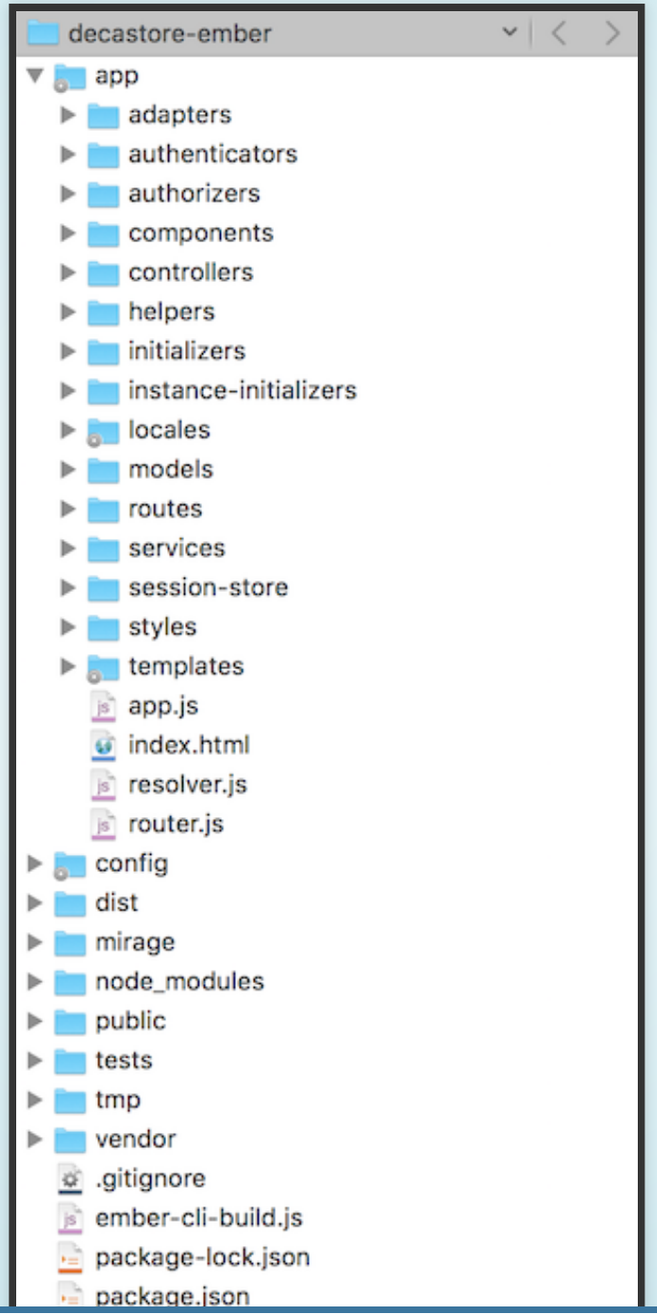
Utiilisez animations et transitions fournies avec Liquid Fire dans votre app Ember JS

A QUOI RESSEMBLE UN PROJET RUBY ON RAILS ?


decastore-api
app
auth
channels
controllers
jobs
lib
mailers
models
serializers
views
bin
config
db
migrate
schema.rb
seeds.rb
lib
log
public
spec
auth
controllers
factories
models
requests
support
rails_helper.rb
spec_helper.rb
tmp
vendor
.gitignore
.rspec
config.ru
Gemfile
Gemfile.lock



ET UNE APLICATION EMBER ?


decastore-ember
app
adapters
authenticators
authorizers
components
controllers
helpers
initializers
instance-initializers
locales
models
routes
services
session-store
styles
templates
app.js
index.html
resolver.js
router.js
config
dist
mirage
node_modules
public
tests
tmp
vendor
.gitignore
ember-cli-build.js
package-lock.json
package.json

Déçu de Ruby on Rails en back-end ?

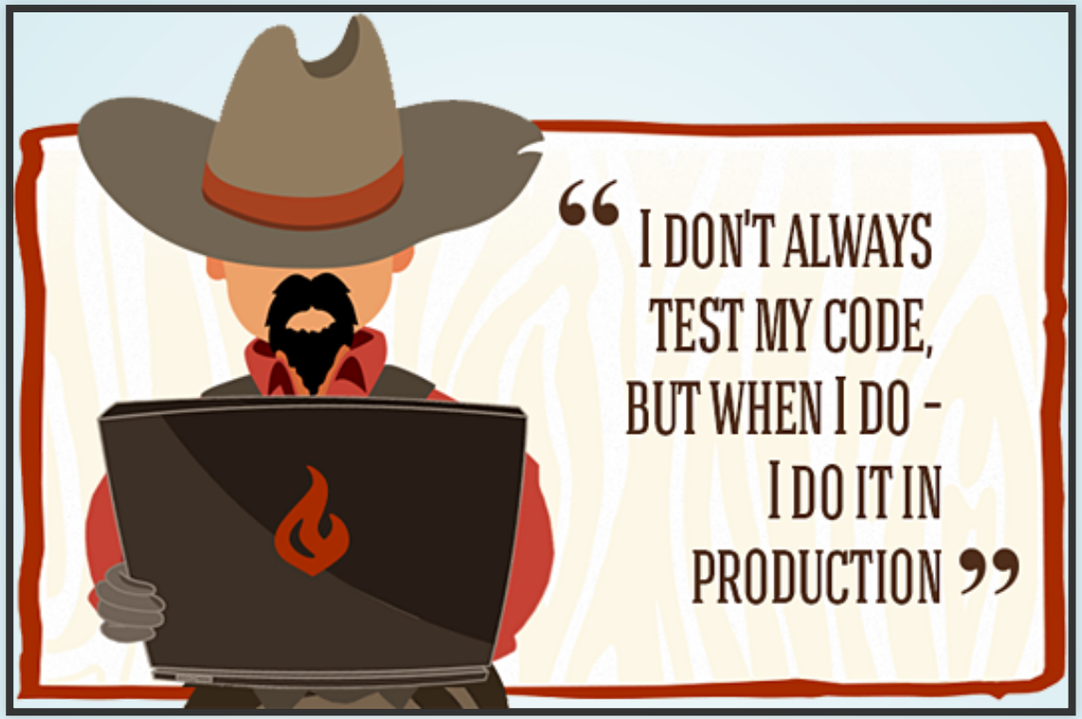
Pas de problèmes, - replacer-le avec ce que vous voulez à condition que votre back-end envoie JSON `application/vnd.api+json`.

et **Ember JS** ferra son job comme avant.

Ember JS ne vous convient pas non plus ?

Pas trop grave non plus, - choisissez ce que vous aimez et branchez-le avec Ruby on Rails en back-end.

COMMENT FAIRE AVEC LES TESTS ?



RSpec est choisi pour tester Ruby code et QUnit est un framework de test fourni par défaut avec Ember

RSpec vous permet de créer:

- Model specs
- Request specs
- Controller specs
- Feature specs
- Mailer specs
- Job specs
- View specs
- Routing specs
- Helpers specs

Un exemple de RSpec request test:

```
RSpec.describe "home page", :type => :request do
  it "displays the user's username after successful login" do
    user = FactoryGirl.create(:user, :username => "jdoe", :passwo
    visit "/login"
    fill_in "Username", :with => "jdoe"
    fill_in "Password", :with => "secret"
    click_button "Log in"

    expect(page).to have_selector(".header .username", :text => "
  end
end
```

Par défault, lorsque vous créez un nouveau component, helper, service, ou un autre module dans une aplication Ember, Ember CLI va automatiquement créer un fichier de test QUnit basé sur le générateur utilisé:

- Components
- Helpers
- Controllers
- Routes
- Models

ET SI JAMAIS VOUS RENCONTREZ UNE SITUATION À COMPORTEMENT BIZARRE DANS VOTRE APPLICATION ?

Pas de soucis !

Coté Ruby vous pouvez utiliser **Byebug** gem (default) ou creuser encore plus avec Pry.

Coté Ember il y a aussi excellent

EMBER INSPECTOR

un add-on du navigateur pour vous aider à debugger applications Ember.
Disponible pour Chrome, Firefox ou d'autres navigateurs via bookmarklet.

APPAREILS MOBILES ?

Si jamaisi vous avez besoin d'utiliser Inspector sur un smartphone, vous pouvez installer un add-on Ember CLI Remote Inspector.

Et comment faire pour le déploiement?

OUTILS DE DEPLOIEMENT POUR RUBY/RAILS

- Capistrano
- Phusion Passenger
- Tools pour EC2, Elastic Beanstalk, EC2 Container Service(Amazon) ou Engne Yard
- Heroku CLI pour Heroku
- En conteneur avec Docker

OUTILS DE DÉPLOIEMENT POUR EMBER JS

- Surge
- Heroku CLI pour Heroku
- ember-cli-deploy, le plus compliqué mais aussi le plus flexible.

Questions ?

thank you!