13.Ruby on Rails

19 décembre 2023

Développement web dlm3

Ruby on Rails

HE-Arc 2016/17 DGR et YBL

Connexion

Nom de domaine et port SSH sur: http://srvz-webapp2.he-arc.ch/.

```
# Exemple

$ ssh -p 2030 yoan@srvz-webapp2.he-arc.ch
yoan@yoan$ more ~/README.md
```

Mise à jour de Rails

```
yoan@yoan$ rails -v
Rails 5.0.0.1
yoan@yoan$ gem update
Updating installed gems
...
yoan@yoan$ rails -v
Rails 5.0.1
```

Une application Ruby

Comme pour Laravel, c'est une bonne pratique d'avoir un répertoire pour le contenu publiable sur Internet.

```
$ cd /var/www/app
$ ls
config.ru
Gemfile
Gemfile.lock
public/nginx-puma.png

$ more config.ru
```

Rack 101

```ruby
run ->(env) do
  [
    200,
    {"Content-Type" => "text/html; charset=utf-8"},
    [
      "<!DOCTYPE html>",
      "<meta charset=utf-8>",
      "<title>Hello!</title>",
      "<h1>Hello</h1>",
      "<p>:-)"
    ]
  ]
end
```

Une fonction, Proc ou lambda qui :

- reçoit un tableau associatif de son environement;
- retourne un triplet de réponse HTTP.

Réponse HTTP:

- le code HTTP;
- un tableau associatif des entêtes HTTP;
- un itérateur sur le corps du document.

Gemfile

Un paquet Ruby se nomme une *gemme*.

```
# Gemfile
source "https://rubygems.org"

gem "puma", "~> 3.6.2"
gem "rack"
```

Comme le composer.json pour PHP.

NGINX

```
root /var/www/app/public;

location / {
    try_files $uri/index.html $uri @rack;
}

location @rack {
    proxy_pass http://puma;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_redirect off;
}

upstream puma {
    server unix:/tmp/puma.sock fail_timeout=0;
}
```

Le serveur HTTP qui sert les fichiers statiques (public) et redirige le reste vers le serveur d'application Ruby (puma).

Puma

Le serveur d'application pour Ruby.

```
#!/usr/bin/env puma

environment "production"

directory "/var/www/app"
bind "unix:///tmp/puma.sock"

# À ajouter.
plugin :tmp_restart
```

En PHP, nous utilisions PHP-FPM. Qu'utilisez-vous avec JEE?

Serveur

```
$ ls /etc/services
cron nginx puma sshd syslog

$ pstree
tini   runsvdir   runsv   cron
                  runsv   nginx   4*[nginx]
                  runsv   syslog-ng
                  runsv   bundle   {reactor.rb:151}
                                   {ruby-timer-thr}
                                   {server.rb:301}
                                   6*[{thread_pool.rb*}]
```

Exercice 1

Modifiez l'environnement *puma* en `development`.

`http://[ PRENOM.NOM | GITHUB ].srvz-webapp2.he-arc.ch/` doit afficher:

```
RACK_ENV
    development
```

Première application

Archivez `app`.

```
$ cd /var/www
$ mv app demoapp
```

Créez une nouvelle application Rails.

```
$ rails new app --database=postgresql
$ cd app
```

Si vous changez le nom, vous devez modifier les configurations des serveurs.

Plein de fichiers

```
app/                # votre code
bin/
config/             # fichiers de config
config.ru           # point d'entrée, « index.php »
db/                 # migrations et seeds
Gemfile             # comme le composer.json
Gemfile.lock
lib/
log/
public/             # fichiers publics
Rakefile
README.md
test/               # tests unitaires, fonctionnels, etc.
tmp/
vendor/
```

Exercice 2

Que peut-on faire à l'aide de la commande `rails`?

Et de la commande `bundle`?

Avant Rails 5, rails et rake avaient des rôles séparés, condensés dans rails.

Premier démarrage

```
$ sudo sv restart puma
```

Kaboom!

Connexion

Utilisez pgAdmin[1] pour vous connecter à votre base de données.

```
$ echo $GROUPNAME $PASSWORD
```

Ou pour les durs à cuire :

```
$ psql -h $POSTGRES_HOST -U $GROUPNAME
> \l
> \dn
> \dt
> \q
```

Configuration

```
default: &default
  adapter: postgresql
  encoding: unicode
  pool: <%= ENV.fetch("RAILS_MAX_THREADS") { 5 } %>
  host: <%= ENV.fetch("POSTGRES_HOST") { "localhost" } %>
  port: <%= ENV.fetch("POSTGRES_PORT") { 5432 } %>
  database: <%= ENV["GROUPNAME"] %>
  username: <%= ENV["GROUPNAME"] %>
  password: <%= ENV["PASSWORD"] %>

development:
  <<: *default

test:
  <<: *default
  schema_search_path: test

production:
  <<: *default
  schema_search_path: production
```

---

[1] https://www.pgadmin.org/

Application de démo

Téléchargez l'application pré-configurée pour vous.

```
$ cd /var/www
$ rm -rf app

$ git clone \
        https://github.com/HE-Arc/rails-intro \
        app

$ cd app
$ bundle install
```

Migration

Installation de la base de données.

```
$ rails db:migrate
```

Que s'est-il passé?

(hint: `git status`)

Exercice 3

Créez un produit possédant un titre, une description et un prix.

Réponse

Nous obtenons une migration, un modèle et un test unitaire.

```
$ rails generate model \
    product \
        title:string \
        description:text \
        price:decimal
```

RAD!

Exercice 4

Corrigez le test qui échoue en corrigeant les *fixtures*.

```
$ rails db:rollback

$ git reset --hard
$ git clean -fd
$ git checkout product
$ rails db:migrate

$ rails test
```

Test unitaire

```
# test/models/product_test.rb

class ProductTest < ActiveSupport::TestCase
  test 'T-shirt has a price' do
    product = Product.find_by(title: 'T-shirt')
    assert 0 < product.price
  end
end
```

Solution

```yaml
# test/fixtures/products.yml

tshirt:
  title: T-shirt
  description: Superbe maillot de corps
  price: 9.99
```

Validation

Selon Ruby on Rails, la logique métier ne doit pas se trouver dans la base de données.

```ruby
# app/models/product.rb

class Product < ActiveRecord::Base
  validates :title, presence: true
  validates :price, numericality: { greater_than: 0 }
end
```

Exercice 5

Testez les règles de validations ci-dessus en ajoutant des tests.

```
$ git reset --hard
$ git checkout validation
$ rails test
```

Solution

```
# test/models/product_test.rb

test 'must have a title' do
  assert_not Product.create(price: 10).valid?
end

test 'must have a price greater than zero' do
  assert_raise do
    Product.create!(title: 'Untitled', price: 0)
  end
end
```

Contrôleur

```
$ rails g controller products index

app/assets/javascripts/products.coffee
          /stylesheets/products.scss
   /controllers/products_controller.rb       # def index; end
   /helpers/products_helper.rb
   /views/products/index.html.erb            # index.html.erb

config/routes.rb                             # get 'products/index'

test/controllers/products_controller_test.rb # should get index
```

Par convention, un modèle est au singulier et un contrôleur au pluriel.

Test unitaire

```
# test/controllers/products_controller_test.rb

test 'should get products on /' do
  get '/'

  assert_response :success
```

```
  assert_not_nil assigns(:products)
end
```

Exercice 6

Corrigez le test du contrôleur.

```
$ git reset --hard
$ git clean -fd
$ git checkout controller

$ rails test
```

Solution

```
# config/routes.rb
root 'products#index'

# app/controllers/products_controller
def index
  @products = Product.all
end

# app/views/products/index.html.erb
<% @products.each do |product| %>
  <h2><%= product.title %></h2>
<% end %>
```

Taille

Création d'un modèle pour les tailles de nos t-shirts.

```
$ rails generate model size name:string
```

Exercice 7

Créez un seeder pour les tailles allant de `XS` à `XXL`.

```
$ git reset --hard
$ git clean -fd
$ git checkout sizes
$ rails db:migrate

$ rails db:seed

# Test
$ rails console
> pp Size.all
```

Solution

```
# db/seeds.rb

Size.create([
  {name: 'XS'},
  {name: 'S'},
  {name: 'M'},
  {name: 'L'},
  {name: 'XL'},
  {name: 'XXL'}
])
```

Relation Produits - Tailles

```
$ rails g migration associate_products_and_sizes

# db/migrate/..._associate_products_and_sizes.rb

create_table :products_sizes do |t|
  t.references :product, :index => true
  t.references :size, :index => true
end
```

Many-to-many

Dans chaque modèle.

```
# app/models/product.rb
has_and_belongs_to_many :sizes, uniq: true

# app/models/size.rb
has_and_belongs_to_many :products, uniq: true
```

Test

```
$ git reset --hard
$ git clean -fd
$ git checkout habtm
```

Tests depuis la console.

```
$ rails console
> xxl = Size.find_by(name: 'XXL')
> xxl.products.size
=> 0
```

xxl.products.create(title: ‘A’, description: ‘B’, price: 10)

Administration

```
$ more Gemfile

# Automagic admin interface.
gem 'rails_admin', '~> 1.1'

$ bundle install
$ rails g rails_admin:install
$ touch tmp/restart.txt
```

```
$ git reset --hard
$ git checkout admin
$ bundle install
```

Image

Ajoutez une image à vos produits

```
$ more Gemfile

# Toughtbot's paperclip to upload files
gem 'paperclip', '~> 5.1'

$ bundle install
```

Migration

```
$ rails g migration add_image_to_product

def change
  change_table :products do |t|
    t.attachment :image
  end
end
```

Exercice 8

Faites qu'on puisse attacher une image depuis l'interface d'administration.

```
$ git reset --hard
$ git clean -fd
$ git checkout images
$ rails db:migrate
```

Indice: lire la documentation de `paperclip`.

Solution

```
# app/models/product.rb

has_attached_file: image
validates_attachment_content_type :image, \
    content_type: /\Aimage/
validates_attachment_file_name :image, \
    matches: [/png\z/, /jpe?g\z/]
```

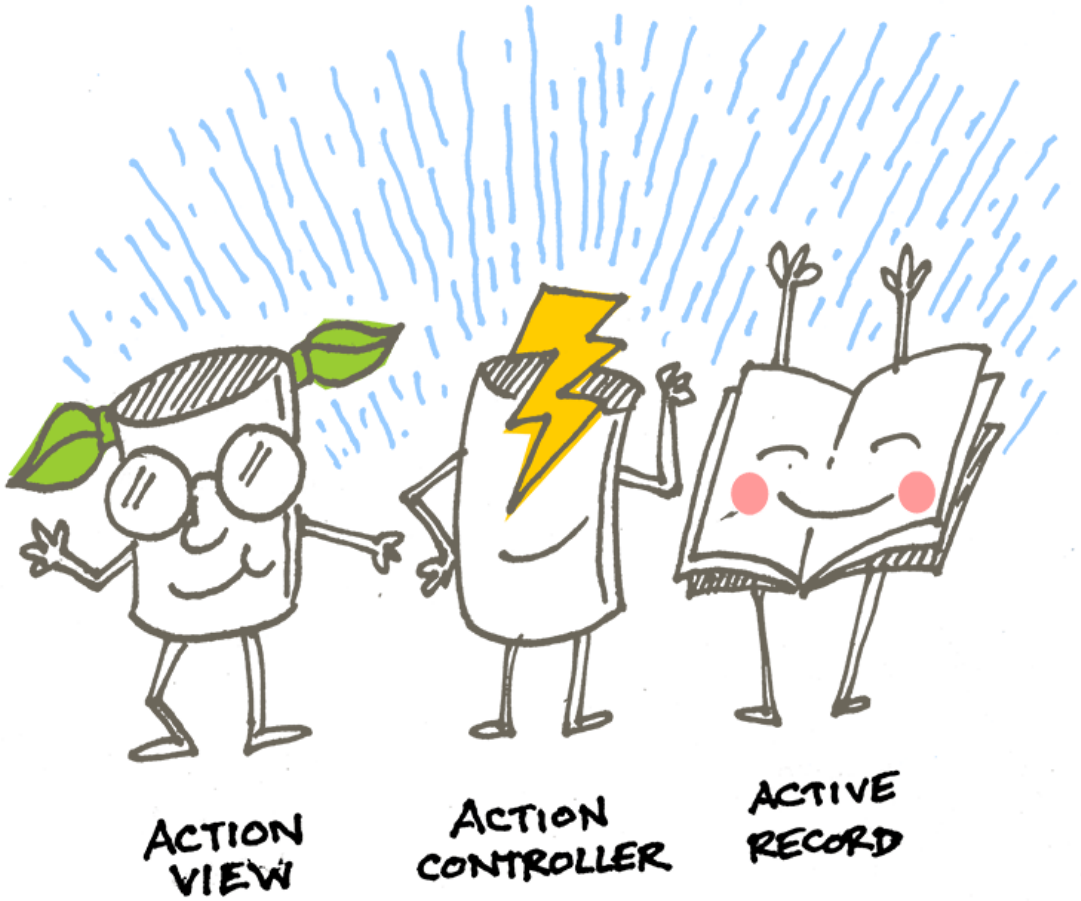
Ressource

Il aurait été possible de créer modèle, contrôleur et routes de type REST.

```
$ rails generate resource person
$ rails routes
...
```

Testez!

Détails intéressants de Rails



CSS et JavaScript

- `foundation-rails`
- `bootstrap-sass, ~> 3.3.7`
- `bootstrap, ~> 4.0.0.alpha6`
- `basscss-rails`

- `bulma-rails`
- `mui-sass`
- etc.

Voir Asset Pipeline[2]

Rails 5.1 proposera de gérer ces éléments-là via webpack ou yarn. D'ici là, il nous faut passer par les gems associées.

ActionCable

La nouveauté de Rails 5.0.

Gestion simplifiée des `WebSocket` permettant d'incorporer des fonctionnalités « temps-réel ».

Voir Action Cable Overview[3]

ActiveJob

Gestion des tâches de fond, comme envoyer des e-mails, redimensionner des images, …

Voir Active Jobs Basics[4]

ActionView

La bonne méthode pour créer des formulaires et les lier à des données.

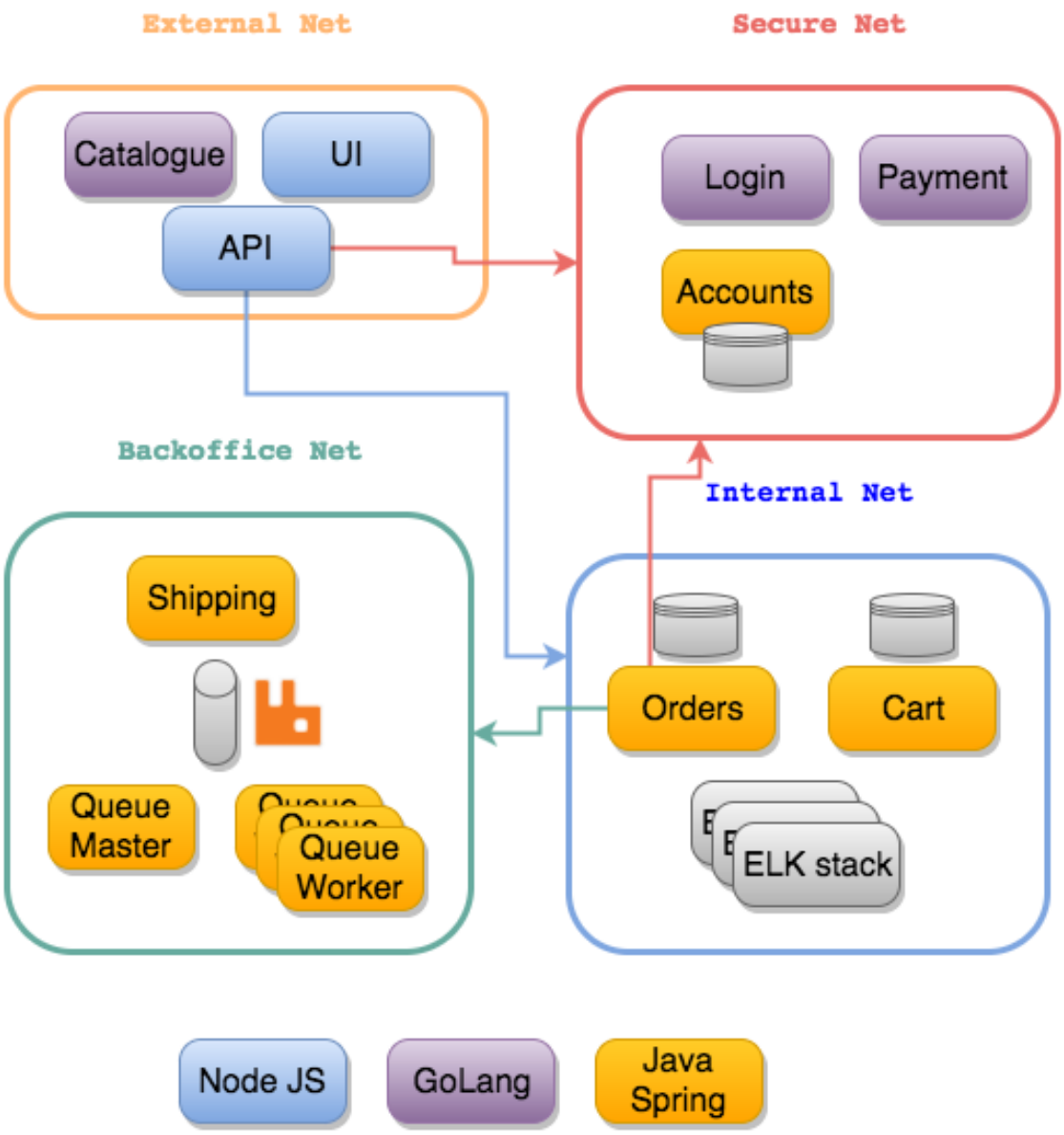
```
<%= form_for @article, url: {action: 'create'} do |f| %>
  <%= f.text_field :title %>
  <%= f.submit 'Create' %>
<% end %>
```

Voir Form Helpers[5]

---

[2] http://guides.rubyonrails.org/asset_pipeline.html
[3] http://guides.rubyonrails.org/action_cable_overview.html
[4] http://guides.rubyonrails.org/active_job_basics.html
[5] http://guides.rubyonrails.org/form_helpers.html

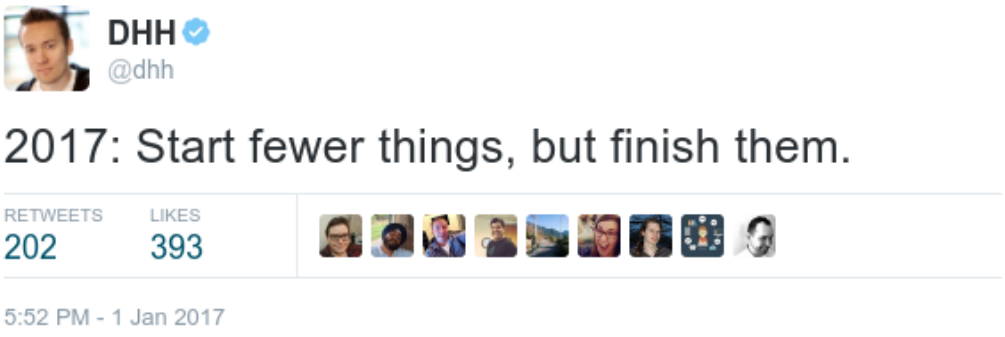
Problème avec Ruby on Rails



Conclusion

- Laravel tire son inspiration première de Ruby on Rails.

- Rails est plus cohérent dans son ensemble tirant partie des fonctionnalités de Ruby.



(1)

Difficultés pour vous

- Construisez un produit au fur et à mesure
- Déployez souvent
- Essayer des bibliothèques
- Et ayez un plan!

Sources

1. HEINEMEIER HANSSON, David. 2017: Start fewer things, but finish them. [en ligne]. 2017. [Consulté le 7 février 2017]. Disponible à l'adresse : https://twitter.com/dhh/status/815601578329575424