

TP – Ruby on Rails

FERTILLE Eliott

31/12/2023

—

**Développer des composants
d'accès aux données**

—

Mr. BENTIFRAOUINE

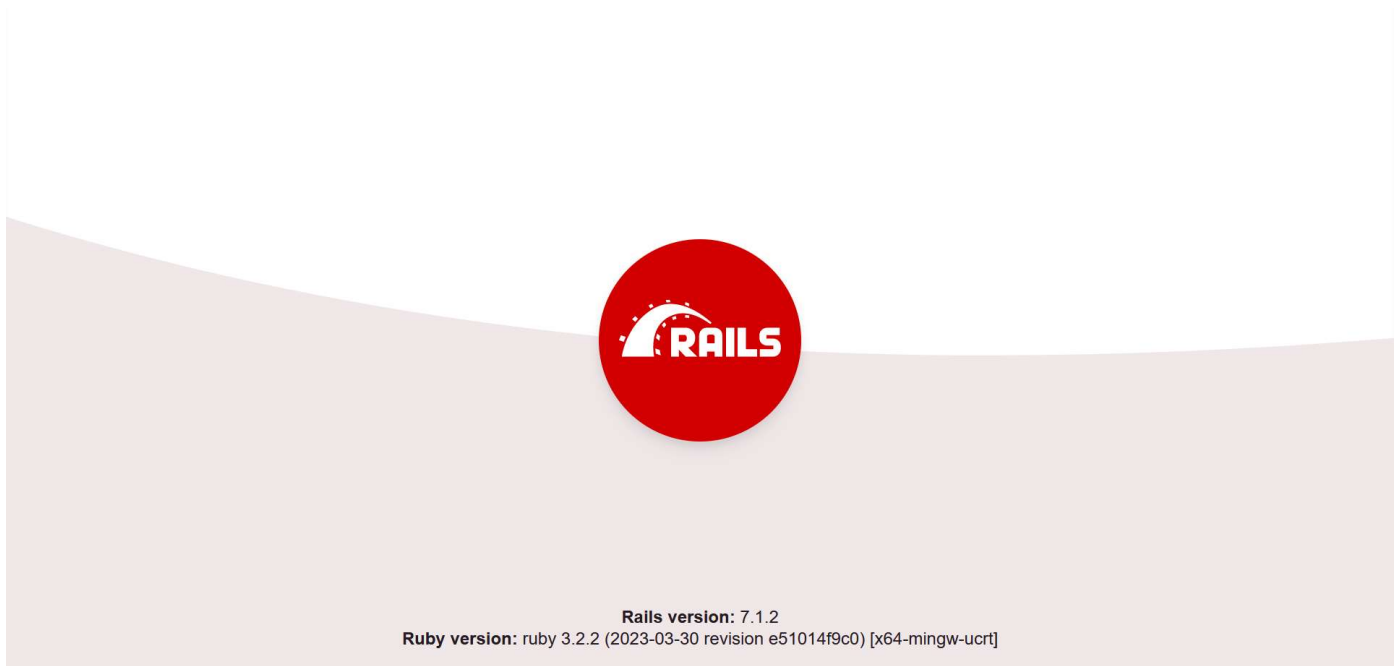
Création de l'application Ruby

J'ai créé mon app Ruby en lançant les commandes suivantes :

- rails new LeCarnetDeVoyage
- cd LeCarnetDeVoyage
- bundle install
- gem update
- bundle
- rails server

De ce fait, mon projet se nomme 'LeCarnetDeVoyage' et toutes les dépendances liées au Framework Ruby sont bien installées et à jour.

La commande 'rails server' me permet de lancer mon projet une première fois pour voir si tout va bien :



Configuration de la base

Maintenant pour créer mes premiers models User et Trip, j'utilise un outil de Ruby : Scaffold

Grâce à ça, je peux créer mes models simplement avec les commandes suivantes :

- rails g scaffold User nom:string email:string
- rails g scaffold Trip destination:string description:text dateDebut:date dateFin:date

Après avoir créé mes models, je migre ma base avec la commande 'rails db:migrate' pour mettre à jour le schéma de la base.

Sans Scaffold, j'aurais tout simplement créé mon model User et Trip comme suit :

- rails g model User nom:string email:string password:string
- rails g model Trip destination:string description:text dateDebut:date dateFin:date

Puis ensuite j'aurais ajouté toutes les opérations CRUD dans les controller créé au préalable avec les commandes :

- rails g controller User
- rails g controller Trip

Pour établir la relation one-to-many, j'ai tout simplement ajouté les relations dans les models User et Trip.

User :

```
user.rb U X trip.rb U
LeCarnetDeVoyage > app > models > user.rb
1 class User < ApplicationRecord
2   has_many :trips
3 end
4
```

Trip :

```
user.rb U trip.rb U X
LeCarnetDeVoyage > app > models > trip.rb
1 class Trip < ApplicationRecord
2   belongs_to :user
3 end
4
```

Manipulation des données

Pour ce qui est des routes, j'ai décidé de mettre la page index des trips en route par défaut pour que le projet se lance directement sur les voyages que comporte la base et donner la possibilité d'en ajouter en se connectant.

```
routes.rb U X users_controller.rb U trips_controller.rb U sessions_controller.rb U application.css U
LeCarnetDeVoyage > config > routes.rb
1 Rails.application.routes.draw do
2   root 'trips#index'
3
4   resources :users
5   resources :trips
6
7   get 'login', to: 'sessions#new'
8   post 'login', to: 'sessions#create'
9   delete 'logout', to: 'sessions#destroy'
10
11
12
13 # Define your application routes per the DSL in https://guides.rubyonrails.org/routing.html
14
15 # Reveal health status on /up that returns 200 if the app boots with no exceptions, otherwise 500.
16 # Can be used by load balancers and uptime monitors to verify that the app is live.
17 get "up" => "rails/health#show", as: :rails_health_check
18
19 # Defines the root path route ("/")
20 # root "posts#index"
21 end
```

Les deux models User et Trip sont bien gérés par des routes REST ce qui facilite les opérations CRUD.

```

LeCarnetDeVoyage > app > controllers > trips_controller.rb
1 class TripsController < ApplicationController
2   # Pour effectuer toutes actions CRUD sur un Trip il faut être connecté
3   before_action :logged_in_user, only: [:new, :create, :edit, :update]
4
5   # GET /trips or /trips.json
6   # Affiche tous les voyages
7   def index
8     @trips = Trip.all
9   end
10
11  # GET /trips/1 or /trips/1.json
12  # Affiche un voyage en fonction de son id
13  def show
14    @trip = Trip.find(params[:id])
15  end
16
17  # GET /trips/new
18  # Gestion des nouveaux voyages
19  def new
20    @trip = Trip.new
21  end
22
23  # GET /trips/1/edit
24  def edit
25  end
26
27  # POST /trips or /trips.json
28  # Création d'un voyage
29  def create
30    @trip = current_user.trips.build(trip_params)
31
32    respond_to do |format|
33      if @trip.save
34        format.html { redirect_to trip_url(@trip), notice: "Trip was successfully created." }
35        format.json { render :show, status: :created, location: @trip }
36      else
37        format.html { render :new, status: :unprocessable_entity }
38        format.json { render json: @trip.errors, status: :unprocessable_entity }
39      end
40    end
41  end
42
43  # PATCH/PUT /trips/1 or /trips/1.json
44  # Modification d'un voyage
45  def update
46    respond_to do |format|
47      if @trip.update(trip_params)
48        format.html { redirect_to trip_url(@trip), notice: "Trip was successfully updated." }
49        format.json { render :show, status: :ok, location: @trip }
50      else
51        format.html { render :edit, status: :unprocessable_entity }
52        format.json { render json: @trip.errors, status: :unprocessable_entity }
53      end
54    end
55  end
56
57  # DELETE /trips/1 or /trips/1.json
58  # Suppression d'un voyage
59  def destroy
60    @trip = Trip.find_by(id: params[:id])
61    if @trip
62      @trip.destroy
63      redirect_to trips_path, notice: 'Voyage supprimé avec succès.'
64    else
65      redirect_to trips_path, alert: 'Voyage introuvable.'
66    end
67  end
68
69  private
70  # Use callbacks to share common setup or constraints between actions.
71  def set_trip
72    @trip = Trip.find(params[:id])
73  end
74
75  # Only allow a list of trusted parameters through.
76  # Paramètres de base d'un voyage
77  def trip_params
78    params.require(:trip).permit(:destination, :description, :dateDebut, :dateFin)
79  end
80 end
81

```

Toutes les actions CRUD ont été créées automatiquement par Scaffold mais j'ai codé moi-même la logique de chaque fonction.

J'ai créé des views assez simple juste pour faire plus propre dans l'affichage, mais toute la logique métier fonctionne très bien. J'ai notamment utilisé l'outil Bootstrap pour construire les views.

Vue globale :

Le carnet de voyages

Accueil

Créer un compte

Bonjour, Eliott

Déconnexion

Ajouter un voyage

Destination: Bruxelles

Description: Visite famille

Du : 04/02/2023

Au : 10/02/2023

Voir les détails

Destination: Montreal

Description: Voyage d'étude

Du : 29/04/2024

Au : 22/06/2024

Voir les détails

Vue précise sur un voyage :

Le carnet de voyages

Accueil

Destination: Bruxelles

Description: Visite famille

Du : 04/02/2023

Au : 11/02/2023

Modifier

Supprimer

Création d'un voyage (formulaire avec vérification de cohérence des dates) :

Le carnet de voyages

Accueil

Créer un compte

Bonjour, Eliott

Déconnexion

New trip

Destination

Description

Datedebut

jj / mm / aaaa

Datefin

jj / mm / aaaa

Create Trip

[Back to trips](#)

Modification d'un voyage :

Editing trip

Destination	<input type="text" value="Bruxelles"/>
Description	<input type="text" value="Visite famille"/>
Datedebut	<input type="text" value="04 / 02 / 2023"/>
Datefin	<input type="text" value="11 / 02 / 2023"/>
<input type="button" value="Update Trip"/>	

À noter que j'ai ajouté la possibilité de s'inscrire en tant que nouvel utilisateur pour poster soi-même un voyage dans le carnet.

J'ai également ajouté des fonctionnalités :

- Un utilisateur ne peut supprimer ou modifier un voyage uniquement si c'est le sien ou qu'il est administrateur
- Seul un administrateur peut donner les droits d'admin à un autre et il peut lui retirer
- Un utilisateur peut modifier les données liées à son compte, son mot de passe lui sera alors redemandé