Anatomie d’une application Rails

Ce document se veut une référence détaillant la structure générale d’une application Rails, et est donc une source d’information utile pour la maintenance des diverses applications de visualisation et de manipulation de données développées, notamment OVAP et le visualisateur de réseau.

La plateforme de développement Rails répond au concept de MVC, soit Model-View-  
Controller, où

* le modèle représente des classes d’objet et dans certains cas des tables de données associées pour le stockage de l’information,
* le view est le code de l’interface lui-même basé sur un principe de template (une page est rendue par le serveur et peut donc inclure des variables ruby)
* le controller reçoit des requêtes et exécute les commandes en fonctions de celles-ci

Le controller jouant le rôle d’intermédiaire, la majorité de la complexité du code est conservé dans les fichiers de view ainsi que dans le modèle sous un principe orienté-objet, c’est-à-dire que le code est contenu dans des méthodes spécifique à chacune classes.

# Installation

Toute application Rails repose sur le langage Ruby. Il est donc essentiel de procéder à l’installation du langage sur la machine qui hébergera la ou les applications. Sur Linux ou Mac, il suffit d’installer rvm (www.rvm.io), tandis que sur Windows, l’installation repose sur le sous-système msys, qui fournit entre autre un environnement UNIX de base ainsi qu’un compiler, qui est essentiel pour obtenir certaines extensions ruby (appelés gem). MSYS ainsi que l’installeur ruby pour windows peuvent être obtenus à l’adresse suivante : <https://rubyinstaller.org> . Les applications développées reposent aussi sur la base de données PostgreSQL/PostGIS, pour lesquels un installeur est aussi disponible pour Windows.

Une fois ces outils installés, il est possible de commencer à interagir avec l’application dans une console. Le terminal de Windows étant plutôt limité (impossible de sélectionner/copier-coller), une implémentation alternative peut être intéressante dans le but de simplifier les opérations. À cet effet, ConEmu peut combler plusieurs de ces lacunes (<https://conemu.github.io/>).

Après s’être déplacé dans le dossier de l’application, il est important de procéder à l’installation des gems nécessaires. La commande « bundle install » permet d’accomplir ce travail (lors de la première installation, il est possible qu’il soit nécessaire d’installer bundler, avec la commande « gem install bundler »). Une fois ces étapes complétées, il ne reste qu’à configurer l’application avant de pouvoir commencer à en faire usage. Cette configuration se limite à toute fin pratique à l’édition du fichier database.yml, qui fournit les informations de connexion à la base de données (utilisateur/mot de passe). Il ne reste alors qu’à créer la base de données et y mettre en place le schéma avec les commande « rake db:create » et « rake db:migrate ».

## migrations

La structure de la base de données d’une application rails repose sur un système de migration. Celles-ci sont des fichiers contenant un ensemble de commande permettant de générer une structure pour la base de données et d’établir les liens entre celles-ci. Cette façon de faire permet d’éviter la création de longues requêtes SQL pour la création ou la modification d’une base de données, en plus d’assurer que les modifications à la structure seront portables à long terme : si les modifications sont faites dans le cadre de nouvelles migration, l’exécution de l’ensemble des migrations ramènera toujours l’application à l’état final désiré.

## rake

Rake est un exécuteur de tâches prédéfinies. Il permet de lancer rapidement un ensemble d’opérations sans avoir besoin d’entrer dans la ligne de commande (rails c). Toutes les applications développées reposent sur rake afin de lancer diverses tâches d’importation, d’exportation ou de maintenance de l’application. Les tâches définies par l’utilisateur sont contenues dans le dossier lib/tasks.

# Structure de l’application

Tel que mentionné précédemment, les applications reposent sur trois grands ensembles, soit les controllers, les models et les views. Ceux-ci sont respectivement situés dans les dossiers app/controllers, app/models et app/views.

## controllers

Les controllers ont pour principale mission de recevoir des commandes de la part du navigateur. Lorsqu’une commande est appelée par un navigateur, l’application tente d’abord de déterminer à quel controller la requête doit être envoyée. La définition de ce chemin est faite dans le fichier config/routes.rb. Une ligne du fichier routes prend la forme suivante :

get ‘nom\_de\_la\_page’, to: ‘controller#methode’

Ce qui signifie que si on appelle l’application à l’adresse nom\_de\_la\_page, la requête sera envoyée à la méthode « methode » du controller « controller ». Celui-ci exécutera alors le code à l’intérieur de la méthode et renverra de l’information au navigateur, soit sous la forme d’une page générer à partir du fichier view, soit en répondant par un fichier défini à même le controller (par exemple, un fichier json qui sera interprété par le navigateur).

## models

Les fichiers models contiennent la grande majorité du code ruby de l’application. Leur rôle est de fournir des méthodes aux différentes classes créées. Puisque ceux-ci sont connectés directement aux tables de la base de données, ils permettent d’exécuter des requêtes directement sur la structure sans nécessairement devoir écrire du code SQL.

## views

Les fichiers views sont la porte d’entrée visuelle des applications. Elles contiennent le code HTML à générer lors d’une requête. Ce code est défini sous la forme de « template », et des commandes ruby peuvent y être incluses pour insérer des variables ou d’autres éléments. Le code HTML est aussi complété par du javascript, permettant de rendre l’interface dynamioque et d’interagir avec le serveur via des requêtes AJAX. Notamment, les applications développées utilisent des requêtes de type « post » permettant d’obtenir de l’information de certains controller, pour ensuite interpréter les données reçues et les afficher aux endroits appropriés (carte, graphique, tableau, etc.).

Le fichier layout.html.erb contient quant à lui les en-têtes de toutes les vues qui seront générées, et contient par conséquent les appels à divers librairies javascript ainsi qu’à des fichiers CSS permettant d’appliquer des styles à l’application. Les librairies MapboxGL et jQuery sont utilisés dans toutes les applications, la première permettant de générer des cartes, la seconde offrant de nombreux utilitaires permettant de simplifier de nombreuses commandes, notamment les requêtes AJAX. Des fichiers de javascript et de style locaux sont aussi stockés dans le répertoire app/assets.

# Utilitaires

Si la plupart des commandes de maintenance sont utilisables sous la forme de tâches rake, il demeure possible d’interagir autrement avec l’application. Rails fourni notamment une console interactive qui peut être atteinte en exécutant la commande « rails c » dans la ligne de commande.

Une fois l’application dans l’état désiré, il est aussi nécessaire de lancer le serveur, avec la commande « rails s ». Il est aussi important de s’assurer d’ajouter des paramètres au serveur de développement pour que l’application soit accessible. La forme suivante permettra d’atteindre les objectifs pour l’application soit disponible au bon port, et qu’elle soit accessible sur le réseau local :

rails s –p 3000 –b 0.0.0.0

où 3000 représente le port désiré.