# TP 2 : Construire un blog avec Ruby On Rails

## *Objectif*

Réaliser un blog en utilisant Ruby On Rails, PostgreSQL via ActiveRecord, HAML et Twitter Bootstrap.

## *Étapes*

### *Création du projet*

1. Créer un répertoire qui contiendra le nouveau projet Ruby On Rails
2. Se placer dans ce répertoire
3. Utiliser la version **`2.3.0`** de Ruby pour ce projet :
   **`rbenv local 2.3.0`**
4. Vérifier la version de Ruby avec **`ruby –v`**
5. Installer la gem **`rails`** :
   **`gem install rails`**
   **`rbenv rehash`**
6. Initialiser dans ce répertoire un nouveau projet Rails qui utilisera PostgreSQL :
   **`rails new . –d postgresql`**
7. Lancer le serveur :
   **`bundle exec rails server`**
8. Vérifier l'installation en accédant depuis votre navigateur à la page : http://localhost:3000

### *Configuration et connexion à la base de données*

1. Démarrer PostgresSQL si nécessaire
2. Créer un utilisateur PostgreSQL non superuser autorisé à créer des bases de données :
   **`createuser ––interactive –P`**
3. Ouvrir le fichier **`config/database.yml`** et éditer les configurations de **`development`** et **`test`**.
   Par exemple pour **`development`** :

```
development:
     adapter: postgresql
     encoding: unicode
     database: tpblograils_development
     pool: 5
     username: pguser
     password: 123soleil
```

4. Vérifier que la config est bonne en créant la base de données :
   **`bundle exec rake db:create`**

### *Création du modèle de données*

1. Créer le modèle **`Article`** dont les instances auront un titre (**`title`**) et un corps de texte (**`body`**) pour attributs :

```
bundle exec rails generate model article title:string
body:string
```

2. Lancer la migration :
```
bundle exec rake db:migrate
```
3. Un article ne doit pas être enregistré en base s'il ne contient pas de titre et/ou de corps de texte. Ajouter des validateurs au modèle Article de sorte que le titre et le corps du texte soient des champs requis :
```
validates :title, presence: true
validates :body, presence: true
```
4. Vérifier dans la console qu'un article sans titre ne peut pas être enregistré. Idem pour un article sans corps de texte
```
bundle exec rails console (pour lancer une console)
```
5. Générer 10 articles fictifs en les définissant dans **db/seeds.rb**
6. Charger le seed :
```
bundle exec db:seed
```
7. Comme vous pouvez chaîner l'appel des tâches rake, il est aisé de reconstruire une base de données :
```
bundle exec rake db:drop db:create db:migrate db:seed
```

### *Liste et consultation des articles*

1. Ajouter la gem **haml-rails** au fichier **Gemfile**
2. Installer la nouvelle gem et relancer le serveur :
```
bundle install
bundle exec rails server
```
3. Créer un contrôleur pour afficher la liste des articles et consulter un article spécifique :
```
bundle rails generate controller articles index show
```
4. Charger la liste des articles dans l'action **index** du contrôleur et afficher les extraits des articles dans la vue correspondante.
5. Pour chaque article de la page **articles#index**, ajouter un lien vers la page de consultation de l'article complet
6. Charger un article spécifique en fonction du paramètre **id** dans l'action **show** du contrôleur et l'afficher dans la vue correspondante
7. Définir la racine du site dans les routes de sorte qu'elle pointe vers la liste des articles

### *Styles et templates*

1. Convertir le layout **app/views/layouts/application.html.erb** en **app/views/layouts/application.html.haml**
2. Inclure Twitter Bootstrap au layout :
```
= stylesheet_link_tag
"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.
css"
= javascript_include_tag
"//https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min
.js"
```

3. Ajouter une en-tête de page et un pied de page stylisés avec Bootsrap (voir le fichier app/views/layout/application.html.haml)
4. Ajouter du style à la page **`index`** et la page **`show`**

### *Pagination*

1. Ajouter la gem **`will_paginate`** (https://github.com/mislav/will_paginate) au fichier **`Gemfile`**, installer la nouvelle gem et relancer le serveur
2. Limiter le nombre d'articles par page à 3
3. Ajouter les liens de pagination en utilisant le helper adéquat

### *Création d'un article*

1. Ajouter l'action **`articles#new`** et créer la vue correspondante
2. Ajouter un formulaire de création d'article dans cette vue (http://guides.rubyonrails.org/form_helpers.html)
3. Ajouter l'action **`articles#create`** qui redirigera vers la liste des articles si la création s'est déroulée correctement, et affichera à nouveau la page de création de l'article sinon.
4. Ajouter un lien depuis la page d'index des articles vers la page contenant le formulaire