Projet "Grassouillet"

Langages de scripts et développement Web

Jason JAMET
11 février 2015

Professeur : Gilles HUNAULT
Master informatique

Table des matières

1 Introduction **2**
1.1 Cahier des charges 2
1.2 Outils utilisés 2
1.3 Fonctionnalités implémentés 2

2 Choix d'implémentation **3**
2.1 Création 3
2.2 Gems 3
2.3 Tests 4

3 Difficultés rencontrées **4**

4 Conclusion **4**

5 Annexes **5**
5.1 Commandes 5

6 Bibliographie **5**

1 Introduction

1.1 Cahier des charges

Pour le contrôle continu, vous devez réaliser un site Web fonctionnel sur votre portable sous Linux, opérationnel sur un autre ordinateur en utilisant Ruby on Rails 3 ou 4. Pour cela, vous lirez le tutoriel Apprendre Rails par l'exemple de M. Hartl, chapitres 1 à 6 et vous réaliserez tout ce qu'il propose au moins jusqu'au chapitre 6, à savoir un site d'utilisateurs avec nom et e mail. Si cela peut vous gagner du temps, vous pourrez ignorer tout ce qui heroku (mais pas ce qui concerne git).

Vous viendrez ensuite compléter le site avec la saisie d'une date de naissance et l'affichage de l'age, la saisie d'un poids actuel et d'un poids idéal (qui doit être inférieur au poids actuel) et l'affichage de l'IMC, l'indication je ne fais pas de régime et j'aimerais faire un régime tous deux en mode oui/non et le dépôt d'un CV au format PDF avec la consultation possible par les autres utilisateurs. Au passage, vous en profiterez pour traduire tous les affichages en français, histoire de prouver que vous savez d'où vient chaque page.

En option : vous pourrez aussi implémenter le service des pages en XHTML strict plutôt qu'en HTML 5 (une validation sans erreur ni avertissement avec HTML Validator est conseillée), avec l'Icône de validation associée, la liste des gens non en cours de régime qui décident d'en faire un, le masquage et l'affichage en Javascript de j'aimerais faire un régime pour ceux qui en font déjà un, une liste bien présentée disponible en PDF à la volée de tous les utilisateurs, un graphique SVG qui fournit un box-plot des poids réels et actuels, un graphique PNG de l'histogramme de la distribution des poids réels...

1.2 Outils utilisés

— Ruby on Rails 4.2.0
— Ruby 1.9.3
— Diverses gem
— SQLite 3
— XHTML 1.0 strict
— CSS 3
— Atom 0.175.0
— Debian 7
— Git (dépôt Redmine)

1.3 Fonctionnalités implémentés

"Grassouillet" est un site web proposant l'inscription, le listing, la modification et la suppression d'utilisateurs. Des statistiques concernant le poids (et Indice de masse corporelle) des personnes sont proposées sous forme de graphes. Il est également possible d'afficher une liste des utilisateurs, soit dans sa forme complète (disponible en export pdf), soit dans une forme restreinte suivant le critère "Utilisateur ne faisant pas un régime, mais souhaitant en faire un". Lors du remplissage du formulaire d'inscription, il est possible d'uploader un CV, qui sera ensuite consultable dans la section "détails" de l'utilisateur.

2 Choix d'implémentation

2.1 Création

Afin de simplifier la création d'un utilisateur et ses différentes interfaces associées permettant l'affichage, la modification et la suppression d'un utilisateur, j'ai utilisé la technique de "scaffolding" (*rails generate scaffold User name :string mail :string*).

Par la suite j'ai ajouté de nouveaux champs au modèle (*rails generate model User name :string mail :string date_of_birth :date actual_weigth :integer ideal_weigth :integer size :integer in_diet :boolean like_diet :boolean*).
La base de données contient donc les champs suivants :
- integer "id"
- string "name"
- string "mail"
- date "date_of_birth"
- integer "actual_weight"
- integer "ideal_weight"
- integer "size"
- boolean "in_diet"
- boolean "like_diet"
- string "cv"
- datetime "created_at"
- datetime "updated_at"

Puis, j'ai enchéri le site de quelques nouvelles pages (*rails generate controller Pages home about*).

Après l'ajout de javascript, css, d'un formulaire et d'un peu d'html, nous obtenons un site fonctionnel où nous pouvons ajouter de nouveaux utilisateurs, voir tous leurs champs et les éditer. Grâce à "JQuery" et "Highcharts" il nous est possible d'effectuer des rendus statistiques sous forme de graphes.

2.2 Gems

Configuration minimale :
- **rails 4.2.0** Framework de web-application utilisant le modèle MVC
- **sqlite3** Base de données sqlite

PDF :
- **carrierwave** Upload de fichiers
- **pdfkit** Génération de pdf
- **wkhtmltopdf-binary** Transformation html en pdf (utilisé par "pdfkit")

Validation :
- **validates_timeliness 3.0** Validation des dates

Debug :
- **debugger** Debugger par défaut pour ruby
- **web-console 2.0** Affichage d'une console sur le navigateur

Style :
- **sass-rails 5.0** Rendu de style sass

Autre :

— **turbolinks** Rend plus rapide les liens
— **magic_encoding** Permet l'encoding du texte en utf-8

2.3 Tests

Pour les tests j'ai décidé d'utiliser "rake" en effet le scaffold utilisé précédemment à automatiquement généré des tests fonctionnels. C'est donc pour des raisons de praticité que je me suis tourné vers cette solution (j'aurais cependant pu utiliser "rspec" qui m'aurais permis de faire sensiblement la même chose).
Tous les tests de controleur effectués concernent toutes les pages, cependant les tests du modèle (unitaires) concernent uniquement les utilisateurs. Au final il y a 35 tests dont 14 fonctionnels (j'ai pris la liberté d'en rajouter pour tester le cas ou les champs de la base de données étaient mal renseignés), et 21 unitaires testant les champs entrés dans le formulaire. Les tests unitaires vont vérifier le comportement du serveur si l'utilisateur rentre des données erronées :

— Nom vide
— E-mail vide, déjà existant, ne correspondant pas à l'expression régulière
— Date de naissance correcte (entre 0 et 120 ans)
— Poids actuel renseigné et correct (entre 0 et 600 kg)
— Poids souhaité renseigné et correct (entre 0 et poids actuel)
— Taille renseignée et correcte (entre 0 et 300 cm)
— Mention "actuellement en régime" renseignée
— Mention "souhaitant un régime" renseignée

A noter que tous ces tests sont également effectués en javascript mais sont actuellement désactivés (il suffira de modifier la fonction javascript "fullControl" pour réactiver).

Les tests concernant les booléens sont difficilement faisable, cependant, si une personne s'amuse à entrer autre chose qu'un booléen dans le champ, ruby considérera que la valeur sera à "false".

3 Difficultés rencontrées

Le passage du formulaire en xhtml 1.0 strict valide à été compliqué, en effet avec rails 4.2.0, la fonction form_for créée par défaut deux "input" cachés. Ceux-ci n'étant pas placés dans une balise paragraphe ou div, génerent une erreur xhtml. J'ai pu, grâce aux options "authenticity_token : false" et "enforce_utf8 : false" supprimer ces "input", elles sont malheureusement indispensables au bon fonctionnement du formulaire, je les ai donc rajoutés à la main au sein d'une balise div.

N'étant pas très à l'aise avec les tests, j'ai eu quelques difficultés à les mettre en place. Mais après quelques explications de la part de mes camarades, et un peu de persévérance, j'ai finalement réussi à en implémenter.

4 Conclusion

La réalisation de ce projet m'a permis de découvrir "Ruby on rails". Solution open sources et possédant une communauté plutôt très active, "Ruby on rails" donne la possibilité de créer un code bien organisé (MVC), et ce assez simplement (création de la base de données et quelques fonctionnalités basiques en seulement quelques minutes). La partie concernant les tests m'a permis d'apprendre leurs buts et la manière de les implémenter. Malgré quelques difficultés pour réaliser l'installation de base, je serais prêt à utiliser ce langage lors de mes futurs projets WEB.

5 Annexes

5.1 Commandes

Liste des commandes pouvant être utile pour le lancement du serveur ou des tests.

— **Installation des gems** bundle install
— **Mise à jours des gems** bundle update
— **Lancement du serveur** rails s
— **Lancement de tous les tests** rake test
— **Lancement des tests unitaire** rake test :units
— **Lancement des tests foncitonnels** rake test :functionals

Une fois le serveur lancé, rendez vous à l'adresse suivante "localhost :3000" pour acceder au site web. Pour tout problèmes ou toutes questions relatives à ce projet, n'hésitez pas à me contacter (jason.jamet@gmail.com ou jason.jamet.free.fr rubrique "contact").

6 Bibliographie

— http ://forge.info.univ-angers.fr/ gh
— http ://french.railstutorial.org/chapters/beginning
— https ://www.railstutorial.org/book
— http ://www.github.com/
— http ://www.highcharts.com/
— http ://stackoverflow.com/
— http ://guides.rubyonrails.org/