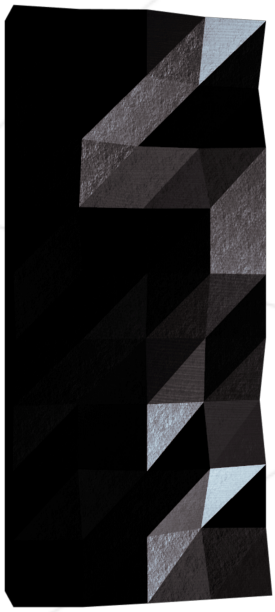

D07 - Ruby on Rails Training

Getting advanced situation and practices

Sta ff 42 bocal@staff.42.fr
Stéphane Ballet balletstephane.pro@gmail.com

Summary: A day on applied practices of web dev. You are put in a position via 'stories' that will lead to the creation of a basic e-commerce site.

Contents

Chapter I

Preamble

Write a user story (scenario, story)

Une histoire utilisateur est une suite d'actions effectuées par un utilisateur de notre application suivies d'une ou plusieurs assertions (test) validant que notre histoire s'est bien déroulée.

Pour expliquer en langage courant, une histoire ressemble à un problème de mathé- matiques dans lequel on a :

- des hypothèses
- un élément perturbateur
- quelque chose à démontrer

Exemple de problème mathématique (version collège) :

- Soit 2 personnes (Pierre et Jean)
- Pierre possède 3 bananes
- Jean possède 2 fois plus de bananes que Pierre quand Jean mange une banane
- alors combien de bananes possède Jean ?

Transformons cet exemple de manière mathématique (version prépa, merci M. Bool) :

- Soit 2 personnes (Pierre et Jean)
- Pierre possède 3 bananes
- Jean possède 2 fois plus de bananes que Pierre quand Jean mange une banane
- alors prouvons que Jean possède 5 bananes.

Si maintenant nous voulons écrire une histoire utilisateur validée par notre système, nous écrirons :

- Soit 2 personnes (Pierre et Jean)
- Pierre possède 3 bananes
- Jean possède 2 fois plus de bananes que Pierre quand Jean mange une banane
- **alors Jean devrait posséder 5 bananes. C'est ce qu'on appelle un test : Jean devrait posséder 5 bananes.**

Si nous écrivons des tests, c'est parce que notre système doit se comporter comme tel. Cela nous permet de valider que notre application réagit bien TOUJOURS de la même manière, même après de nombreux mois / années (donc après de nombreuses modifica- tions).

Docs, sources et references :

- Rspec, cucumber book
- Le wiki du github de cucumber
- Le site de cucumber
- Motivational

Chapitre II

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Si aucune information contraire n'est explicitement présente, vous devez assumer les versions de langages suivantes :
  - Ruby 2.3.0
  - pour d09 Rails > 5
  - mais pour tout les autres jours Rails 4.2.6
  - HTML5
  - CSS 3
- Nous vous interdisons FORMELLEMENT d'utiliser les mots clés while, for, redo, break, retry et until dans les codes sources Ruby que vous rendrez. Tout utilisation de ces mots clés est considérée comme triche (et/ou impropre), vous donnant la note de -42.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas, nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre la procédure de rendu pour tous vos exercices : seul le travail présent sur votre dépot GIT sera évalué en soutenance.
- Vos exercices seront évalués par vos camarades de piscine.
- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicite- ment specifiés par les énoncés des exercices.
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Votre manuel de référence s'appelle Google / man / Internet / ....
- Pensez à discuter sur le forum Piscine de votre Intra !
- Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne

sont pas autrement précisées dans le sujet...

- Par pitié, par Thor et par Odin ! Réfléchissez nom d'une pipe !

Chapitre III

Règles spécifiques de la journée

- All the work of the day will be improved versions of the same application rails.

- Any addition to the supplied Gem fi is prohibited.
- Any global variable is prohibited.
- You must make a seed in accordance with the presented functionalities. When your defense your concealer should be able to view your work.
- rubycritic award you must score at least 89/100
- you must imperatively manage any errors rails error page will be tolerated for correction.

Tips: If you want your corrections are simple, fast and friendly, write tests! It simplifies life for everyone and it is a crucial habit, nay, MANDATORY make for your future.

chapter IV

00 Exercise: MySQL is a bad habit

	Exercise: 00
00 Exercise: MySQL is a bad habit	
Render Folder: *ex* 00 / Files to	
make: acme	
Permitted functions: functions_authorized	
Remarques : n/a	

Commençons par un peu d’admin sys, installez postgresql sur la machine et créez un template et un user. De ce fait vous pourrez creer une application rails nomée ’acme’ et utilisant le gemfile que vous trouverez dans la tarball d07.tar.gz

La commande "rake db :create" doit passer sans erreurs.

Story:

- L’application s’appele "acme". Je veux pouvoir mettre en ligne l’application sur Heroku

Chapitre V

Exercice 01 : You Sir ?

	Exercice : 01 Exercice
01 : You Sir ?	
Dossier de rendu : *ex* 01/ Fichiers	
à rendre : acme	
Fonctions Autorisées : functions_authorized	
Remarques : n/a	

Vous avez une db, dans une app toute fraiche. Hier vous avez crée une authentification à la main, mais dans la vraie vie on ne fait pas ca.
En effet, meme pour un bloginet (c'est un petit blog), un serveur est present, et doit
etre protegé.

Pour ce faire, on utilise la très bonne librairie devise, outre le petit dèj (à ce stade, le café est déjà trop loin), elle gère les authentifications.

Si vous inspectez le gemfile, vous verrez qu'elle est deja présente. Il vous suffit main- tenant de l'installer et de la faire gérer un model "User" de sorte que votre seed puisse executer les commandes suivantes :

```
User.create!(bio: FFaker::HipsterIpsum.paragraph, name: 'admin',

email:'admin@gmail.com',
password:'password',
password_confirmation: 'password' )
```

Stories:

- Un utilisateur peux créer un compte avec un mot de passe (nécessaire), un nom (nécessaire), un email(nécessaire) et une biographie (optionelle).
- Il peut re-editer tout ces champs apres creation du compte via l' application (confi- gurez vos "strong parameters")

Chapitre VI

Exercice 02 : Give me something to sell

	Exercice : 02
Exercice 02 : Give me something to sell	
Dossier de rendu : *ex* 02/ Fichiers	
à rendre : acme	
Fonctions Autorisées : functions_authorized	
Remarques : n/a	

Créez des produits à vendre et des marques, et comme tout produit outre une des- cription tres avantageuse et vantant des mérites que la physique réfute, il faut une image, et c'est là notre problème.

En effet, pour permettre correctement l'upload de fichier d'image, on va utiliser la gem carrierwave (the classier solution), rien de trop compliqué jusque là. Le hic c'est qu'un hé- bérgement gratuit supporte trèèès mal le stock de données qui peuvent être volumineuses.

C'est à cet effet que figure dans le Gemfile, une gem appelée coudinary, vous devez creer un compte gratuit chez cloudinary, et vous octroyer par la un espace de stockage distant spécialisé dans les images et autres medias.

A ce stade votre seed peut executer :

```
50.times do |tm|
    mk = Brand.create!(name: FFaker::Product.brand,
                       avatar: open(FFaker::Avatar.image))

    50.times do |tw|
        Product.create!(name: FFaker::Product.product,
                        pict: open(FFaker::Avatar.image), description:
                        FFaker::HipsterIpsum.paragraph, brand_id: mk.id, price:price.sample)

    end
end
```

Stories:

- En se connectant sur le site de l’application, on voit un catalogue listant des produits.
- Chaque produit est contitué d’un nom, d’une image, d’une description d’une marque et d’un prix.
- Une marque a un nom et une image.
- Peut importe la taille de l’image uploadée la page des produit doit afficher les 2500 images,dans une version thumbnail afin de charger rapidement.
- On peut créer et/ou editer touts les champs en ligne des marques et des produits.
- A terme des roles seront attribués, déterminant qui peux editer quoi.

Chapitre VII

Exercice 03 : Panier

	Exercice : 03 Exercice
03 : Panier	
Dossier de rendu : *ex* 03/ Fichiers	
à rendre : acme	
Fonctions Autorisées : functions_authorized	
Remarques : n/a	

Nous avons besoin de faire un panier, un Cart qui va se voir associer des copies des produits sous forme de CartItem, copiés en OrderItem et rassemblés dans un Order lors de la validation du panier.

Ces objets particuliers ne nécéssitant pas de CRUD à proprement parler, seul un mo- del leur est utile. Cependant, des fonctionalités devront etre placés dans un "Concern" afin de pouvoir inclure des méthodes relatives au panier dans d'autres controleurs comme celui des "Products". Fabriquez vous une méthode current_cart qui se base sur la ses- sion_id.

Remember to destroy objects and unnecessary records, for example, when the cancellation of a basket, the 'CartItems' associates have to be destroyed.

Stories:

- In the catalog page, insert a 'basket' is present.
- items can be added by clicking the "Add to cart" present on each item.
- The user can start an order, fill your basket and close the browser. Upon returning to the site's shopping cart is reloaded.
- The user can increase and decrease the number of each item in the basket.
- The basket of lines ffi Chent one type of item quantity and a button

button less, and the price on the formula: quantity * price.

- The user can cancel a basket and make void with a button.
- Un bouton "Checkout" affiche un recapitulatif de la commande avec le prix total.

Chapitre VIII

Exercice 04 : One panel to rule them all

	Exercice : 04
Exercice 04 : One panel to rule them all	
Dossier de rendu : *ex* 04/ Fichiers	
à rendre : acme	
Fonctions Autorisées : functions_authorized	
Remarques : n/a	

Vous devez maintenant créer un paneau d'administration digne de ce nom. Dans le Gemfile, une gem rails_admin est présente, allez voir la doc et initialisez la.

On peut y accéder (pour l'instant) des qu'on possede un compte. Creez sur la page catalogue un link qui mène au dashboard fraichement disponible.

Stories:

- Un utilisateur enregistré peux se connecter et aller sur le panneau d'administration du site.

- De là, on peut éditer et visualiser l'ensemble des données du site.

Chapitre IX

Exercice 05 : One account to rule them all

	Exercice : 05
Exercice 05 : One account to rule them all	
Dossier de rendu : *ex* 05/ Fichiers	
à rendre : acme	
Fonctions Autorisées : functions_authorized	
Remarques : n/a	

Il est, vous me l'accorderez, plutot inutile de disposer d'un panneau d'administration, si tout le monde a la possiblité d'y faire sa loi, pour peu qu'il ait pris la peine de s'y inscrire.

Pour cette occasion, j'ai inclus la gem cancancan qui facilite la gestion des droits, ainsin que la gem rolify qui elle crée un modele de groupe et y attribue les id des users

La "rolification" doit etre présente aussi dans la seed.

Ces deux outils se combinent plutot bien, mais qu' en est il de l'association avec rails admin ?

Stories:

- Un administrateur crée en meme temps que la db peut attribuer des roles.
- Deux rôles sont disponibles : "admin" et "mod".
- Un administrateur peux TOUT editer.
- Un moderateur peux lui SEULEMENT editer les marques et les produits : creation, modification et suppression
- Un utilisateur simple peux s'identifier mais n'aura aucun de ces privilèges,a moins qu'un admin ne lui attribue un role.

- Aucun "url re-writing" ne permet a quiquonque d’outrepasser les permissions dues à son role

Chapitre X

Exercice 06 : Show me what you got

	Exercice : 06
Exercice 06 : Show me what you got	
Dossier de rendu : *ex* 06/ Fichiers	
à rendre : acme	
Fonctions Autorisées : functions_authorized	
Remarques : n/a	

Creez un compte sur Heroku <3 (oups, ca m'a echappé). Créez un compte disais-je et mettez en ligne votre application.

Stories:

- Notre communauté de beta testeurs sont prets l'application est disponnible sur le web.
- Elle est disponible @ : "https ://votrelogin-acme.herokuapp.com"
- Un administrateur peux TOUT editer.
- An application stand script can fill the application with 2,500 products, 50 brands, 20 users with an "admin" and 5 "models
- All fonctioanlitées put in evidence through the stories of the days are Functional ern online, upload pictures, authentication, roles basket etc etc ....