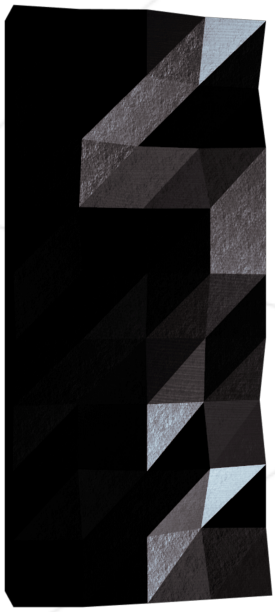


D09 - Formation Ruby on Rails

Temps reél

Stéphane Ballet balletstephane.pro@gmail.com
42 Staff pedago@staff.42.fr

Résumé: Afin de rendre vos pages plus ludiques et rendre l'UX plus agreable, il vous faut agir sur le coté client et celui la ne connait pas Ruby. Vous allez donc avoir à user de JavaScript.
La premiere partie de la journee va être destinée à rendre le `CRUD` plus dynamique avec de l'`AJAX`. Pour la seconde partie, il va être question de temps réel multiclient avec des WebSockets mis votre à disposition au travers d'Action Cable (la nouvelle feature de rails 5).

Table des matières

I	**Préambule**	**2**
II	**Consignes**	**3**
III	**Règles spécifiques de la journée**	**5**
IV	**Francis_1**	**6**
V	**Francis_2**	**8**
VI	**Francis_3**	**9**
VII	**Francis_4**	**10**
VIII	**ChatOne**	**11**
IX	**ChatTwo**	**12**
X	**ChatThree**	**13**

Chapitre I

Préambule

Puisqu'il est question de temps réel et de JavaScript, je vous ai preparé une selection de jeux en JavaScript :

- Wipeout-like
- Text based rpg
- L'indetronable ruine-souris
- Survival Rpg oldschool
- Puzzle game
- co cow coW cOW COW

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 explicitement 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

Il est obligatoire d'utiliser `rails 5` pour toute la journée. Les Gems qui favorisent (ou plutôt font à votre place) l'AJAX sont interdites.

Liste non exhaustive :

- "best_in_place"
- "better-edit-in-place"
- "super_inplace_controls"
- "rest_in_place"
- "on_the_spot"
- "edit_mode"
- "best_in_placeish"
- "crest_in_place"
- ...

Vous devrez justifier tout ajout de Gem non rails-built-in lors de votre correction si le correcteur le demande.

`Aucune` librairie JavaScript additionelle n'est autorisée. Votre `application.js` ne contient que les imports suivants :

```
jquery
jquery_ujs
turbolinks
```

Chapitre IV

Francis_1

	Exercice : 00
Exercice 00 :Francis_1	
Dossier de rendu : $ex00/$	
Fichiers à rendre : `Xnote`	
Fonctions Autorisées :	
Remarques : `n/a`	

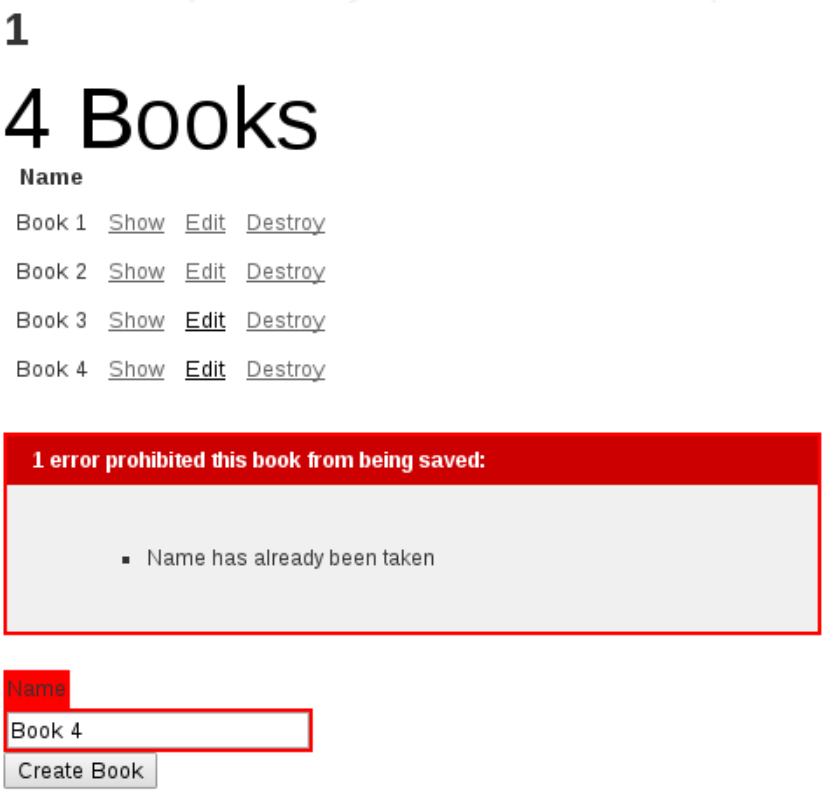
Vous devez créer une biblioteque listant des livres. L'application se nomme : "Xnote". Pour cet exercice, un scaffold est suffisant :

```
rails g scaffold book name
```

Vous devez ensuite :

- Creez une règle de validation d'unicité sur 'name' de 'book'.
- Vous devez permettre l'ajout de books en AJAX :
  - le formulaire doit apparaitre en cliquant sur le 'link_to' qui pointe sur 'new_book_path'
  - que les donnés de la liste de 'books' se mettent à jour lors du submit du formulaire (lequel doit aussi alors disparaitre)
- Vous devez voir aussi les erreurs, comme par exemple vous souhaitez utiliser un nom deja pris.

Exemple :



Et `tout` ca `sans` rafraichissement de page entière. Pour verifier cela, vous devez mettre dans votre layout :

```
##ex00/Xnote/app/views/layout/application.html.erb:
...
<body>
  <% $refresh ||= 0 %>
  <h1><%= $refresh +=1  %></h1>
  <%= yield %>
</body>
...
```

Un peu a la maniere d'un mouchard, votre page sera agrémentée d'un "refresh count" qui doit rester a `1`.

Chapitre V

Francis_2

	Exercice : 01
Exercice 01 :Francis_2	
Dossier de rendu : *ex*01/	
Fichiers à rendre : `Xnote`	
Fonctions Autorisées :	
Remarques : `n/a`	

Maintenant que vous avez compris le principe, il vous sera facile de faire de meme avec le "link_to" pointant sur la méthode destroy.
Cliquer sur celle ci doit, apres le popup de confirmation, supprimer de la DB l'entrée et mettre à jour la liste.

`la variable globale $refresh doit rester à 1`

Chapitre VI

Francis_3

	Exercice : 02
Exercice 02 :Francis_3	
Dossier de rendu : *ex*02/	
Fichiers à rendre : `Xnote`	
Fonctions Autorisées :	
Remarques : `n/a`	

Une fois n'est pas coutume, c'est au tour de la methode "edit" de fonctionner sans rechargement de page..

Vous devez insérer le formulaire à la place de la ligne de tableau correspondant au 'book' edité et pour autant permettre aux erreurs de s'afficher. Ainsi soumettre un nom de livre en double affiche les erreurs de validations.

`la variable globale $refresh doit encore rester à 1!`

Chapitre VII

Francis_4

	Exercice : 03
Exercice 03 :Francis_4	
Dossier de rendu : $ex03/$	
Fichiers à rendre : `Xnote`	
Fonctions Autorisées :	
Remarques : `n/a`	

Eloignons nous un peu du CRUD pour cet exercice.
Réalisez en tete de page un compte du nombre (total) de livres. Il doit être mis à jour à chaque modification de celui ci, que ce soit un ajout de livre ou une suppression.

`la variable globale refresh doit toujours rester à 1!`

Chapitre VIII

ChatOne

	Exercice : 04
Exercice 04 :ChatOne	
Dossier de rendu : $ex04/$	
Fichiers à rendre : `Chat`	
Fonctions Autorisées :	
Remarques : `n/a`	

Jusqu'alors, nous n'avions eu qu'à produire des partie en AJAX, qui n'est autre qu'un pattern. Maintenant, dirigons nous ensemble vers le multiclient.

Faites une application de chat qui comporte des messages d'utilisateurs authentifiés avec la Gem 'devise', qui apparaisent en temps réel chez tous les utilisateurs connectés et qui se nomme très justement : "Chat".

Un conseil, ActionCable gère ca tres bien, servez vous en.

`Ouvrez plusieurs fenêtres en mode 'incognito' et authentifiez vous avec des logins différents.`

Concevez cette application dans l'optique de gérer `beaucoup` de trafic en temps réel, il `faut` implémenter un système de mise en buffer de tâches.

`les 'ApplicationJob'ou 'Active Job' ne sont pas uniquement réservés à nos chers amis canidés`

Chapitre IX

ChatTwo

	Exercice : 05
Exercice 05 :ChatTwo	
Dossier de rendu : $ex05/$	
Fichiers à rendre : `Chat`	
Fonctions Autorisées :	
Remarques : `n/a`	

En réutilisant la base de l'application précédement crée, implementez le concept de ChatRoom : des salons ou ce qui y est dit y reste. Vous allez faire en sorte que, dès qu'il est identifié, un utilisateur peut créer une Chatroom.

Cet utilisateur est considéré comme seul et unique créateur de cette ChatRoom. Supprimer cet utilisateur annihile par la meme occasion tous les salons qu'il a créé mais aussi tous les messages qui s'y trouvent. Les messages postés apparaissent uniquement dans le salon ou ils ont été crée initialement.

Chapitre X

ChatThree

	Exercice : 06
Exercice 06 :ChatThree	
Dossier de rendu : $ex06/$	
Fichiers à rendre : `Chat`	
Fonctions Autorisées :	
Remarques : `n/a`	

Toujours dans la meme application "Chat", créez un systéme de notifications, representé dans `Chat/apps/views/layouts/application.html.erb` par une liste où une entrée est ajoutée à chaque nouveau message **si** l'auteur du message est différent de l'utilisaeur connecté. C'est logique : vous ne voyez pas les notifications de vos propres messages. Ceci doit être appliqué sur tous les salons auxquel l'utilisateur est connecté.

En temps réel, vous devez en permanence et ce sur toutes les pages afficher une liste de notifications et un compte de leur nombre total. `AUCUNE` ne peut etre de vos propres messages. Cela doit fonctionner en multi-client et en simultané.

Et pendant que vous y êtes, mettez y un son aussi à cette notification.