



Ruby on Rails Training - 3

Final

Summary: In order to make your pages more fun and the UX more pleasant, you need to work on the client side, but it doesn't know Ruby. This is why you will have to use

[JavaScript](#).

*The first part of the day will aim to make the **CRUD** more dynamic with some **AJAX**. The second part will be about multi-user real time with WebSockets provided through **ActionCable** (a new feature in Rails 5).*

Version: 1

Contents

I	Preamble	2
II	General rules	3
III	Today's specific instructions	4
IV	Francis_1	5
V	Francis_2	7
VI	Francis_3	8
VII	Francis_4	9
VIII	ChatOne	10
IX	ChatTwo	11
X	ChatThree	12
XI	Submission and peer-evaluation	13

Chapter I

Preamble

Since we're going to deal with real time and JavaScript, I've made a list of games in JavaScript:

- [Wipeout-like](#)
- [Text based rpg](#)
- [the ultimate mouse killer](#)
- [Oldschool Survival Rpg](#)
- [Puzzle game](#)
- [co cow coW cOW COW](#)

Chapter II

General rules

- Your project must be realized in a virtual machine.
- Your virtual machine must have all the necessary software to complete your project. These softwares must be configured and installed.
- You can choose the operating system to use for your virtual machine.
- You must be able to use your virtual machine from a cluster computer.
- You must use a shared folder between your virtual machine and your host machine.
- During your evaluations you will use this folder to share with your repository.
- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a 0 during the evaluation.
- We encourage you to create test programs for your project even though this work **won't have to be submitted and won't be graded**. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.

Chapter III

Today's specific instructions

You must use `rails 5` all day long. Gems facilitating the AJAX (or just making it for you) are prohibited.

Non-exhaustive list:

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


You will have to justify the use of any non rails-built-in Gem during the evaluation if your assessor requires it.

ANY additional JavaScript library is prohibited. Your `application.js` doesn't contain any of the following imports:

```
jquery
jquery_ujs
turbolinks
```

Chapter IV

Francis_1

	Exercise 00
Exercise 00:Francis_1	
Turn-in directory : <i>ex00/</i>	
Files to turn in : Xnote	
Allowed functions :	

You must create a library listing books. The application is named "Xnote". For this exercise, a scaffold is enough:

```
rails g scaffold book name
```

Then you will:

- Create a unicity validation rule on 'name' of 'book'.
- Allow the add of books in [AJAX](#):
 - the form will appear clicking the 'link_to' pointing on 'new_book_path'
 - the 'books' list data are updated when the form is submit (it must also appear)
- See errors. For instance, if you want to use a name that's already in use.

Example :

1

4 Books

Name

Book 1 [Show](#) [Edit](#) [Destroy](#)

Book 2 [Show](#) [Edit](#) [Destroy](#)

Book 3 [Show](#) [Edit](#) [Destroy](#)

Book 4 [Show](#) [Edit](#) [Destroy](#)

1 error prohibited this book from being saved:

- Name has already been taken

Name

Book 4

Create Book


And everything should work without refreshing the whole page. To verify that, you must put in your layout:

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

A bit like a bug, your page will include a "refresh count" that must stick to 1.

Chapter V

Francis_2

	Exercise 01
Exercise 01:Francis_2	
Turn-in directory : <i>ex01/</i>	
Files to turn in : Xnote	
Allowed functions :	


Now you know the drill, you will be able to do the same with the "link_to" pointing on the destroy method.
Clicking on it should, after a confirmation popup, delete the entry from the DB and update the list.



the global variable must always be 1!

Chapter VI

Francis_3

	Exercise 02
Exercise 02:Francis_3	
Turn-in directory : <i>ex02/</i>	
Files to turn in : Xnote	
Allowed functions :	

And now, this is the "edit" method's turn to work without reloading the page...


You must insert the form as the first line of the table matching the edited 'book' and allow errors to be displayed. When submitting a book already submitted, validation errors must appear.



the global variable must always be 1!

Chapter VII

Francis_4

	Exercise 03
Exercise 03:Francis_4	
Turn-in directory : <i>ex03/</i>	
Files to turn in : Xnote	
Allowed functions :	


Let the CRUD aside for this exercise.
In the header, create a count for the (total) number of books. Is must be updated each time the DB is modified, whether a book is added or deleted.



the global variable must always be 1!

Chapter VIII

ChatOne

	Exercise 04
Exercise 04:ChatOne	
Turn-in directory : <i>ex04/</i>	
Files to turn in : Chat	
Allowed functions :	

Until now, we only had to produce parts in AJAX, which is just a pattern. Now, let's head towards the multi-user together.

Make a chat application that includes user messages authenticated with the 'devise' Gem, that appear in real time to all the logged-in users. It will surprisingly enough be named: "Chat".

A piece of advice: ActionCable is very good at managing this. Use it!



Open several windows in 'invisibility' mode and register with different logins.


Design this application so it can deal with a lot of real-time traffic. You must implement a system that will set the tasks in buffer.



the 'ApplicationJob' or 'Active Job' were not just made for our canid friends.

Chapter IX

ChatTwo


	Exercise 05
Exercise 05:ChatTwo	
Turn-in directory : <i>ex05/</i>	
Files to turn in : Chat	
Allowed functions :	

Using the base of the application you've just created, implement the ChatRoom concept: rooms that will keep what comes in. You will make sure that as soon a user is logged in, they can create a chatroom.

This user is considered the sole creator of this ChatRoom. Deleting this user also deletes all the rooms they have created and the messages they include. Posted messages only appear in the room they were initially created in.

Chapter X

ChatThree

	Exercise 06
Exercise 06:ChatThree	
Turn-in directory : <i>ex06/</i>	
Files to turn in : Chat	
Allowed functions :	

Always in the same "chat" application, create a notification system represented in `Chat/apps/views/layouts/application.html.erb` by a list where an entry is added for each new message **if** the message's author is not the logged in user. It's just logic: you're not gonna get notifications for your own messages. This must be applied to all the chatrooms the user is logged in to.

In real time, you will always and on every page, display a list of notifications and a count of the total number. You can get **NO** notification of your own messages. This must work in multi-user and simultaneously.

As long as you're at it, why not adding a little sound to your notification.

Chapter XI

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.



The evaluation process will happen on the computer of the evaluated group.