SCALE FOR PROJECT PISCINE RUBY ON RAILS (/PROJECTS/42CURSUS-PISCINE-RUBY-ON-RAILS) / DAY 03 (/PROJECTS/42CURSUS-PISCINE-RUBY-ON-RAILS-DAY-03)

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere.msk.21-school.ru:vogsphere/intra-uuid-c9167b35-3902-45
```

Introduction

For the smooth running of this evaluation, please respect the following rules:

- Remain polite, kind, respectful and constructive whatever happens during this conversation. It's a matter of confidence between you and the 42 community.

- Highlight the potential problems you 've had with the work you're presented to the person or the group you're grading, and take the time to talk about and discuss those issues.

- Accept the fact that the exam subject or required functions might lead to different interpretations. Listen to your discussion partner's perspective with an open mind (are they right or wrong ?) and grade them as fairly as possible.
42's teaching methods can make sense only if peer-evaluation is taken seriously.

Guidelines

- You must only evaluate what you will find in the student's or group's GiT repository.

- Take the time to check that the GiT repository matches the student or group and the project.

- Double check that no malicious alias was used to mislead you and make you grade something different from the official repository content.

- If a script supposed to help evaluate the exam is supplied by either side, the other side will have to strictly check it to avoid nasty surprises.

- If the evaluating student has not yet taken this project, they will have to read the exam subject in its entirety before starting the evaluation.

- Use the flags available on this grading system to signal an empty or non-funcional project, a norm flaw, cheating, etc. In that case, evaluation stops and final grade is 0 (or -42 if it's a cheating problem). However, if it's not a cheating problem, you are invited to keep talking about the work that has been done (or not done, as a matter of fact) in order to identify the issues that lead to this stalemate and avoid it next time.

- You must stop grading when one exercise is not correct, even if the other ones are.

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/46589/en.subject.pdf)

d03.tar.gz (/uploads/document/document/8254/d03.tar.gz)

Foreword

This section is dedicated to the evaluation start and the checking of prerequisites. It's not graded, but if something's wrong or a condition is not met, here or anytime during the evaluation, the grade is 0 and a flag can be ticked if necessary.

Observing instructions

- The repo contains the evaluated student's or group's work.
- The evaluated student or group can explain their work anytime during the evaluation.
- General and specific instructions of the day will be observed during the whole evaluation.
- If you the following keywords have been used: for, while ou until, evaluation stop and give the student -42. Tick the "Cheat" flag. This will go for the whole evaluation.

✓ Yes ✗ No

Ruby Rails Training D03

- If you find an error in an exercise, the whole exercise is incorrect. No half-measure, no "slight error". The work is either perfect, either not. - The student must have provided a Gem or a rails project for the ex03. - Display of the pages must be managed: no strange behaviour, no special character (no accent) the display would not manage, etc. We want clean, or beyond.

ex00 I like

- The Gem is named deepthought.
- The test library is minitest.
- The license must be '0.0.1'.
- The "grep -Hrn 'TODO' –color=always" command doesn't return ANYTHING when executed at the root of the Gem.
- The "gem build deepthough.gemspec" is running properly when executed at the root of the Gem.
- The test checks that Deepthought.new returns the expected object and doesn't return any fail. IF the object is not the expected nature (that is :Deepthought) the test must fail!
- The test checking the return value of both "respond" method cases is also present, passes and returns a fail if it has to (vérifier l'énoncé français, il y a un truc pas très clair) (modify the class code to verify the test efficiency). The exercise is incorrect is one of the requirements is not perfectly executed.

✓ Yes ✗ No

ex01ft_wikipedia.

- As in previous exercise: all the required tests are present and run OK.
- As in previous exercise, the Gem goes "build"
- You can also install: "gem install ft_wikipedia.gem", and in an interactive console, "require 'ft_wikipedia'" returns true.
- All the errors suggested in the wording are managed.
- When you execute "require 'ft_wikipedia'" in irb or pry, and call the class as follows: 'Ft_wikipedia.search("Kiss")', the return is identical to the example in the wording. The exercise is incorrect if one of the requirements is not perfectly executed.

✓ Yes ✗ No

ex02 TDD

- The "taillste_test.rb" of the repo is IDENTICAL to the one in the wording.
- The rake command at the root of the folder launches the test.
- They all pass thanks to otherworldly efforts? Congratulations! The exercise is validated.

✓ Yes ✗ No

ex03 Rails

- The repo is a Rails project.
- Validation of the Install: on the evaluated student's session, the "rails -v" and "rails new toto" commands respond properly. The first one displays the rails' version. The second one displays how it works.
- You can run a server at the root of the repo ("rails server").
- Once the server is running, you access the application via: "http://localhost:3000/" and its content is "Hello World!" as title of the first level (h1). The exercise is incorrect if one of the requirements is not perfectly executed.

✓ Yes ✗ No

Ratings

Don’t forget to check the flag corresponding to the defense

✓ Ok

★ Outstanding project

Empty work

No author file

Invalid compilation

Norme

Cheat

Crash

Incomplete group

Forbidden function

Conclusion

Leave a comment on this evaluation

проверка проходила онлайн. сделано

Finish evaluation