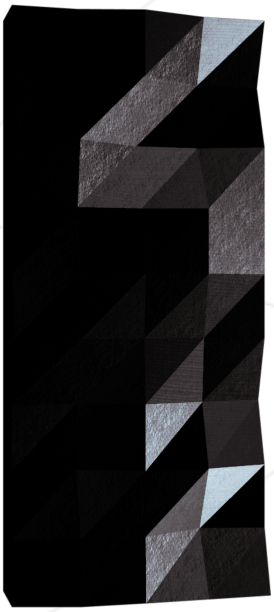

D02 - Ruby on Rails Training

Classes and Inheritance Exceptions

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

Abstract: In this day D02, you will learn to make up classes and get to the heart of OOP, if your exercises seem too wordy is that your code is not DRY enough!

Contents

I Preamble 2

II instructions 3

III Specific Rules of the day 5

IV 00 Exercise: HTML 6

V 01 Exercise: Raise HTML 8

VI Exercise 02: Rescue HTML 9

VII Exercise 03: Elem 11

VIII Exercise 04: Dejavu 13

IX Exercise 05: Validation 15

Chapter I

Preamble

Besides being awesome, Ruby is beautiful !

This guide (written by Bozhidar Batsov) has emerged as a convention of programmed internal Ruby tion to his business. After a while, it seemed to him that this work could be interesting for community members ruby in general and the world did not really need an internal programming convention more. But the world could certainly benefit of a set of practices, idioms and style advice for the Ruby programming.

From the start of this guide, he received a lot of feedback from members of the ex ceptional Ruby community worldwide. Thank you for all the suggestions and support! Together we can create a beneficial resource to all develop- Ruby fears here today.

- The style is what separates the good from the excellent.
- The very widely used Rubocop

Chapter II

Instructions

- Only this page serve as a reference: Do not firm the hallway noise.
- The subject can change up to an hour before rendering.
- If no contrary information is explicitly present, you must assume the following language versions:
  - Ruby 2.3.0
  - for d09 Rails> 5
  - but for all the other days rails 4.2.6
  - HTML5
  - CSS 3
- We prohibit EXPRESSLY using keywords while, for, redo, break, retry and until in Ruby source code that you will make. Any use of these keywords is considered cheating (and / or unfit), giving you the score of -42.
- The exercises are precisely ordered from simple to more complex. In any case, we do not bear attention or take into account a complex exercise if a simpler exercise is not very successful.
- Attention to the rights of your files and your directories.
- You must follow the rendering process for all your exercise: only the present work on your GIT repository will be evaluated in defense.
- Your exercises will be evaluated by your pool mates.
- You should leave in your repertoire no other file than those explicitly specified by the forward drills.
- You have a question ? Ask your neighbor to the right. Otherwise, try your left neighbor.
- Your reference manual called Google / man / Internet / ....
- Think discuss on the forum of your pool Intra!
- Read the examples carefully. They might require things that

not otherwise specified in the subject ...

- Please, by Thor and Odin! Refléchissez dammit!

chapter III

Specific Rules of the day

- All files will be returned with an appropriate shebang and warning flag.
- No code in the global scope. Make functions or classes!
- Each file must have made a series of tests to prove its good functioning:

```
if $ PROGRAM_NAME == __FILE__
   ## your code here end
```

- Each file is to be tested in an interactive console (irb or pry choice) like this:

```
require_relative "name \ _of \ _fichier.rb"
>
```

- No authorized import, except for those explicitly mentioned in the "Permitted functions" of the cartridge each year.

chapter IV

00 Exercise: HTML

	Exercise: 00 00
Exercise: HTML	
Render Folder: *ex* 00 / Files to make: ex00.rb	
Permitted functions: n / A	
Remarks: n / A	

Take a class html able to create and populate a file HTML. To do this, you need to Implement:

- A constructor that takes parameters as a file (without extension):
  - which a method called head
  - that assigns the title file the instance variable @ page_name
- A attr_reader for @page_name
- A method Head which must aposer head of a file header html followed by a valid start tag body.
- a method "dump" which takes a string parameter and starts following the "body" tag our string surrounded by "p" tags.
- a method "finish" that ends the file a body closing tag

All inserts must be in the form of lines.

In a console **ruby,** we must get this:

```
>   require_relative "ex00.rb" => true

>   Html.new a = ( "test")
=> # <Html: @ 0x00000001d71580 page_name = "test">
>   10.times {| x | a.dump ( "titi_number # {x}")}
=> 12
>   a.finish
=> 7
```

And in our shell:

```
cat -e test.html <!
DOCTYPE html> $ <html>
$ <head> $

<Title> Test </ title> $ </ head> $
<body> $

    <P> titi_number0 </ p> $ <p>
   titi_number1 </ p> $ <p>
   titi_number2 </ p> $ <p>
   titi_number3 </ p> $ <p>
   titi_number4 </ p> $ <p>
   titi_number5 < / p> $ <p>
   titi_number6 </ p> $ <p>
   titi_number7 </ p> $ <p>
   titi_number8 </ p> $ <p>
   titi_number9 </ p> $ </ body> $ $>
```

Use as in the example a loop to fulfill its fi le **MUST** function.

chapter V

01 Exercise: Raise HTML

	Exercise 01 Exercise 01:
HTML Raise	
Render Folder: *ex* 01 / files to make: ex01.rb	
Permitted functions: n / A	
Remarks: n / A	

For this exercise, you will reuse the code ex00 and incorporate error handling, throw exceptions when the behavior requires, thus avoiding invalid HTML pages pro- duction and / or wacky. All the following exceptions must be reproduced accurately, replacing < filename> the file name which creates the error:

- Create a file that already exists (same name) must throw an exception " A file named <filename> already exist ".
- Write text with dump in a file having no tag body or- vrante should throw an exception " There is no body tag in <filename> ".
- Write text with dump after a closing body tag must throw the exception:
  "Body HAS already been closed in <filename>".
- Close the body with finish while the closing tag body is already present must throw the exception: "< filename> has already been closed. "

```
>   require_relative "ex01.rb" => true

>   Html.new a = ( "test")
=> # <Html: @ 0x0000000332b9c0 page_name = 'test'>
>   Html.new a = ( "test")
RuntimeError: test.html already exist! /ex01.rb:15:in from "head"

>   a.dump ( "Lorem_ipsum")
=> Nil
>   a.finish
=> Nil
>   a.finish
RuntimeError: test.html HAS already been closed from / ex01.rb: 39: in "finish"
```

chapter VI

Exercise 02: Rescue HTML

	Exercise 02 Exercise 02:
Rescue HTML	
Render Folder: *ex* 02 / Files to make: ex02.rb	
Permitted functions: n / A	
Remarks: n / A	

Now that you have problem behaviors in their hands, we will profit.

To save process execution correcting shots, with a affichage specific errors, and solutions to ensure FUNCTIONING.

Create two classes: **Dup_ the fi** and **Body_closed** who will inherit **StandardError.** They will both contain implementations of " show_state "," correct " and "Explain".

- show_state a affiche the state before correction
- correct corrects the error
- explain a affiche the state after correction

When an error creating file because already now the fi Dup_ the exception is raised. She must :

- a ffi expensive list files similares (with full pwd)
- create a new file by adding '.new' after the extension, several times if necessary: test.html, test.new.html, test.new.new.html etc etc ... Example:

```
A file named <filename> was already there: /home/desktop/folder_2/<filename>.html .new Appended in order to create requested file:
/home/desktop/folder_2/<filename>.new.html
```

When you try to write AFTER the closing tag of body, the Body_closed exception is thrown and must:

- a ffi expensive line and its number in the file
- remove the so-called tag, insert the text and put a closing tag after example:

```
In <filename> body was closed:
    >   In: 25 </ body>: text has-been inserted and tag Moved at end of it
```

chapter VII

Exercise 03: Elem

	Exercise 03 Exercise
03: Elem	
Render Folder: *ex* 03 / Files to	
make: ex03.r	
Permitted functions: n / A	
Remarks: n / A	

Now is the time to change method. Your first engine test HTML is conclusive and prometeur but it is now question of pushing the paradigm of the subject a little further.

You will now create a class to represent your HTML so that a method to_s at his instance a ffi che code HTML generated.

So, with his method add_content, the class Elem will be able to have content for another instance of Elem.

This architecture permettera use these:

```
html = Elem.new (.....) head =
Elem.new (......) body = Elem.new (....)

title = Elem.new (Text.new ( "bla bla")) head.add_content (title)

html.add_content ([head, title, body]) puts html
```

To ensure proper implementation, we provide a file in the folder test EX02 / in the tarball d02.tar.gz present with this.

It recapitulates:

- A class Elem with as a construction parameter, a tag type, an array of content, a type of tag (orphaned or not), and a Hash permetant information to implement the in-tag (src, style, data ...).

- A class text that is built with a simple String parameter.
- A **overload** Method to_s.
- Running the test script must pass **IN-TE-GRA-LEMENT.**

chapter VIII

Exercise 04: Dejavu

	Exercise 04 Exercise
04: Dejavu	
Render Folder: *ex* 04 / Files to make: ex04.rb	
Permitted functions: n / A	
Remarks: n / A	

Congratulations! You are now able to generate any element that HTML
and content. However, it is a bit heavy to generate each item by stating where each attribute to each instantiation. This is an opportunity to use inheritance to other small simplest classes of use.

Perform the following classes in the drifting of Elem:

- Html, Head, Body
- title
- meta
- img
- Table
- Th, Tr, Td
- Ul, ol, Li
- H1, H2
- P
- div
- span
- Hr
- Br

Your code should execute these commands without errors:

```
Html.new puts ([Head.new ([Title.new ( "Hello ground!")])
        Body.new ([H1.new ( "Oh no, not again!")
        Img.new ([], { 'src': 'http: //i.imgur.com/pfp3T.jpg'})])])
```

And ffi expensive:

```
<HTML>
<Head>
<Title> Hello ground! </ Title> </ Head> <Body>


<H1> Oh no, not again! </ H1>
<Img src = 'http: //i.imgur.com/pfp3T.jpg' /> </ body> </ html>
```

If you feel that exercise is daunting is that there may be another solution;)

chapter IX

Exercise 05: Validation

	Exercise 05 Exercise 05:
Validation	
Render Folder: *ex* 05 / Files to make: whereto.rb	
Permitted functions: n / A	
Remarks: n / A	

Despite real progress in your work, you would like that everything is a little cleaner, a little framed, you are like that: you like the constraints and challenges. So why not impose a standard structure for your documents HTML? Commencez by copying the classes of the two previous years in the record of this exercise.

Create a class Page which the manufacturer must take as a parameter an instance of a class inheriting Elem. Your class Page must implement a method isValid ()
which must return true if all the following rules repectées and false if not :

- If during the course of the tree, a node is not type html, head, body, title, meta, img, table, th, tr, td, ul, ol, li, h1, h2, p, div, span, hr br or
text, the shaft is invalid.
- html must contain exactly Head, **then** a Body.
- Head should only contain a single title and only this Title.
- Body and div should contain elements of the following types: H1, H2, Div, table, ul, ol, Span, or Text.
- Title, H1, H2, Li, Th, Td must not contain a single text and only this
Text.
- P should contain only Text.
- span should contain only text or some P.
- IU and ol must contain at least one Li and only Li.

- **Tr must contain at least one Th or td and only Th or some Td. The Th and the td must be mutually** exclusive.
- Table should contain only Tr and only Tr.
- **img: must contain a field src and its value is a Text.**

Demonstrate how your class Page by tests of your choice by many sufficient to cover all the features. For example, the execution of:

```
if $ PROGRAM_NAME == __FILE__
    foo = Html.new ([Head.new ([Title.new (Text.new ( "Hello ground!"))]) Body.new ([H1.new (Text.new ( "Oh no, not
    again! ")), Img.new ([], { 'src': Text.new ( 'http://i.imgur.com/pfp3T.jpg')})])]) = Page.new test (foo) test.is_valid?



    tata Html.new = ([Head.new ([Title.new (Text.new ( "Hello ground!"))])
                Body.new ([H1.new (Text.new ( "Oh no, not again!")), Img.new ([], { 'src': Text.new ( 'http://i.imgur.com
      /pfp3T.jpg ')})])]) = test2 Page.new (tata) test2.is_valid? end
```

MUST a ffi expensive:

```
Currently Evaluating has Html:
-   root element of the type "html"
-   Html -> Must contains a Head DNA has after-Body Head it is OK

Evaluating a multiple node Currently Evaluating a
Text:
- Text -> Must contains a single text string is OK happy
Evaluating node has multiple Currently Evaluating a Text:


- Text -> Must contains a single string Text happy is OK

Currently Evaluating has Img: Img happy is OK

                              FILE IS OK
```