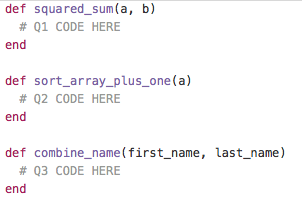
ITU SWE 610 Ruby on Rails MidTerm Exam 04/21/2016

This exam is open book, open notes, open internet. Please do not collaborate with other students as I have tools to identify plagiarism specially in programming questions. You are suppose to complete this exam on your own. This Exam is timed at 4 hours. You should be able to complete this exam in 4 hours. You can take 30 minutes extra if needed. So after 4.5 hours I will start deducting points at the rate of 1 point per hour late.  Your time will start as soon as you download this exam from EMS. If you use the textbook, the internet, or other external sources, give references.

Name: Nikita Sonthalia ITU Student ID: 89679

Start Time: 10.27 AM End Time:

1. This Question involves writing scripts in ruby in order to get a feel for the Ruby programming language.



1. In above Ruby Code there is a function called check\_squared\_sum. This function takes two integers, a and b, and calculates the sum of a and b, then returns the square of the sum.
2. The second method you have to fill in is sort\_array\_plus\_one. This method takes in an array of integers, sorts it, then increments every element by 1, and returns it. It does not matter if this method is destructive or not.
3. The third method takes in two strings, a first\_name and last\_name. It will return the first\_name concatenated with the last\_name, along with a space in between.

Example:

combine\_name 'Howard', 'Chen'

* + 'Howard Chen'

1. Following exercises reinforces your knowledge into Ruby Programming. Please review attached Ruby code for Number Guessing Game we discussed in the class. Ruby Code can be found at following location: <https://www.dropbox.com/s/k3agw2ie3unbt3r/NumberGuess.rb?dl=0>

Run the above code as shown in the class and experience the Number Guessing Game. Based on the questions below make improvements to number guessing game, the Ruby Number Guessing Game.

1. Currently, the game allows players to play as many times as they wish. It does not provide any feedback on how the players are doing, however. Modify the game so that it keeps track of the number of games played as well as the average number of guesses made per game.
2. The game challenges the player to guess a randomly generated number between 1 and 100 in as few guesses as possible. Make the game more challenging by increasing the range of numbers to between 1 and 1000.
3. The game lets players know whether their guesses are too high or too low. It also rejects any guesses outside of its supported range of numbers, such as negative numbers, numbers greater than 1000, or alphabetic or special characters. Modify the game so that it notifies players when invalid guesses have been made and reminds players of what constitutes valid input.
4. Currently, the game allows players to make an unlimited number of guesses. Therefore, players cannot lose the game. Modify it so that players are allowed a maximum of 10 guesses, after which the game is declared lost.
5. You should thoroughly test all your Ruby scripts to make sure they are running as expected. With the Number Guessing game, that means playing the game repeatedly. To make this easier on you, add a hidden “cheat” to the game that allows you to display the game’s number.

Solve each questions above and submit your answers by pushing it to github repository as SolutionA, SolutionB etc. Post your github link. Please explain how you went about making the change and what data structures, programming concepts you used to solve the problems.

1. Consider the following Ruby code:

def method1

x = 11

method2 do |x|

puts x

end end

def method2

x = 22

yield 33 end

def method3

x = 11

method2 do |y|

puts x

end end

(a) What output (if any) is generated when method1 is called?

Ans : 33

(b) What output (if any) is generated when method3 is called?

Ans: 11

(c) Give 2 examples of how Rails takes advantage of meta-programming facilities in the Ruby language.

Ans: 1. “validates\_format\_of” causes new validations to be added to a model.

2. Rails introspect the database and creates methods such as “find\_all\_by\_name”.

(d) In your opinion, which programming language is better, Ruby or Javascript? List 2 specific reasons to justify your choicce.

Ans: I think ruby is better

1. Ruby has more features.
2. Ruby provide meta-programming facilities.

3. Label each of the tasks below with “Model”, “View”, or “Controller” to indicate where that task would typically be implemented in a Web application using an MVC architecture.

(a)  Validate form data Model

(b)  Make sure a user is logged in Controller

(c)  Invoke the link\_to method View

(d)  Return a “redirect” to the browser Controller

(e)  Define an event handler for a custom form element View

(f)  Generate a new session token Controller

(g)  Invoke the find\_all\_by\_name method Controller

(h)  Create a “salt” for a password Model

1. Write a Ruby class definition that meets the following criteria:
   1. class is called Troll
   2. class has publicly accessible attributes ugliness, smelliness, and strength
   3. upon instantiation, an object of this class has a member variable, a String, called grunt, whose initial value is "UNGAH" (that's pronounce "oon-guh").
   4. class has an instance method called speak() that prints the value of the instance variable grunt 42 times
   5. class has an instance method called reverse() that prints the value of the instance variable grunt backwards
   6. class has a static/class method called propagate(), which returns a Troll instance whose grunt attribute is "eegah"
   7. Imagine a Troll instance fred, which, when the following method is called:
   8. fred.respond\_to?("fight")  returns true. What is missing from your class definition in order for this example to be accurate?
2. Short Answer Questions:
3. Does the respond\_to?() method illustrate object-oriented polymorphism? If so, in what manner?

Ans: Returns true if obj responds to the given [method](http://apidock.com/ruby/Object/method)

1. According to Ruby conventions, what kind of value would you expect to receive from a method that ends in a question mark (?) ?

Ans: boolen datatype

1. According to Ruby conventions, what is the difference between pairs of methods like do\_this and do\_this! (notice the bang) ?

Ans: The general convention is to use do..end for multi-line blocks and curly braces for single line blocks

1. Briefly explain Ruby's type system. What is it (by name)? What does it mean?

Ans:Ruby is dynamically AND strongly typed. A **dynamically** typed language is a language where the type of a variable can be altered at any time and  **strongly** typed language is a language that is being strict about what you can do with your typed variables.

1. What type of Ruby object does the following expression yield? %w( master rails and then try another framework you'll never go back)

Ans: Used for single-quoted array elements.

1. Given an array of strings called @happy\_places, would these two snippets of code do the same thing?

Ans: yes

1. @happy\_places.each do |happy\_place|
   1. puts happy\_place end
2. @happy\_places.each {|hp| puts hp}
3. Given a function that needs to return a value to its caller, does the function need an explicit return statement? If so, explain why. If not, then what can you always expect a Ruby function to return?

And:Ruby always return the last statement value.

6. The Hash class supports 50 methods, which is many more than can be covered in this chapter. To round out your understanding of arrays, visit http://ruby-doc.org/core/classes/Hash.html and view the documentation for the various methods. Then complete the following exercises using IRB:

1. Create a hash named Accounts and add the following items to it:
   1. 10001, AFord
   2. 20001, JAlex
   3. 30001, AJolie

2.2.1 :004 > account={10001=>"AFORD",20001=>"JAlex",30001=>"AJolie"}

=> {10001=>"AFORD", 20001=>"JAlex", 30001=>"AJolie"}

1. Use the Hash class’s delete method to remove the key-value pair with a key of 30001 from the Accounts hash.

2.2.1 :005 > account.delete(30001)

=> "AJolie"

1. Use the Hash class’s inspect method to verify the contents of the hash.

2.2.1 :006 > account.to\_s

=> "{10001=>\"AFORD\", 20001=>\"JAlex\"}"

1. Use the Hash class’s invert method to convert the Accounts hash’s keys to values and convert its value to keys.

2.2.1 :007 > account.invert

=> {"AFORD"=>10001, "JAlex"=>20001}

1. Use the Hash class’s has\_key? method to determine if any of the following keys are stored in the Accounts hash: JAlex, 10001.

2.2.1 :008 > account.has\_key?("10001")

=> false

2.2.1 :009 > account.has\_key?("JAlex")

=> false

2.2.1 :010 > account.has\_key?("AFORD")

=> false

1. Use the Hash class’s size method to display a count of the number of key-value pairs stored in the Accounts hash.

2.2.1 :011 > account.size

=> 2

1. Use the Hash class’s clear method to remove all items from the Accounts hash and then display the contents of the hash.

2.2.1 :013 > account.clear

=> {}

1. Use the Hash class’s inspect method to verify that the array is now empty.

2.2.1 :014 > account.to\_a

=> []

7. String substitution is an important feature of regular expressions. The following exercises are designed to give you the opportunity to demonstrate your knowledge and understanding of how to perform string substitutions using the sub and gsub methods. Solve each of these exercises using IRB and the sub and gsub methods. Capture and turn in your results using screen prints.

2.2.1 :017 > s="Jack and Jill went up the hill to fetch a pail of water."

=> "Jack and Jill went up the hill to fetch a pail of water."

1. Enter the following statement into IRB, then use the sub method to replace the first vowel found in the string with an asterisk character:

"Jack and Jill went up the hill to fetch a pail of water."

2.2.1 :018 > s.sub(/[aeiou]/, '\*')

=> "J\*ck and Jill went up the hill to fetch a pail of water."

1. Enter the following statement into IRB. then use the gsub method to replace each vowel found in the string with an asterisk character.

"Jack and Jill went up the hill to fetch a pail of water."

2.2.1 :019 > s.gsub(/[aeiou]/, '\*')

=> "J\*ck \*nd J\*ll w\*nt \*p th\* h\*ll t\* f\*tch \* p\*\*l \*f w\*t\*r."

1. Enter the following statement into IRB, then use the gsub method to replace the string may be more than with the string can only be.

"In the end there may be more than one."

"In the end there may be more than one.".gsub('may be more than ', 'can only be')

=> "In the end there can only beone."

1. Enter the following statement into IRB, then use the gsub method to replace the word bob with the word Bob throughout the string.

"Big bob told little bob to take baby bob home."

2.2.1 :027 > "Big bob told little bob to take baby bob home.".gsub('bob','Bob')

=> "Big Bob told little Bob to take baby Bob home."

1. Enter the following statement into IRB, then use the gsub method to replace a comma for each blank space in the string.

"Bob Sue Sam Dick Jane Peter Paul John"

2.2.1 :028 > "Bob Sue Sam Dick Jane Peter Paul John".gsub(' ',',')

=> "Bob,Sue,Sam,Dick,Jane,Peter,Paul,John"

1. Enter the following statement into IRB, then use the sub method to remove the pound sign and all the characters that follow it.

"804-991-3434 #Client phone number"

2.2.1 :034 > "804-991-3434 #Client phone number".sub(/#.\*/,'')

=> "804-991-3434 "

1. Modify your solution to the above exercise to include a second regular expression that removes any blank space and hyphens from the string.

2.2.1 :058 > "804-991-3434 #Client phone number".sub(/#.\*/,'').gsub(/[-\s]/,'')

=> "8049913434"

8. Rails related questions

1. Name four ActiveRecord callbacks that you can bind methods to.

Ans: (1) before\_validation

(2) after\_validation

(3) before\_save

(4) before\_create

1. The Rails convention maps HTTP methods to certain controller methods, and those methods usually involve specific CRUD operations on models. Given the following CRUD database methods:

create, read, update, and delete

and the following HTTP methods:

GET, PUT, POST, DELETE

and the following controller actions: index, new, create, edit, update, destroy

Complete the following table.

|  |  |  |
| --- | --- | --- |
| HTTP method | controller action | CRUD operation |
| GET | index | READ |
| GET | new | CREATE |
| POST | create | CREATE |
| GET | edit | READ |
| PATCH | update | UPDATE |
| DELETE | destroy | DELETE |

1. Rails "simulates" PUT and DELETE requests. Why?
2. What is the difference between the two Rails environments 'production' and 'development' ?
3. Usually, Rails controllers incorporate plural nouns, such as ProtestsController and RevolutionsController. In what case should a controller have a singular name like GeocodingController?
4. What is a Rails "helper method" and when should they be defined and used by you, the developer?

9. Textbook: Ruby S., Thomas D., Heinemeier Hansson D. - Agile Web Development with Rails, 4th Edition. Skim through Chapter 5 and Chapter 6 and discuss the “Canonical Depot Application” explained and built in these chapters with respect to Rails Framework, Steps followed and potential pain points you may observe in this process. (no more than 1-2 page or flowchart) This book can be downloaded at: [https://www.dropbox.com/s/1hm9itzdaztsr1n/Agile Web Development with Rails.pdf?dl=0](https://www.dropbox.com/s/1hm9itzdaztsr1n/Agile%20Web%20Development%20with%20Rails.pdf?dl=0)

Depot application is an incremental model—where we start with some basic functionalities

And than we get the feedback and add more requirements.

The things which we have to keep in mind are:

1. Use case
2. Data flow
3. Page flow

Step for basic model view controller are:

Step1: rails new depot

Step2: cd depot

Step3: rails generate scaffold Product\ title:string description:text image\_url:string price:decimal

Step4: rake db:migrate

Step5: rails server

10. Project Based Questions:

A. Explain what REST means and how it relates to Rails. (About 1/4 to 1/2 page.) Include CRUD and HTTP in your discussion. If you use the textbook, the internet, or other external sources, give references.

REST stands for Representational State Transfer. (It is sometimes spelled "ReST".) It relies on a stateless, client-server, cacheable communications protocol -- and in virtually all cases, the HTTP protocol is used

REST is an architecture style for designing networked applications simple HTTP is used to make calls between machines. RESTful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations. Although, in Rails, REST and CRUD are bestest buddies, the two can work fine on their own. In fact, every time you have written a backend system that allows you to add, edit and delete items from the database, and a frontend that allows you to view those items, you have been working with CRUD.

B. Create Scaffold Project. Give commands, screenshots, code where applicable. Exam submitted without explanation, steps, commands and code won’t be graded.

1. Create a Rails scaffold-based project named MidtermPartC that keeps track of medal winners in the 2012 London Olympics. Use a scaffold with the model name Olympic medals with the fields athlete\_name, event, country, medal\_type (gold, silver, bronze), event\_date.
2. Change the dropdown menu for event\_date so that only 2012 is displayed.
3. Use validations to insure that athlete, and country are non empty.
4. Use a validation with the :inclusion argument to insure that medal\_type is "gold", "silver", or "bronze".
5. validates :medal\_type, :inclusion => { :in => ["gold", "silver", "bronze"] }
6. Use the layout page to display this image of olympic rings on all pages of the project.
   1. In the CSS file scaffolds.css.scss:
   2. Change the text color to white.
   3. Change the background color to black.
   4. Set the color of all hyperlinks to white.
   5. Set the font size to 110% of the default.
   6. Set the width of the olympic rings image in Part d to be 150 pixels.

Steps to complete it:

Step 1: create repository

Step2:bundle install

Step3: rails generate scaffold Olympicmedles athlete\_name:string event:text country:string medal\_type:string eventdate:date

Step4: rake db:migrate

Step5: in routes.rb file add root 'olympicmedles#index'

Step6: add below code in olympicmedle.rb in model

validates :athlete\_name, presence: true

validates :country, presence: true

validates :medal\_type, :inclusion => { :in => ["gold", "silver", "bronze"] }

Step7: In the CSS file scaffolds.css.scss: change body css

background-color: #333;

color: #fff;

font-size: 110%;

line-height: 150%;

step8: and in a css change{

color: #fff;

&:visited {

color:#fff

}

step9: add below code in application.html.erb

<img alt="Rails logo" src="/assets/olymipicring.jpeg" />"

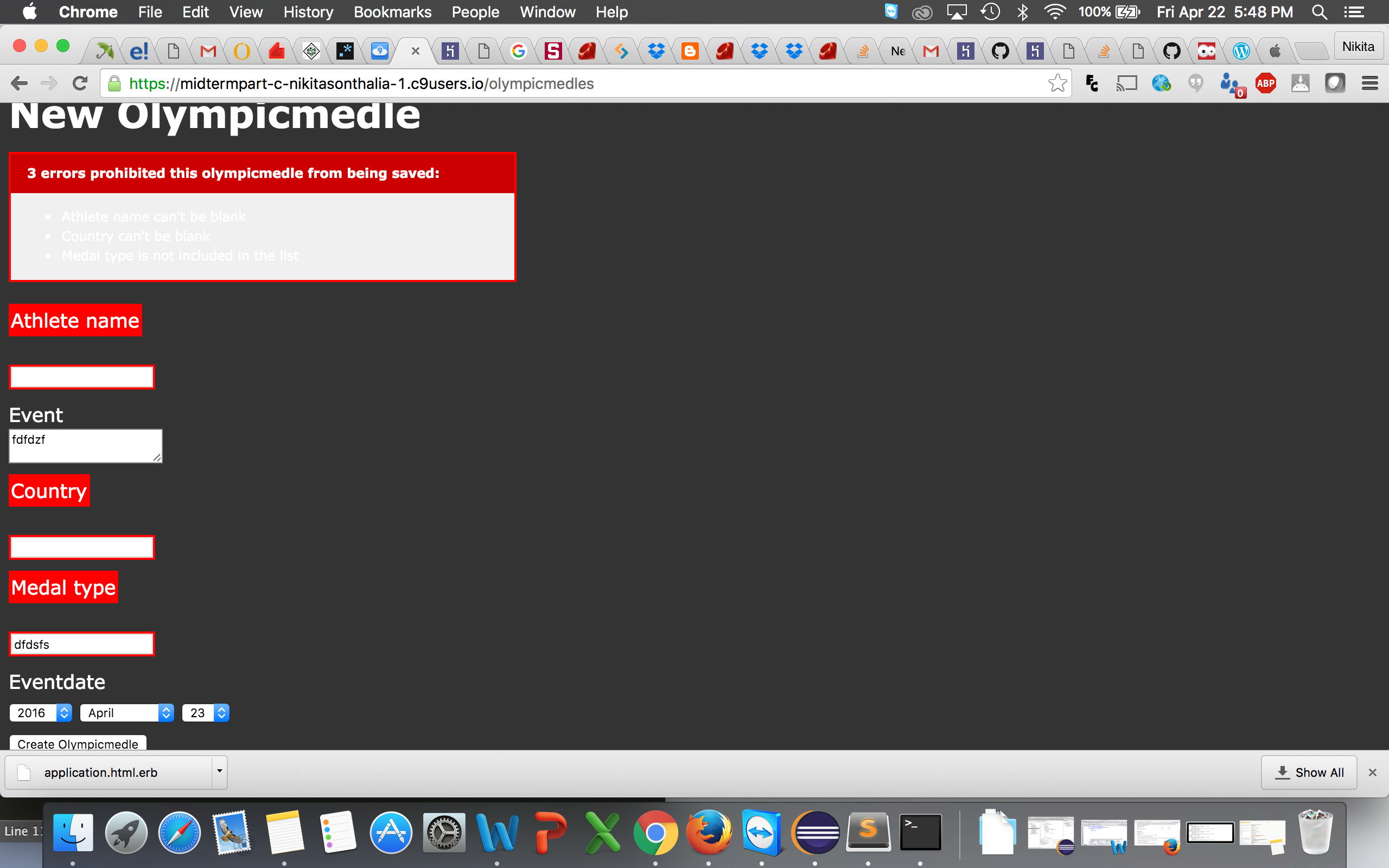
step10: add below code in application css

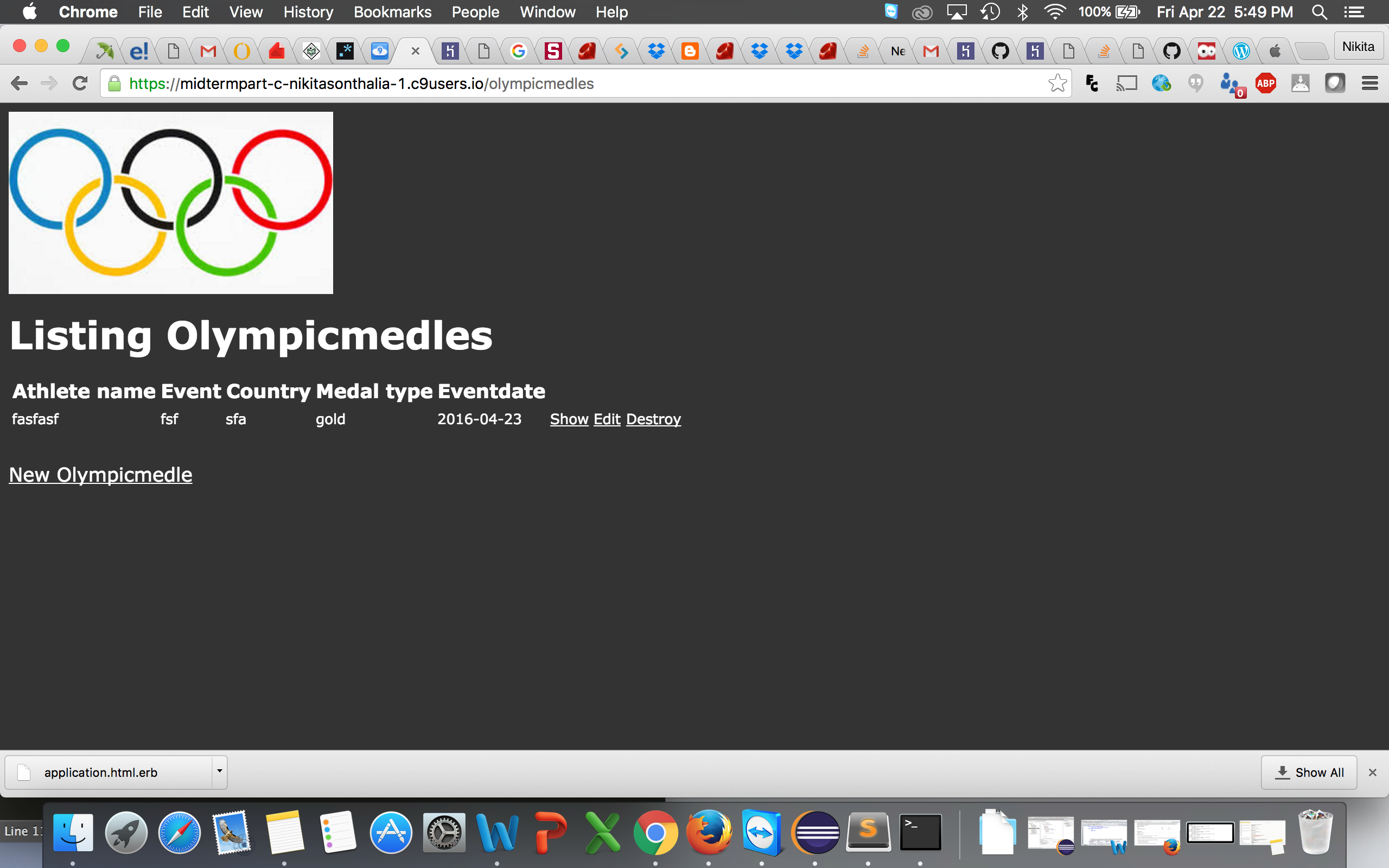
img {

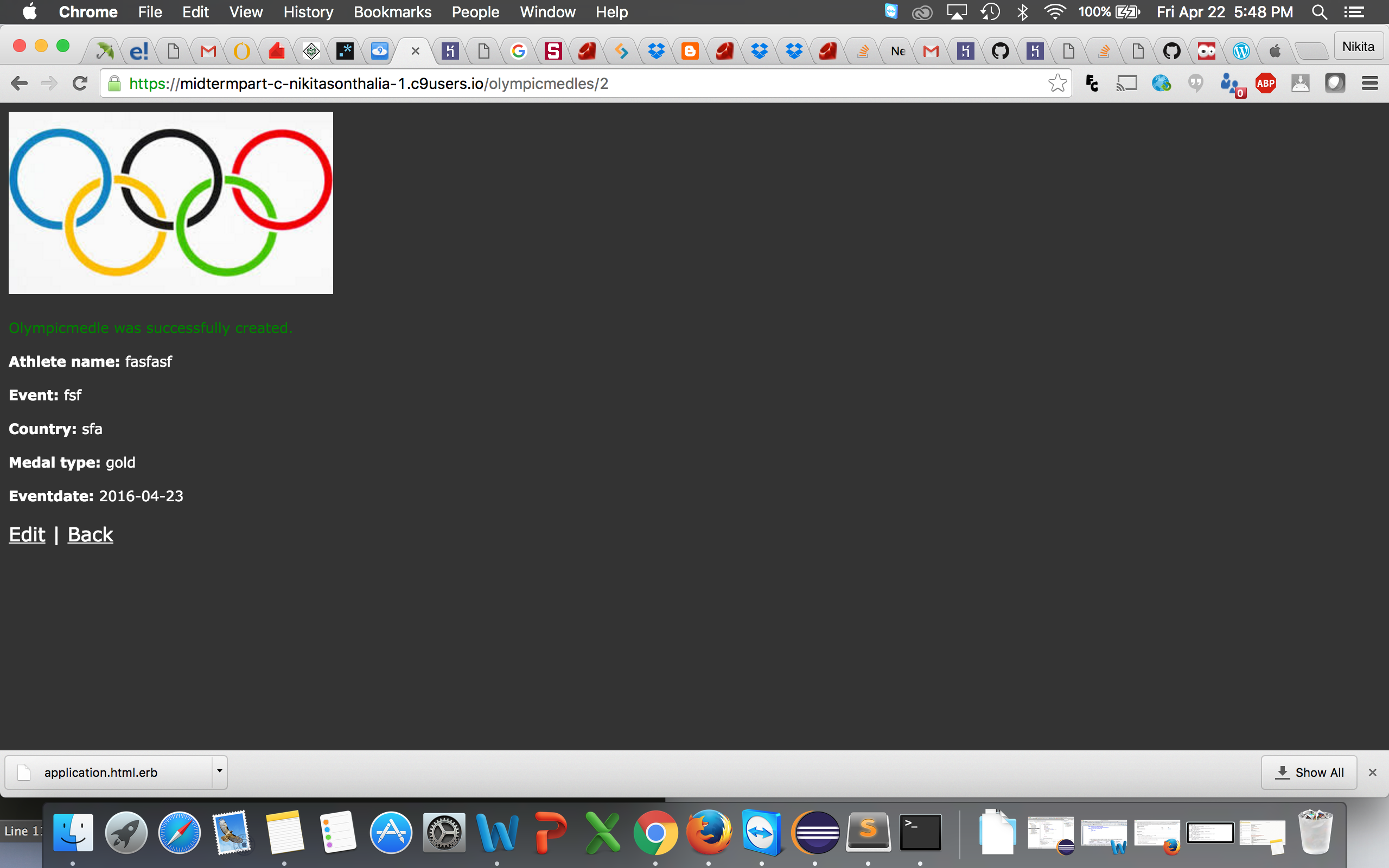
width: 150px;

}

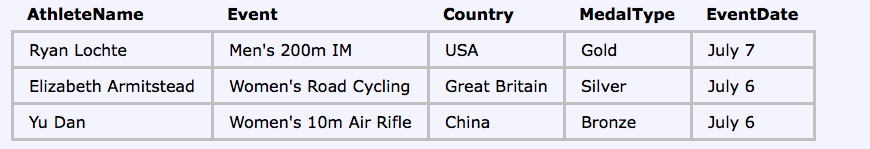
Screen shot:







1. Start the server and view your project in a browser. Use the new view of your project to enter the following olympic medal information in your project's database:



C. Find the Errors.

Fix the errors in the following source code. All your code should be XHTML complient. To check your answer, you can build a Rails project named MidtermPartC using the corrected source code. Generate a controller named ShowDateTime with a view named display. Copy these images to the assets/images folder before viewing the project.

jan.jpg feb.jpg mar.jpg apr.jpg may.jpg jun.jpg

jul.jpg aug.jpg sep.jpg oct.jpg nov.jpg dec.jpg

sun.png moon.jpg

Images can be downloaded from: <https://www.dropbox.com/s/v7fyrsefs84pu5i/images.zip?dl=0>

There are about 20 intentional mistakes. Complete these layout elements and push the static page RoR app to github print your repo link here. Deploy this to Heroku and print your Heroku link after fixing the above 20 mistakes and noting them here.

----- View code in app/views/show\_date\_time/display.html.erb -----------------------

<p>Date and Time: <%= @datetime %><br>

<%= image\_tag @timeimg, :class => "img" %>

<p>Month: <%= @monthname %><br />

<% image-tag @monthimg, :class = "img" %></p>

----- Layout code in app/views/layouts/application.html.erb ------------------------

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

<head>

<title><%= @title %>

<%= stylesheet\_tag "application" %>

<body>

<h2><%= @title %></h2>

yield

</body>

</html>

----- Controller code in app/views/show\_date\_time\_controller/display.html.erb ------

class ShowDateTimeController < ApplicationController

def display

@title = "Takehome Midterm -- Show Date and Time"

t = Time.now

@date\_time = t

if t.hour >= 6 && t.hour <= 18

@timeimg = "sun.png"

else

@timeimg = "moon.jpg"

end

if t.mon == 1

@monthname = "January"

@monthimg = "jan.jpg'

elsif t.mon = 2

@monthname = "February"

@monthimg = "feb.jpg"

elsif t.mon == 3

@monthname = "March"

@monthimg = "mar.jpg"

elseif t.mon == 4

@monthname = "April"

@monthimg = "apr.jpg"

elsif t.mon == 5

@monthname = "May"

@monthimg = "may.jpg"

elsif t.mon == 6

@monthname = "June"

@monthimg = "jun.jpg"

elsif t.mon == 7

@monthname = "July"

@monthimg = "jul.jpg"

elsif t.mon == 8

@monthname = "August"

@monthimg = "aug.jpg"

elsif t.mon == 9

@monthname = "September"

@monthimg = "sep.jpg"

elsif t.mon == 10

@monthname = "October"

@monthimg = "oct.jpg"

elsif t.mon == 11

@monthname = "November"

@monthimg = "nov.jpg"

elsif t.mon == 12

@monthname = "December"

@monthimg = "dec.jpg"

end

end

----- Stylesheet code in assets/stylesheets/application.html.erb -------------------

/\*

\*= require\_self

\*= require\_tree .

\*/

body { font-family; Helvetica:

font-size: 110%;

text-color: navy;

background\_color: lightblue;

.img { width: 200px; }