Day two

June, Tuesday 23rd

Recap

Recap

- Sinatra routes & redirects
- Sinatra views & ERB
- Exercises

Day two, Tuesday 23rd

- Sinatra sessions & Active Record
- Open class

Sinatra sessions


Pioneer

Sinatra sessions (I)

- HTTP is stateless, so there's no context between requests out of the box

- We have session cookies in order to do that

- Something saved in the web client, only for that specific domain

Sinatra sessions (2)

- Just activate them through enable :sessions
- A sessions hash will be available inside each route definition

Sessions in action

```ruby
require 'sinatra'
require 'sinatra/reloader'

enable :sessions

get '/' do
  session[:message] = 'Some message for later'
  redirect to('/show_message')
end

get '/show_message' do
  session[:message] # We have the message here
end
```

LIVE CODING FROM BCN!!!!111

Sessions' session!

LET'S GET READY TO RUMBLE

Exercise SL6

IRON HACK

A quick intro on DBs

Active Record

The Active Record pattern

- A best practice in software engineering
- There are several implementations, and of course in Ruby there is one of them
- It abstracts us from having to deal with databases (YAY!)

And more Active Record

- It basically puts an interface between an object and its database representation

- It maps a class to a table, and its relations to foreign keys

- Included by default in ActiveRecord

Useful methods (I)

- Class methods
  - find(*id*), returns the object for that id
  - find_by_*attr*(*value*), returns the first object matching the value for the attr
  - all, returns all the objects for that class
  - where(*conditions*), returns all the objects matching the conditions
  - delete_all, removes all the records from the DB

Useful methods (2)

- Instance methods
  - save, updates the database representation
  - valid?, which checks if it plays by our rules
  - destroy, removes the record from the DB

Setting it up

```
ActiveRecord::Base.establish_connection(
  adapter: 'sqlite3',
  database: 'activerecord.sqlite'
)
's | DDL
class Student < ActiveRecord::Base
end
```

Validations (I)

- The ability, through a DSL, to specify how our instances of a certain class have to be

- Only the valid classes will be stored into the database

- You can use the valid? instance method!

Validations (2)

- You can use a lot of the default validators that ActiveRecord provides

- Of course, you can also define your own validations

- More info:

guides.rubyonrails.org/active_record_validations.html

A validated student

```ruby
class Student < ActiveRecord::Base
  # we have name, surnames, birthday, website, number_of_dogs
  # and first_programming_experience

  AGE_MINIMUM = 16

  validates_presence_of :name, :surnames
  validates_format_of :website, with: /^http:/
  validates_numericality_of :number_of_dogs, greater_than: 0
  validate :proper_age

  private

  def proper_age
    unless birthday < AGE_MINIMUM.years.ago
      errors.add(:birthday, 'is too young')
    end
  end
end
```

LET'S GET READY TO RUMBLE

Exercise SL7

IRON HACK

sqlite

- One of the most simple DBMS (database management systems) that exist

- Perfect when you just want a database, nothing more

- Manage it with SQLite Browser (Mac OS X) or SQLite database browser (Ubuntu)

Models are not islands

- In a relational model, as you might expect, models have relations

- We can navigate through these relations

- *e.g. the Ironhack edition of a student*

Just another column

- The most common relation is one-to-many

- It is implemented using a column in the *many* part, which points to the *one* part…

- …and some magic from Active Record

An example

```ruby
class Student < ActiveRecord::Base
  # ... some unrelated stuff
  belongs_to :ironhack_edition
end

class IronhackEdition < ActiveRecord::Base
  has_many :students
end
```

Other kinds of relations

- One-to-one: a column in one of the classes
  - *e.g. two people within a couple*
- Many-to-many: we create another class which has two columns pointing outwards
  - *e.g. the uses of a tag in a blog*

And way, way, much, everything-you-can-imagine more.

Take a look at

github.com/rails/rails/tree/master/activerecord

to see what more things you can do!

LIVE CODING FROM BCN!!!! I I I

Let's have some relations! (in an Active Record way)

Open class

Exercises OC1 & OC2