software studio

code review: project 1

Daniel Jackson

controllers/trackedsites_controller.rb

no spec

mention of route: DRY!

use model validation

avoid cute comments

better: TODO

create, find_or_create

model code in controller?

famous last words...

```
# POST /trackedsites/visit
def visit
  # Check to make sure we have all the parameters we need.
  # Did not make an isInteger function because I'd only need it once.
  unless params[:url_string] && params[:path_string] && params[:total_time] && \
      !!(params[:total_time] =~ /^[-+]?[1-9]([0-9]*)?$/) && Integer(params[:total_time]) > 0

    # Preconditions not met, Time to abandon ship.
    render :text=> "Bad request", :content_type=>"text/plain",:status => 400 # Bad Request
  else
    # I used 'url_string' just in case 'url' is a reserved word
    @trackedsite = Trackedsite.find_by_url(params[:url_string])
    unless @trackedsite
      # Not a registered site, so let's register it!
      @trackedsite = Trackedsite.new(:url => params[:url_string])
      @trackedsite.save
    end

    # I used 'path_string' just in case 'path' is a reserved word
    @path = @trackedsite.paths.find_by_location(params[:path_string])
    unless @path
      # Path has not been tracked before, create this new path to start tracking.
      @path = @trackedsite.paths.create(:location => params[:path_string])
    end

    ## Register the visit
    @path.hit(Integer(params[:total_time])/1000.0) # Divide by 1000 to convert from milliseconds
    render :text => "OK", :status => 200 # Only using this once so not making this a method.
  end
end
```

controllers/sites_controller.rb

```
# Method that receives the HTTP request from the JavaScript from the user's page
# and processes this request
# Assume that site already exists (i.e. the user has to register the site first)
# Update the appropriate page (or create a new one), under the site
# Log visit of this page.
def updateVisit
 set_cors_headers
 #I still print out xhr.response. So I send "received".
 render :text => "received"
 @site = Site.find_by_root_url(params[:hostname])
 @page = @site.pages.find_or_create_by_url(params[:pathname])

 if not @page.visit.blank?
   # using methods defined in Visit model to update num_visits and duration
   @page.visit.update_duration(params[:duration].to_f)
   @page.visit.visit_increment()
   @page.visit.save
 # If the page has not been tracked before (new page).
 # Create a new visit entity for the page.
 # Set num_visit = 1 because we assume that the user must have visited the page in order
 else
   @page.visit = Visit.new
   @page.visit.num_visits = 1
   @page.visit.duration = params[:duration].to_f.round(1)
   @page.visit.save
 end
end
```

why is page an instance variable?

model code in controller

rounding: belongs in view?

app/models/page.rb

```
class Page < ActiveRecord::Base
  attr_accessible :name, :site_name, :site_id
  belongs_to :site
  has_many :visits

  # Returns the total number of visits to this site.
  def total_visits
      return visits.length
  end

  # Finds the total number of visits from unique IP addresses.
  def unique_visits
    uniques = 0
      seen_ips = Array.new
      # Count visits without previously seen IPs.
      for visit in visits
        if not seen_ips.include? visit.ip_address
            seen_ips.push(visit.ip_address)
            uniques += 1
        end
      end
      return uniques
  end
end
```

note visits is method!

performance?

partial encapsulation

controllers/visits_controller.rb

```
# GET /visits/new
# GET /visits/new.json
def new
  set_cors_headers
  @site = Site.find(params[:site_id])
    @visit = @site.visits.create(params[:visit])
    # Save the visitor's IP address.
    @visit.ip_address = request.remote_ip
    @visit.save

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @visit }
  end
end
```

encapsulate constructor too
model class = abstract type

breaks REST naming convention

config/routes.rb

```ruby
X::Application.routes.draw do
resources :visits
resources :sites

match 'sites' => 'sites#index', :via => :get
match 'sites/:id/visit' => 'sites#visit', :via => :get, :as => 'visit_site'

match "/restricted/resource" => "home#resource_preflight", :constraints => { :method => "OPTIONS" }
match "/restricted/resource" => "home#resource"

end
```

GET request may fetch from cache!

controllers/pages_controller.rb

```
class PagesController < ApplicationController

 #It is for the router match 'sites/:site_id/pages/:id/visits/update' => 'pages#monitor_visit'
 #When the site has visited, the counter inside the db will increment by 1.
 #This will process the http request (from any site), take out the duration parameter, and store in the db
 def monitor_visit
  #The setting to accept all the no original return call
  headers['Access-Control-Allow-Origin'] = "*"
  site = Site.find(params[:site_id])
  @page = site.pages.find(params[:id])
  @page.increment_visit
  @duration= [params[:duration]][0]
  @page.record_duration(@duration)
  @page.save!
 end
end
```

bad method name

implementation oriented spec

route in spec: DRY!

model code in controller

controllers/sites_controller.rb

```
class SitesController < ApplicationController

 # POST /sites/visit
 def visit
  @site = Site.find_by_url(params[:url])

  # check to see if site exists
  if @site.nil?
   @site = Site.create(:url => params[:url])
  end     @site = Site.where(:url => params[:url]).first_or_create

  # site needs to be saved before visit can be added
  @site.save
  @site.visits.create(:visit_time => params[:visit_time])
 end
end
```

which controller should visit creation belong to?

controllers/visits_controller.rb

```
class VisitsController < ApplicationController

 # POST /visits
 # POST /visits.json
 def create
  @visit = Visit.new(params[:visit])

  respond_to do |format|
   if @visit.save
    format.html { redirect_to @visit, notice: 'Visit was successfully created.' }
    format.json { render json: @visit, status: :created, location: @visit }
   else
    format.html { render action: "new" }
    format.json { render json: @visit.errors, status: :unprocessable_entity }
   end
  end
 end
```

/controllers/sites_controller.rb

```ruby
class SitesController < ApplicationController
  # DELETE /sites/1
  # DELETE /sites/1.json
  def destroy
    @site = Site.find(params[:id])
    @site.destroy

    respond_to do |format|
      format.html { redirect_to sites_url }
      format.json { head :no_content }
    end
  end
end
```

dead code? maybe not!

models/site.rb

```
# Record a visit by:
#   Update site's averate duration
#   Create a new visit for site
#   FInd or create (if not already exist) the relevant page and record a visit
def record_visit(page,duration)
  self.ave_dur = self.compute_ave_dur(self.ave_dur,
    self.visits.count + self.page_visits, duration)
  if page != '/'
    @page = self.pages.find_or_create_by_name(page)
    @page.ave_dur = self.compute_ave_dur(@page.ave_dur, @page.visits.count, duration)
    @page.visits.create(:duration => duration)
    @page.save
  else
    self.visits.create(:duration => duration)
  end
  self.save
end
```

type of page?

/config/routes.rb

```ruby
X::Application.routes.draw do
    resources :sites

    match "/sites/:id/visits" => "visits#visit_preflight", :constraints => { :method => "OPTIONS" }
    match "/sites/:id/visits" => "visits#create"
end
```

/controllers/sites_controller.rb

```ruby
class SitesController < ApplicationController

    # List all Sites in order of their id number
    def index
        @sites = Site.find(:all, :order=> "id")
        render "index", :layout => false
    end

    # Create new Site object with the given base url
    # Redirect to the site page with the JS snippet and its stats
    def create
        @site = Site.create()
        @site.base_url = params[:base_url]
        @site.save
        #redirect_to :action => :snippet, :id => site.site_number.to_s
        redirect_to @site
    end

    # Show the statistics and the JS snippet for the particular site
    # Statistics include visit breakdown by page and avg visit duration
    def show
        @url = "http://calm-fortress-2141.herokuapp.com/sites/" + params[:id] + "/visits"
        @site = Site.find(params[:id])
        render "site_details", :layout => false
    end

end
```

better: Site.from_url(...)

controllers/visits_controller.rb

```
class VisitsController < ApplicationController

    # If Site exists, finds/creates appropriate Page under it
    # Registers a Visit for that Page with the correct duration
    # Duration is passed via the AJAX call
    def create
        set_cors_headers
        if Site.exists?(params[:id])
            @page = Page.where(:site_id => params[:id], :path => params[:path]).first_or_create()
            @page.register_visit(params[:duration])
        end
    end

    #Set headers so AJAX call works
    def set_cors_headers
      headers["Access-Control-Allow-Origin"] = "*"
      headers["Access-Control-Allow-Methods"] = "POST, GET, OPTIONS"
      headers["Access-Control-Allow-Headers"] = "Content-Type, Origin, Referer, User-Agent"
      headers["Access-Control-Max-Age"] = "3600"
    end

    # Respond to pre-flight request from AJAX call
    def visit_preflight
      set_cors_headers
      render :text => "", :content_type => "text/plain"
    end

end
```

nice: trimmed to essential actions

/models/page.rb

```ruby
# Page represents a single webpage under a site
# Has many visits to represent each visit to that page
# Path represents the relative path on the site

class Page < ActiveRecord::Base
  attr_accessible :path, :site_id
  belongs_to :site
  has_many :visits

  # Register a visit on this page
  def register_visit(duration)
      @visit = Visit.create()
      @visit.page_id = self.id
      @visit.duration = duration
      @visit.save
  end

  # Calculate avg visit duration of all visits on page
  def avg_visit_duration
      total_duration = 0
      self.visits.each do |visit|
            total_duration += (visit.duration/1000.0)
      end
      return (total_duration/self.visits.length).round(1)
  end

end
```

Visit method?

rounding is view function?

controllers/visits_controller.rb

```
class VisitsController < ApplicationController

  # find which page/site this visit belongs to
  def find_visitable
    params.each do |name, value|
      if name =~ /(.+)_id$/
        return $1.classify.constantize.find(value)
      end
    end
    nil
  end

  # GET sites/visits
  # GET sites/page/visits
  def index
    @visitable = find_visitable
    @visits = @visitable.visits
  end

end
```

comments!

aborting loop in middle

Rails meta magic

controllers/sites_controller.rb

```
  # Show the statistics and the JS snippet for the particular site
    # Statistics include visit breakdown by page and avg visit duration
    def show
        @url = "http://calm-fortress-2141.herokuapp.com/sites/" + params[:id] + "/visits"
        @site = Site.find(params[:id])
        render "site_details", :layout => false
    end
```

hardcoded URL?

views/sites/site_details.html.erb

```
<h3> Javascript Snippet </h3>
    Insert the following Javascript snippet into each of the pages <br />
    that you wish to track. The snippet contains your site's generated id number. <br />
    <pre>
    &lt;script type="text/javascript"&gt;
        var startTime = new Date().getTime();
        window.onbeforeunload = sendRequest;
        function sendRequest() {
            var xhr = new XMLHttpRequest();
            xhr.open('POST', '<%= @url%>', false);
            var path = location.pathname;
            var duration = new Date().getTime() - startTime;
            var params = "path=" + path + "&duration=" + duration;
            xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
            xhr.send(params);
        }
    &lt;/script&gt;
    </pre>
```

summary

separation of concerns

› put data semantics in model classes

› remember to encapsulate constructor too

organization

› put actions in the right controllers

conventions

› obey RESTful conventions unless good reason not to

scaffolding

› remove unused actions!

specs

› spec is about external behavior; DRY

MIT OpenCourseWare
http://ocw.mit.edu

6.170 Software Studio
Spring 2013

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.