

1^η Εργαστηριακή Άσκηση

Εφαρμογή ιστού για κοινοχρησία φωτογραφιών στο Ruby-on-Rails

Επέκταση της εφαρμογής treegram με λειτουργίες που επιτρέπουν σε έναν χρήστη να προσθέσει τίτλο σε μια φωτογραφία, να ακολουθεί άλλους χρήστες, και να δει τις φωτογραφίες που ανεβάστηκαν από τους ακολουθούμενους χρήστες. Η υλοποίηση γίνεται με χρήση της γλώσσας Ruby, των βοηθητικών αρχείων και μεθόδων του Rails, και της γλώσσας Haml

1. Προσθήκη τίτλου σε φωτογραφία όταν την ανεβάζουμε στον διακομιστή
2. Προθήκη μοντέλου Follow που επιτρέπει στους χρήστες να ακολουθήσουν άλλους.
3. Προσθήκη μοντέλου Comment που επιτρέπει στους χρήστες να προσθέσουν σχόλια σε φωτογραφίες
4. Προσθήκη λειτουργίας που επιτρέπει στον χρήστη που ανέβασε την φωτογραφία να διαγράψει μαζί με όλα τα σχόλια και ετικέτες της

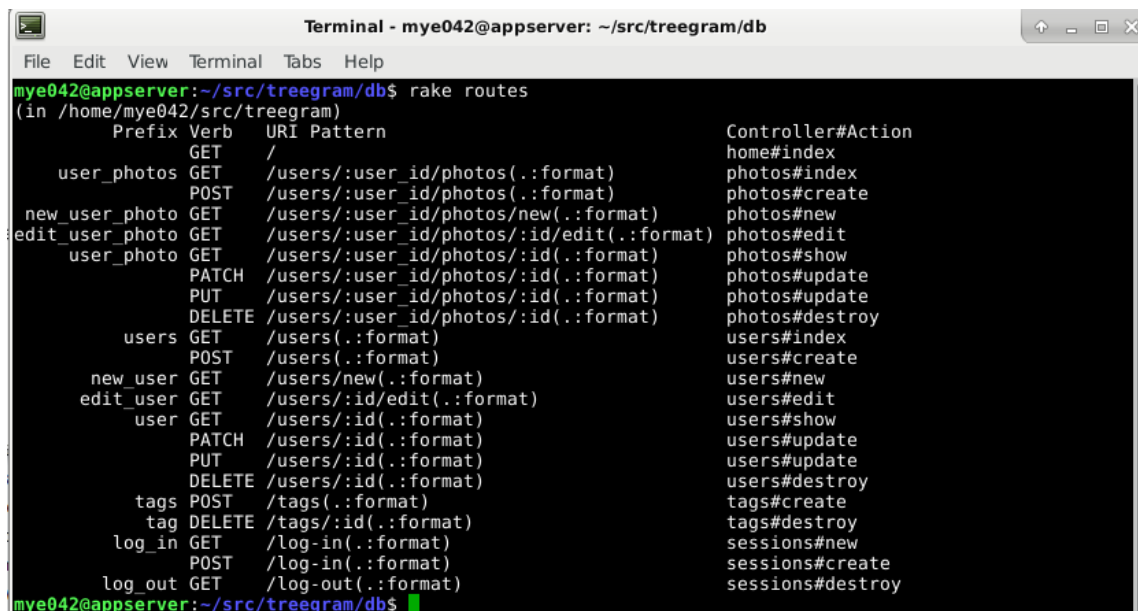
Αρχικά με τις εντολές

- rails generate migration AddTitleToPhoto title:string
- rake db:migrate

προσθέτουμε στο μοντέλο photo μια ιδιότητα title τύπου string και μετά το migration μας το περνάμε στην βάση. Αμέσως μετά για να βρούμε την φόρμα που είναι υπεύθυνη εκτελούμε την εντολή

- rake db:routes

και μπορούμε να δούμε τις διαδρομές της εφαρμογής . Από τον Controller μπορούμε να δούμε που στέλνει το Action στην περίπτωση μας το photos#new έχει φόρμα /views/photos/new.html.haml



```
Terminal - mye042@appserver: ~/src/treegram/db
File Edit View Terminal Tabs Help
mye042@appserver:~/src/treegram/db$ rake routes
(in /home/mye042/src/treegram)
  Prefix Verb   URI Pattern                                     Controller#Action
  user_photos GET    /users/:user_id/photos(.:format)             photos#index
  POST    /users/:user_id/photos(.:format)             photos#create
  new_user_photo GET    /users/:user_id/photos/new(.:format)          photos#new
  edit_user_photo GET    /users/:user_id/photos/:id/edit(.:format)     photos#edit
  user_photo GET    /users/:user_id/photos/:id(.:format)          photos#show
  PATCH   /users/:user_id/photos/:id(.:format)          photos#update
  PUT     /users/:user_id/photos/:id(.:format)          photos#update
  DELETE  /users/:user_id/photos/:id(.:format)          photos#destroy
  users GET    /users(.:format)                             users#index
  POST    /users(.:format)                             users#create
  new_user GET    /users/new(.:format)                         users#new
  edit_user GET    /users/:id/edit(.:format)                   users#edit
  user GET    /users/:id(.:format)                         users#show
  PATCH   /users/:id(.:format)                         users#update
  PUT     /users/:id(.:format)                         users#update
  DELETE  /users/:id(.:format)                         users#destroy
  tags POST    /tags(.:format)                             tags#create
  tag DELETE /tags/:id(.:format)                         tags#destroy
  log_in GET    /log-in(.:format)                           sessions#new
  POST    /log-in(.:format)                           sessions#create
  log_out GET    /log-out(.:format)                           sessions#destroy
mye042@appserver:~/src/treegram/db$
```

Προσθέτουμε στο αρχείο τις γραμμές 4-5 έτσι ώστε να συμπεριλάβουμε από τους χρήστες και τίτλο για την φωτογραφία .

```
1 | %h1 Add a Photo
2 | = form_for Photo.new(), :url => user_photos_path
3 |   , :html => {:multipart => true} do |form|
4 |     = form.file_field :image
5 |     = form.label :title
6 |     = form.text_field :title
7 |     = form.submit 'Upload'
```

Έπειτα βρίσκουμε την φόρμα που είναι υπεύθυνη για την προβολή τις σελίδας που εμφανίζονται οι φωτογραφίες των χρηστών για να συμπεριλάβουμε και τον τίτλο στην περίπτωση μας το Users#show έχει φόρμα το /views/users/show.html.html

```
21 | - @user.photos.each do |photo|
22 |   .well.col-sm-4
23 |   Title:
24 |   = photo.title
25 |   = image_tag photo.image.url(:medium)
```

Προσθέτουμε την γραμμή 23-24 ώστε σε κάθε φωτογραφία του χρήστη από πάνω να φαίνεται ο τίτλος.

Για την προσθήκη του μοντέλου follow και του αντίστοιχου πίνακα που καθορίζει τα αναγνωριστικά του ακολουθητή (follower) χρήστη και των ακολουθούμενων χρηστών (followee) εκτελούμε την εντολή

➤ rails generate model follow follower_id:integer followee_id:integer

Τώρα που υπάρχει το μοντέλο, ήρθε η ώρα να ορίσουμε τις σχέσεις για να διασφαλίσουμε ότι όλα συνδέονται. Στο μοντέλο follow, προσθέτουμε την σχέση belongs-to για έναν follower και έναν followee. Προσθέτοντας το στο τέλος class_name: 'User' των σχέσεων, συσχετίζονται ως υποκατηγορίες στην κλάση user. Βασικά επιτρέπει στα αναγνωριστικά user να εκχωρηθούν σε έναν follower και έναν followee.

```
1 | class Follow < ActiveRecord::Base
2 |   belongs_to :follower, class_name: 'User'
3 |   belongs_to :followee, class_name: 'User'
4 |
5 |   validates :follower_id, uniqueness: { scope: :followee_id }
6 |   validates :followee_id, uniqueness: { scope: :follower_id }
7 | end
```

Τα δύο validates παραπάνω είναι για να διασφαλιστεί ότι ένας χρήστης μπορεί να ακολουθήσει έναν άλλο χρήστη μόνο μία φορά και ένας χρήστης μπορεί να έχει μόνο ένα ακόλουθο από έναν άλλο χρήστη. Στο μοντέλο user, προσθέτουμε τις σχέσεις has_many για έναν follower και έναν followee

```
1 | class User < ActiveRecord::Base
2 |   has_many :photos
3 |   has_many :tags
4 |   has_many :comments
5 |   has_many :followed_users, foreign_key: :follower_id, class_name: 'Follow', dependent: :destroy
6 |   has_many :followees, through: :followed_users, dependent: :destroy
7 |   has_many :following_users, foreign_key: :followee_id, class_name: 'Follow', dependent: :destroy
8 |   has_many :followers, through: :following_users, dependent: :destroy
```

Η γραμμή 5 επιτρέπει σε έναν χρήστη να ακολουθεί πολλούς χρήστες. Μας επιτρέπει να έχουμε πρόσβαση στο foreign key: follower_id από την κλάση Follow. Στην γραμμή 6 το user id μπορεί στη συνέχεια να συσχετιστεί με την σχέση follow σαν follower. Το επόμενο κομμάτι γραμμές 7-8 είναι το ίδιο όπως παραπάνω, μόνο για την αντίστροφη σχέση. Μετά από τις αλλαγές εκτελούμε την εντολή

➤ rake db:migrate

Τώρα που έχουν ρυθμιστεί τα μοντέλα και οι σχέσεις, δημιουργούμε διαδρομές για ένα follow and unfollow.

```
post 'users/:id/follow', to: 'users#follow', as: "follow_user"
delete 'users/:id/unfollow', to: 'users#unfollow', as: "unfollow_user"
```

Η ενέργεια post χρησιμοποιείται επειδή είναι παρόμοια με την πραγματοποίηση μιας αλλαγής ή μιας ενημέρωσης σε έναν χρήστη, τα δεδομένα ενημερώνονται συσχετίζοντας μια παρακολούθηση με τον ακόλουθο και τον ακόλουθο. Το πρώτο μέρος περιλαμβάνει /users/:id/follow έτσι, όταν κάνουμε κλικ στο κουμπί follow, γνωρίζουμε με ποιον χρήστη σχετίζεται λόγω του αναγνωριστικού στη διεύθυνση url. Η ιδέα είναι ίδια και για τη διαδρομή unfollow. Το δεύτερο μέρος σημαίνει ότι όταν κάνουμε κλικ στο κουμπί, η σχετική διαδρομή θα δείχνει στην ενέργεια follow ή unfollow που ορίζεται στον user controller. Το τελευταίο μέρος επιτρέπει την ευκολότερη αναφορά της διαδρομής στον κώδικά μας χωρίς να χρειάζεται να γράψουμε την διεύθυνση url. Το επόμενο βήμα είναι η δημιουργία των μεθόδων εντός του user controller για follow και unfollow

```
49 def follow
50   @user = User.find(params[:id])
51   current_user.followees << @user
52   redirect_to :back
53 end
54
55 def unfollow
56   @user = User.find(params[:id])
57   current_user.followed_users.find_by(followee_id: @user.id).destroy
58   redirect_to :back
59 end
```

Βρίσκουμε τον χρήστη, προσθέτουμε αυτόν τον χρήστη στους ακόλουθους του τρέχοντος χρήστη και, στη συνέχεια, ανακατευθυνόμαστε πίσω στη σελίδα εμφάνισης του χρήστη. Για ένα unfollow, ομοίως, βρίσκουμε τον χρήστη από τους ακόλουθους του τρέχοντος χρήστη, και καταστρέφουμε την σχέση, στη συνέχεια, ανακατευθυνόμαστε πίσω στη σελίδα εμφάνισης του χρήστη.

Επίσης για ευκολία δημιουργήσαμε δυο βοηθητικές συναρτήσεις την search και την showafter για να περιηγούμε ποιο εύκολα στα προφίλ των χρηστών που ψάχνουμε στην βάση χρήστη με id ίδιο με αυτό που θα δίνει ο χρήστης.

```
35 def showafter(results)
36   @user = User.find(params[:search])
37 end
38
39 def search
40   if params[:search].present?
41     @parameter = params[:search]
42     @results = User.all.where("id LIKE :search", search: "%#{@parameter}%")
43     redirect_to showafter(@results)
44   end
45 end
```

Τώρα επιστρέφουμε στην φόρμα προβολής του χρήστη και προσθέτουμε κουμπιά για διαφορετικούς ελέγχους /view/user/show.html.html

```

1 | .row.top_row
2 | .col-sm-6.user_att
3 |   %h2= @user.email
4 |   - if @user.avatar_file_name
5 |     = image_tag @user.avatar.url(:thumb)
6 |     = form_tag search_user_path, method: :get, class: "navbar-form
7 |       navbar-right", role: "search" do
8 |         %p
9 |         = text_field_tag :search, params[:search], class: "form-control"
10 |        = submit_tag "Search By Id", name: nil, class: "btn btn-default"
11 |      %p
12 |      = @user.followers.count
13 |      Followers
14 |      - @user.followers.each do |x|
15 |        [Id:
16 |        = x.id
17 |        ,
18 |        = x.email
19 |        ]
20 |      %p
21 |      = @user.followees.count
22 |      Following
23 |      - @user.followees.each do |z|
24 |        [Id:
25 |        = z.id
26 |        ,
27 |        = z.email
28 |        ]
29 |      %p
30 |      - if @user == current_user
31 |        = link_to 'View Users', users_path, class: ['btn', 'btn-success', 'viewusers_btn']
32 |      - elsif current_user
33 |        = render 'users/follow_button', user: @user
34 |      .col-sm-6
35 |      = link_to 'Logout', log_out_path, class: ['btn', 'btn-danger', 'logout_btn']
36 |      .row
37 |      = link_to 'Add Photo', new_user_photo_path(@user), class: ['btn', 'btn-success', 'add_photo_btn']
38 |      = link_to 'Home', user_path(current_user), class: ['btn', 'btn-default', 'home_btn']

```

6-9: search button - search_user_path

```

2 | get '/users/search', to: 'users#search', as: "search_user"

```

11-27: followers και following χαρακτηριστικά για τον κάθε user

29-32: Ελέγχουμε αν είμαστε στο προφίλ του τρέχων χρήστη και αν ναι εμφανίζουμε κουμπί το οποίο παρέχει λεπτομέρειες για όλους τους υπάρχων χρήστες, αλλιώς σημαίνει ότι παρακολουθούμε την σελίδα κάποιου άλλου χρήστη και κάνουμε render στην φόρμα /views/users/_follow_button

```

_follow_button.html.html x routes.rb x user.rb x users_controller.rb x schen
- if following?(user)
  %h2= button_to "Unfollow", unfollow_user_path(id: @user.id), method: :delete
- else
  %h2= button_to "Follow", follow_user_path(id: @user.id), method: :post

```

Εδώ χρειαστήκαμε ένα module `user_helper` το οποίο το κάναμε `include` στον `user controller` για την αναγνώριση κάθε φορά που βρισκόμαστε σε προφίλ ξένου χρήστη τι κουμπί να εμφανίζει `follow` ή `unfollow` ανάλογα με τον αν κάποιος χρήστης ακολουθεί ήδη αυτόν που βλέπει ή όχι.

```
1 module UserHelper
2   include ActionController::RecordIdentifier
3
4   def following?(user)
5     current_user&.followees&.include?(user)
6   end
7
8   def dom_id_for_follower(follower)
9     dom_id(follower)
10  end
11 end
12
```

Μετά για την προσθήκη σχολίων στις φωτογραφίες δημιουργούμε το παρακάτω migration και προσθέτουμε στο `routes.rb` τα resources

- rails g model Comment content:text photo:reference user:references

```
1 class CreateComments < ActiveRecord::Migration
2   def change
3     create_table :comments do |t|
4       t.text :content, null: false
5       t.references :photo, index: true, foreign_key: true
6       t.references :user, index: true, foreign_key: true
7
8       t.timestamps null: false
9     end
10  end
11 end
```

```
7 resources :photos do
8   resources :comments
9 end
```

Για τον χειρισμό των σχολίων θα χρειαστούμε και έναν `comments controller` και να υλοποιήσουμε τις βασικές του λειτουργίες όπως φαίνεται παρακάτω

```
1 class CommentsController < ApplicationController
2   def index
3     @comments = Comment.all
4   end
5
6   def create
7     @photo = Photo.find(params[:photo_id])
8     @comment = @photo.comments.create(params[:comment].permit(:content, :photo_id, :user_id))
9     redirect_to :back
10  end
11
12  def destroy
13    @comment.destroy
14  end
15
16  def comment_params
17    params.require(:comment).permit(:content, :photo_id, :user_id)
18  end
19 end
20
```

Χρειαζόμαστε δύο φόρμες, μια για συλλογή των σχολίων και μια για παρουσίαση. Η `_comment.html.haml` είναι υπεύθυνη για την παρουσίαση και η `_form.html.haml` για την συλλογή

```
_comment.html.haml x
-| comments.each do |x|
  %p
  = x.content
  [by user with id:
  = x.user_id
  ]
  (
  = time_ago_in_words(x.created_at)
  ago
  )
```

```
_form.html.haml x
|= form_for([photo, photo.comments.build]) do |f|
  = f.hidden_field :user_id, value: current_user.id
  = f.text_field :content
  = f.submit
```

Αντίστοιχα όπως πριν πρέπει να κάνουμε render σε αυτά τα αρχεία από την φόρμα `/views/users/show.html.haml`.

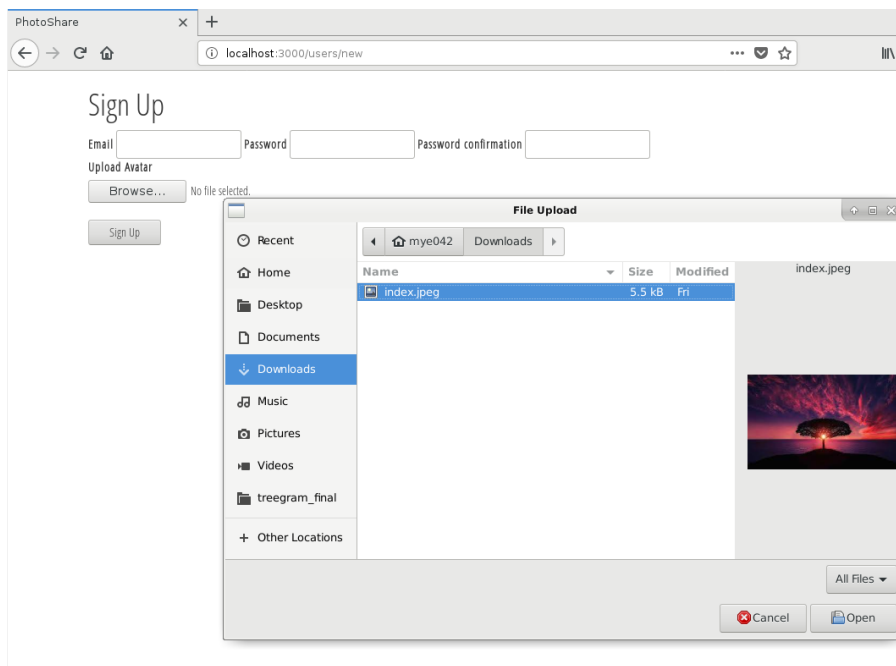
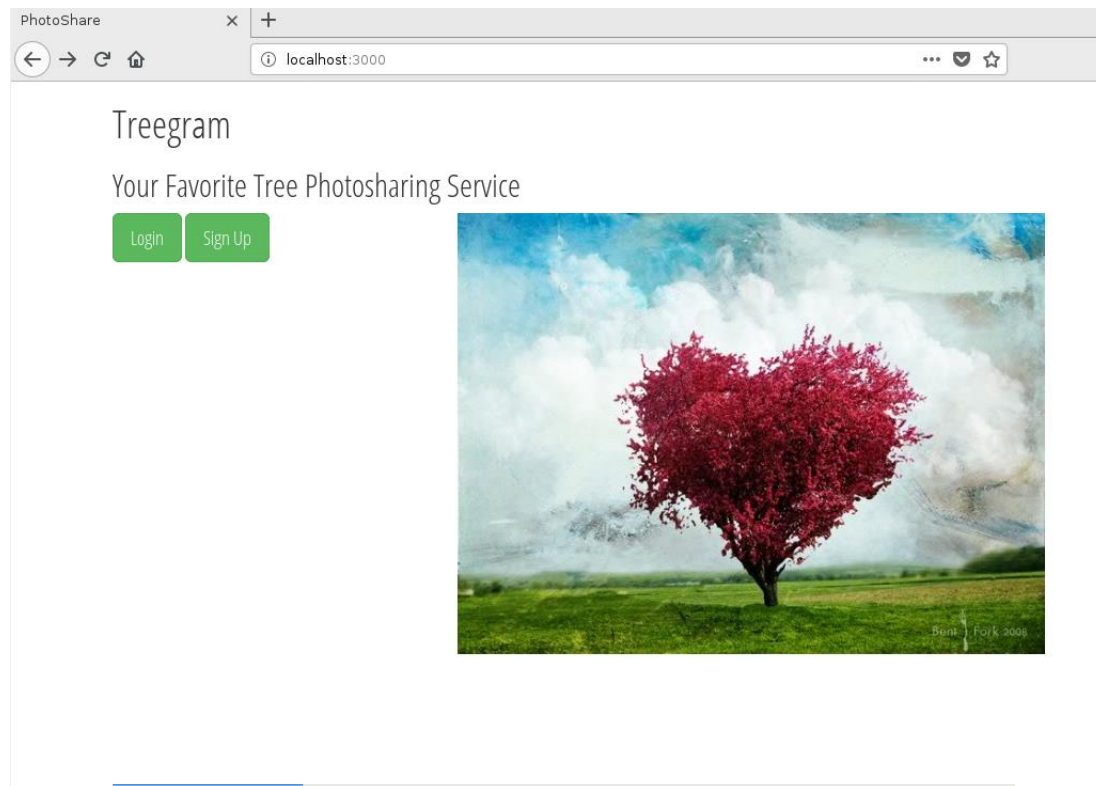
```
48 %h2= "MY PHOTOS"
49 .row
50 - @user.photos.each do |photo|
51   .well.col-sm-4
52     Title:
53     = photo.title
54     %p
55     Comments:
56     = photo.comments.count
57     %p
58     Creator User Id:
59     = @user.id
60     %p
61     - if @user == current_user
62       = button_to 'Delete', user_photo_path(photo.id, @user.id), :method => :
        delete, :confirm => 'Are you sure?'
63     .row
64     = image_tag photo.image.url(:medium)
65     = render 'comments/comment', comments: photo.comments
66     = render 'comments/form', photo: photo
67     = form_for @tag do |f|
68       = f.hidden_field :photo_id, value: photo.id
69       = f.collection_select :user_id, @users, :id, :email
70       = f.submit "Tag User"
71     - photo.tags.each do |tag|
72       = tag.user.email
```

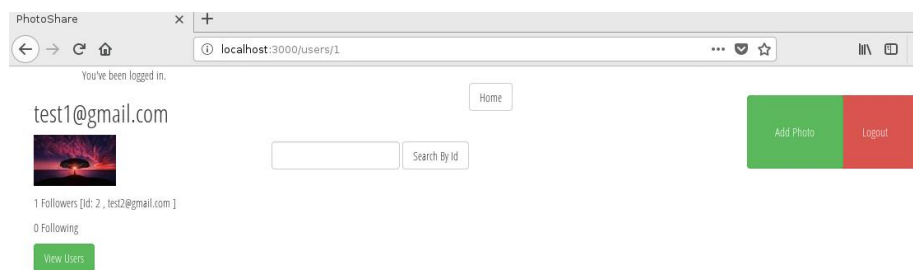
```
83 %h2= "MY FOLLOWEES PHOTOS"
84 .row
85 - @user.followees.each do |followee|
86   .well.col-sm-4
87     - followee.photos.each do |photo|
88       Title:
89       = photo.title
90       %p
91       Comments:
92       = photo.comments.count
93       %p
94       Creator User Id:
95       = followee.id
96       = image_tag photo.image.url(:medium)
97       = render 'comments/comment', comments: photo.comments
98       = render 'comments/form', photo: photo
99       = form_for @tag do |f|
100         = f.hidden_field :photo_id, value: photo.id
101         = f.collection_select :user_id, @users, :id, :email
102         = f.submit "Tag User"
103     - photo.tags.each do |tag|
104       = tag.user.email
```

61-62: delete button – method delete => Photos_Controller.rb

```
25 def destroy
26   @photo = Photo.find(params[:user_id])
27   @photo.destroy
28   redirect_to :back
29 end
```

ΠΡΟΒΟΛΗ Treegram





MY PHOTOS

