Rails routing, kontrollerek
Gyakorlat

Kovács Gábor

2021. november 9.

1. Modellek kiegészítése

Az előző gyakorlaton nem jutottunk a modellek végére, a megfigyeléstípusok enumja nem működött megfelelően. Ezért sztring típusú értékről egész típusú értékre módosítjuk az adatbázisban.

```
rails g migration ChangeSightingToInteger
```

Ez a migráció a következőképp néz ki:

```
class ChangeSightingToInteger < ActiveRecord::Migration[6.1]
  def change
    change_column :sightings, :sighting, :integer
  end
end
```

Az enum ezután helyesen a következőképp néz ki:

```
class Sighting < ApplicationRecord
  enum sighting: [:bee, :gardener, :crop_circle, :nessie, :
      stadium]
end
```

A másik óra végi hiba a megfigyelések és a megfigyelést tevő felhasználók közötti kapcsolat nem megfelelő működése volt, ami az idegen kulcs hiánya miatt lépett fel. Adjuk hozzá az idegen kulcsot

```
rails g migration AddUserIdToSightings user:references
```

Ez a kódrészlet automatikusan generálja a `change` metódus törzsét. Végrehajthatjuk a két migrációt.

```
rails db:migrate
```

Végül a csatolmány modell polimorfikus asszociációjában lemaradt az ezt jelző opció, ezt is adjuk hozzá.

```
class Attachment < ApplicationRecord
  belongs_to :content, polymorphic: true
end
```

2. Tördelés

A következő dolgunk a témák nézet végleges változatának kialakítása. A méhlegelők adatlapján (`beepastures/show.html.erb`) megjelenő megfigyelések számára nincs felső korlát, ez azonban hamar betöltheti a rendelkezésre álló függőleges teret, ezért bevezetjük ezek több lapra való tördelését. Egy oldalon legyen legfeljebb három megfigyelés. A lapozás megvalósításához felveszünk egy új függőséget a `Gemfile`-ba:

```
gem 'will_paginate'
```

majd telepítjük a hiányzó függőségeinket:

```
bundle install
```

A tördelés megvalósítása ezután egyszerűvé válik. A `beepastures/show.html.erb` nézeten felvesszük a lapozó linkeket a `@sightings` példányváltozóra.

```
<%= will_paginate @sightings %>
```

A `@sightings` példányváltozót eddig a `set_beepasture` privát metódusban állítottuk be, azonban erre nincsen szükség minden esetben, csakis a `show` akció esetén. Ezért felveszünk egy új szűrőt, amely meghívja ezen akció esetén a `paginate_beepasture` privát metódust, melynek törzsében a `page` paraméterhez tartozó töredéket vesszük elő az aktuális méhlegelőn történt megfigyelések közül.

```
class BeepasturesController < ApplicationController
  before_action :paginate_beepasture, only: :show
  private
    def paginate_beepasture
      @sightings = @beepasture.get_sightings_page(params[:page])
    end
end
```

A méhlegellő modell osztályt kiegészítjük a töredéket lekérdező példánymetódussal, melynek egy paramétere van, a töredék sorszáma. A megfigyelések töredékét úgy kérdezzük le, hogy egy töredék pontosan 3 elemet tartalmazzon.

```
class Beepasture < ApplicationRecord
```

```
  def get_sightings_page(page)
    self.sightings.paginate(page: page, per_page: 2)
  end
end
```

A `page` paraméter például `?page=2` formában jelenik meg a böngésző címsorában. Ezt Railsben nem szeretjük, ezért egy új útvonalat definiálunk a paraméter számára, ami ugyanúgy az `index` akcióra mutat, de tartalmazza a paramétert. Mivel ez egy konkrét téma példányra vonatkozik, az `on` opció értéke `member` lesz (egyébként `collection`-t használnánk).

```
  resources :beepastures do
    get 'page/:page', on: :member, as: 'page', to: 'beepastures#
        show'
  end
```

3. Fájlok fel- és letöltése

A méhlegelőkhöz, illetve az azokön történt megfigyelésekhez polimorfizmussal csatolmányokat rendelhetünk, amelyet fájlok feltöltésével valósítunk meg. Léteznek erre kész API-k, mint a PaperClip, a CarrierWave vagy az ActiveStorage, most azonban fapados megoldást adunk rá.

A csatolmány modellt az előző alkalommal már létrehoztuk. Egy csatolmányról megjegyezzük a MIME-típusát, azt a megoldást, amelyhez kapcsolódik, a feltöltött fájl elérési útvonalát a fájlrendszeren, az eredeti nevét és méretét. A csatolmányokat a fájlrendszeren a `public` könyvtárban létrehozott `data` alkönyvtárban tároljuk, mert bináris adatot az adatbázis sem tud a fájlrendszernél hatékonyabban tárolni.

A csatolmány fel- és letöltésének eseménykezelőjét egy új, önálló kontrollerrel valósítjuk meg, amelyben két akciót értint a feltöltés (`upload`) és a letöltés (`download`).

```
kovacsg@debian:~/gyakorlat/app/controllers> rails g controller
    attachments upload download
Running via Spring preloader in process 30599
      create  app/controllers/attachments_controller.rb
       route  get 'attachments/upload'
get 'attachments/download'
      invoke  erb
      create    app/views/attachments
      create    app/views/attachments/upload.html.erb
      create    app/views/attachments/download.html.erb
      invoke  test_unit
      create    test/controllers/attachments_controller_test.rb
```

```
      invoke  helper
      create    app/helpers/attachments_helper.rb
      invoke    test_unit
      invoke  assets
      invoke    scss
      create      app/assets/stylesheets/attachments.scss
```

Módosítjuk a most létrehozott fájlfeltöltő és -letöltő útvonalakat! A feltöltésnél az `id` paraméter annak a méhlegelőnek az azonosítóját jelenti, amelyhez új csatolmányt vagy új megfigyelést hozunk létre csatolmánnyal együtt. Hogy melyikről van épp szó, azt a `type` paraméter határozza meg. Letöltéskor az `id` paraméter a csatolmány azonosítóját jelenti.

```
Rails.application.routes.draw do
  post 'attachments/:id/:type/upload', to: 'attachments#upload',
      as: 'upload'
  get 'attachments/:id/download', to: 'attachments#download', as:
      'download'
end
```

A feltöltés helye a felület méhlegelőkhöz való csatomány feltöltése esetén a `edit.html.erb`, amelyet kiegészítünk egy formmal. A formot bináris adatok átküldésére is alkalmassá kell tennünk. Ehhez fel kell vennünk a `multipart` opciót a paraméterek közé, a méhlegelő azonosítója a modellből elérhető. A felhasználó állíthatóságára nincs szükségünk.

```
<%= form_tag upload_path(@beepasture.id, 'Beepasture'), multipart
    : true, method: :post do %>
  <%= label_tag 'file', 'File' %>
  <%= file_field_tag 'file' %>
  <%= submit_tag 'Upload' %>
<% end %>
```

Méhlegelőkhöz tartozó megfigyelések létrehozása a `edit.html.erb` képernyőn történik. Itt a formunk a fájlmező mellett tartalmaz egy legördülő menüt, amellyel a megfigyelés típusa választható ki.

```
<%= form_tag upload_path(@beepasture.id, 'Sighting'), multipart:
    true, method: :post do %>
  <%= label_tag 'sighting', 'Sighting' %>
  <%= select_tag 'sighting', options_for_select([['bee', 0], ['
      gardener', 1], ['crop_circle', 2], ['nessie', 3], ['
      stadium', 4]]) %>
  <%= label_tag 'file', 'File' %>
  <%= file_field_tag 'file' %>
  <%= submit_tag 'Upload' %>
<% end %>
```

Hozzuk létre a feltöltés eseménykezelőjét a csatolmányok kontrollerben. A csatolmány objektum létrehozásához a méhlegelőre, a fájlfeltöltést végző felhasználóra, aki nem más, mint az aktuálisan bejelentkezett felhasználó, magára a csatolt állományra, valamint csatolmány típurásra. A méhlegelőt előkeressük a kérés `id` paramétere alapján. A feltöltés befejeztével visszamegyünk az előző oldalra, ha volt feltöltött fájl, ellenkező esetben nem csinálunk semmit.

```
class AttachmentsController < ApplicationController
  def upload
    id = params[:id]
    type = params[:type]

    unless params[:file].nil?
      beepasture = Beepasture.find id
      Attachment.save_file beepasture, @user, type, params[:file
          ], params[:sighting]
      redirect_back fallback_location: hello_path
    end
  end

end
```

A csatolmány elmentésének logikus helye a csatolmány modell, ahol definiáljuk a fent hivatkozott osztálymetódust. Annak törzsében először definiáljuk a feltöltött állományokat tartalmazó könyvtárt, és létrehozzuk, ha nem létezne. Ezután létrehozuk egy új csatolmány objektumot, aminek beállíthatjuk a fájlnevét, a MIME típusát, és a típustól függően a méhlegelőt vagy a helyben létrehozott megfigyelés példányt, amelyhez tartozik. A fájlt még nem mentettük el, ezért sem a méretét, sem az elérési útját nem tudjuk még, ennek ellenére létrehozzuk a csatolmány objektumot. A következő lépés a fájl elmentése a kijelölt könyvtárba. Az elmentett fájl elérési útjával és méretével frissítjük a csatolmány objektumunkat.

```
class Attachment < ApplicationRecord
  belongs_to :content, polymorphic: true

  def Attachment.save_file(beepasture, user, type, file, sighting
      =nil)
    return if file.nil?
    dir = Rails.root.join('public','data')
    unless File.exists? dir
      Dir.mkdir dir
    end
    filename = file.original_filename
    a = Attachment.new
    a.name = filename
```

```
    a.mime = file.content_type
    unless sighting.nil?
      s = Sighting.create user: user, beepasture: beepasture,
          sighting: sighting.to_i
      Rails.logger.info s.user
      a.content = s
      s.attachments << a
#       a.content_id = s.id
#       a.content_type = 'Sighting'
    else
      a.content = beepasture
      beepasture.attachments << a
#       a.content_id = beepasture.id
#       a.content_type = 'Beepasture'
    a.save!

    path = File.join(dir, a.id.to_s)
    Rails.logger.info path
    File.open(path, 'wb') do |f| f.write(file.read) end
    a.update path: path, size: File.size(path)
  end
end
```

A csatolmányokat az `image` HTML tagek `src` attribútumában használjuk például a `show` nézetben. Az azonosítója az `index` nézetben az iteráció objektumából jön. Mivel egy megfigyeléshez több csatolmány is tartozhat, és itt most csak egynek van helye, ezért amennyiben van csatolmány, akkor csak az elsőt jelenítjük meg.

```
    <% @sightings.each do |sighting| %>
    <tr>
      <td><%= sighting.sighting %></td>
      <td><%= l sighting.created_at %></td>
      <td><% if sighting.attachments.any? %>
            <%= image_tag download_path(sighting.attachments[0]),
                width: 50, height: 50 %>
          <% end %>
      </td>
    </tr>
    <% end %>
```

A letöltés akciót a csatolmányok kontrollerbe helyezzük el. A letöltéshez előkeressük a csatolmányt az `id` paraméter alapján, és a `send_file` művelettel visszaküldjük a felhasználónak. Ha nincs ilyen azonosítóval csatolmány, akkor az erőforrás nem létezik hibát adunk vissza.

```
class AttachmentsController < ApplicationController
  def download
    a = Attachment.find params[:id]
```

```
    unless a.nil?
      send_file a.path, type: a.mime, disposition: :inline,
          filename: a.name
      return
    end
    head 404
  end
end
```