Rails routing, kontrollerek
Gyakorlat

Kovács Gábor

2018. április 23.

1. Útvonalak módosítása

A gyakorlat során az előző gyakorlatokon megkezdett feladat megoldását folytatjuk. Az előző gyakorlaton már átírtuk az útvonalakat a a `routes.rb`-ben, és menüben a felhasználóra vonatkozó útvonalak helyett már a létrehozott helpereket használjuk. Az útvonalakban így már felasználóazonosítók is szerepeljenek.

Az `ApplicationController` kontrollerben a felhasználót csak akkor kell előkeresnünk, ha be van jelentkezve, e feltételt utólag pótoljuk.

```
  def find_user
    @user = User.find session[:user] if session[:user]
  end
```

2. Tördelés

A következő dolgunk a témák nézet végleges változatának kialakítása. A témák listája (`topics/index.html.erb`) nézeten megjelenő témák számára nincs felső korlát, ez azonban hamar betölti a rendelkezésre álló függőleges teret, ezért bevezetjük ezek több lapra való tördelését. Egy oldalon legyen tíz topik. 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 `topics/index.html.erb` nézeten felvesszük a lapozó linkeket az `@topics` példányváltozóra.

```
<%= will_paginate @tasks %>
```

Ez `page` nevű paramétert generál minden kéréshez, amelyet átadhatunk a `paginate` függvénynek. Az összes téma helyett keressük elő a paraméternek megfelelő tördedéket. A keresőművelet logikus helye a feladatsor modell osztályban lesz.

```
class TopicsController < ApplicationController
  def index
    @topics = Topic.get_topic_page(params[:page])
  end
end
```

A téma modell osztályt kiegészítjük a töredéket lekérdező függvénnyel, melynek egy paramétere van, a töredék sorszáma. A témákat létrehozási idejük szerint csökkenő sorrendbe rendezzük, majd lekérdezzük a töredéket úgy, hogy egy töredék pontosan 10 feladatot tartalmazzon.

```
class Topic < ApplicationRecord
  def self.get_topic_page(page)
    Topic.all.order(created_at: :desc).paginate(page: page,
      per_page: 10)
  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 nem egy konkrét téma példányra vonatkozik, az `on` opció értéke `collection` lesz (egyébként `member`-t használnánk).

```
  resources :topics do
    get 'page/:page', on: :collection, to: 'topics#index'
  end
```

3. Fájlok fel- és letöltése

A témákhoz csatolhatunk fájlokat, amit csatolmány feltöltésével valósítunk meg. A csatolmány modellünket már korábban létrehoztuk: Egy csatolmányról megjegyezzük a MIME-típusát, azt a témát, amihez kapcsolódik, a feltöltőjét, a feltöltött fájl elérési útvonalát a fájlrendszeren, az eredeti nevét

és méretét. A megoldásokat 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 kontrollerrel valósítjuk meg, amelyben két akció lesz a feltöltés (`create`) és a letöltés (`download`). A nézeteket töröljük, ezekhez nem kell, hogy önálló nézet tartozzék.

```
kovacsg@debian:~/gyakorlat# rails g controller attachments create
    download
      create  app/controllers/attachments_controller.rb
       route  get 'attachments/create'
              get 'attachments/download'
      invoke  erb
      create    app/views/attachments
      create    app/views/attachments/create.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    coffee
      create      app/assets/javascripts/attachments.coffee
      invoke    scss
      create      app/assets/stylesheets/attachments.scss
```

Módosítjuk a fájlfeltöltő és -letöltő útvonalakat! A fájlfeltöltésnek két paramétere lesz, a téma azonosítója, a feltöltő felhasználó azonosítója, a letöltésnek pedig egy, a csatolmány azonosítója.

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

Módosítsuk a feladatot beadó űrlapunkat (`topics/_attachment.html.erb`), hogy az képes legyen a fájlfeltöltésre. Ehhez fel kell vennünk a `multipart` opciót a paraméterek közé, és az útvonal paraméternek át kell adunk az aktuális téma és felhasználó azonosítóját.

```
<fieldset>
  <legend>Attach a new file</legend>
  <%= form_for(@attachment, { url: upload_path(@topic.id, @user.
      id), html: { multipart: true }, method: :post}) do |f|%>
```

```
    <%= f.file_field 'upload' %><br/>
    <%= f.submit "Attach" %>
    <% end %>
</fieldset>
```

Hozzuk létre a feltöltés eseménykezelőjét a csatolmányok kontrollerben. Az objektum létrehozásához szükségünk van a témára és a felhasználóra, melyek idegen kulcsok a modellben, azokat a paraméterül átadott azonosító alapján előkereshetjük az adatbázisból. A feltöltött fájlra a a nézetben megadott `upload` paraméternévvel hivatkozhatunk. Ha e paraméter rendelkezik értékkel, akkor mentsük el. Önálló nézetünk nincs, ezért a következő oldalunk az előző oldal újratöltése lesz, ami egy téma adatlapja.

```
class AttachmentsController < ApplicationController
  def create
    unless params[:attachment][:upload].nil?
      Attachment.saveFile params[:attachment][:upload], params[:
          topic_id], params[:user_id]
    end
    redirect_back fallback_location: topic_path(params[:topic_id
        ])
  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 megoldást, amelyhez tartozik, ez utóbbit most hozzuk létre, hiszen most szüketett. 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
  def self.saveFile(upload, topic_id, user_id)
    dir = Rails.root.join('public', 'data')
    unless File.exists? dir
      Dir.mkdir dir
    end
    fname = upload.original_filename
    a = Attachment.new
    a.name = fname
    a.mime = upload.content_type
    a.topic_id = topic_id
    a.user_id = user_id
```

```
    a.save
    path = File.join(dir, a.id.to_s)
    File.open(path, 'wb') do |f| f.write(upload.read) end
    a.path = path
    a.size = File.size(path)
    a.save
  end
end
```

A téma adatainak megjelenítésekor a téma előkeresésén túl szükségünk van a téműhoz kapcsolódó összes csatolmányra és linkre. A kommentekkel későbbi gyakorlaton foglalkozunk.

```
class TopicsController < ApplicationController
  def show
    @comments = []

    @attachment = Attachment.new
    @attachments = @topic.attachments

    @links = @topic.links
  end
end
```

A csatolmányokat a `topics/_attachments.html.erb` töredékbe kell beágyaznunk. Ehhez szükség van a letöltés műveletre. A beágyazást HTML objektum által valósítjuk meg, amelyhez szükségünk van egy URL-re. Az URL a letöltés útvonalunk adja, amelynek egy paramétere van, a csatolmány azonosítója.

```
<ul>
<% for a in attachments do %>
  <li><%= link_to a.name, download_path(a.id) %></li>
<% end %>
</ul>
```

A letöltés akciót a csatolányok kontrollerbe helyezzük el. Az akció a paraméterük kapott azonosító alapján előkeresi a csatolmányt az adatbázisból, és annak útvonal attribútuma által kijelölt fájlt visszaküldi közben jelzi a böngészőnek, hogy próbálja beágyazva megjeleníteni a MIME típusa alapján.

```
class AttachmentsController < ApplicationController
  def download
    a = Attachment.find params[:id]
    send_file a.path, type: a.mime, disposition: :attachment
  end
end
```

A linkek megvalósítását a csatolmányok mintájára végezzük. E feladat sokkal egyszerűbb, hiszen csak egy string értéket kell tárolnunk, egyetlen akciónk van. Nézetre itt sincs szükségünk.

```
kovacsg@debian:~/gyakorlat/app/controllers# rails g controller
    links create
       create  app/controllers/links_controller.rb
        route  get 'links/create'
       invoke  erb
       create    app/views/links
       create    app/views/links/create.html.erb
       invoke  test_unit
       create    test/controllers/links_controller_test.rb
       invoke  helper
       create    app/helpers/links_helper.rb
       invoke    test_unit
       invoke  assets
       invoke    coffee
       create      app/assets/javascripts/links.coffee
       invoke    scss
       create      app/assets/stylesheets/links.scss
```

Módosítjuk a linklétrehozó útvonalat! Ennek is két paramétere lesz, a téma azonosítója, a feltöltő felhasználó azonosítója.

```
Rails.application.routes.draw do
  post 'links/create/:topic_id/:user_id', to: 'links#create', as:
       'add_link'
end
```

A linkek beágyazása is egyszerűbb, nincs szükségünk multipart formra, csak az útvonalat írjuk át. A linkek listázása HTML linkekkel történhet.

```
<ul>
<% for l in links do %>
  <li><%= link_to l.url, l.url %></li>
<% end %>
</ul>

<fieldset>
  <legend>Add a new link</legend>
  <%= form_tag add_link_path(@topic.id, @user.id), method: :post
      do %>
    <%= text_field_tag 'url' %><br/>
    <%= submit_tag "Link" %>
    <% end %>
</fieldset>
```

A linkek kontroller egyetlen akciója kezeli le ezt az eseményt egy új `Link` objektum létrehozásával, amelynek közvetlenül átadhatjuk a kérés paramé-

tereit miután fehérlistáztuk azokat a `link_params` függvényben. A feltöltés után újratöltjük az előző oldalt, ami gyakorlatilag egy topik adatlapja.

```
class LinksController < ApplicationController
 def create
   unless params[:url].nil?
     Link.create link_params
   end
   redirect_back fallback_location: topic_path(params[:topic_id
       ])
 end

 private
   def link_params
     params.permit([:url, :user_id, :topic_id])
   end
end
```