Rails routing, kontrollerek
Gyakorlat

Kovács Gábor

2018. november 6.

1. Útvonalak módosítása

A gyakorlat során az előző gyakorlatokon megkezdett feladat megoldását folytatjuk. A harmadik gyakorlaton kialakított menüben a felhasználóra vonatkozó útvonalak közösek az összes felhasználóra, azokat testre kell szabnunk az egyes felhasználók tekintetében, vagyis a felasználóazonosítóknak szerepelniük kell a linkekben, ezért módosítjuk a `routes.rb`-ben a megfelelő bejegyzéseket. Az átalakítást úgy tesszük meg, hogy az hasonlítson a `resource` függvénnyel generált útvonalakra. Az értintett linkek a felhasználói profil megtekintése, a profil szerkesztése nézetek, illetve az utóbbihoz tartozó eseménykezelők. Nevezzük el ezeket az útvonalakat a jobb karbantarthatóság végett.

```
Rails.application.routes.draw do
  get 'users', to: 'users#index', as: 'users'
  get 'users/new', to: 'users#new', as: 'register'
  post 'users/create', to: 'users#create', as: 'registration'
  get 'users/:id/edit', to: 'users#edit', as: 'profile'
  put 'users/:id', to: 'users#update', as: 'user'
  get 'users/:id', to: 'users#show'
  get 'users/forgotten'
  post 'users/recover'
  delete 'users/:id', to: 'users#destroy'
end
```

A felhasználói profil megtekintése nézetet (`users/show.html.erb`) korábban nem készítettük el, pótoljuk most egy egyszerű tartalommal:

```
<h1>User profile</h1>
<p>Name: <%= @user.name %></p>
<p>Email: <%= @user.email %></p>
<p>Neptun: <%= @user.neptun %></p>
```

Módosítsuk ennek megfelelően a bejelentkezett oktató és hallgató felhasználó menüjét. A menüben már az elnevezett útvonalhoz létrejött helper metódussal hivatkozunk az útvonalra az aktuális felhasználóval paraméterezve.

```
<%= link_to 'Profile', user_path(@user.id) %><br/>
<%= link_to 'Edit profile', profile_path(@user.id) %><br/>
<%= link_to 'Users', users_path %><br/>
<%= link_to 'New user', register_path %><br/>
<%= link_to 'New quiz', new_quiz_path %><br/>
<%= link_to 'Edit quizzes', quizzes_path %><br/>
<%= link_to 'Review', tasks_path %><br/>
<%= link_to 'Logout', logout_path, method: :delete %><br/>
```

```
<%= link_to 'Correct', tasks_path %><br/>
<%= link_to 'Solve', quizzes_path %><br/>
<%= link_to 'Edit_profile', '/users/edit' %><br/>
<%= link_to 'Logout', logout_path, method: :delete %><br/>
```

Egyúttal módosítjuk az üdvözlő üzenetet is, hogy az a felhasználót a nevén szólítsa. Az aktuálisan bejelentkezett felhasználót a kontrollerek ősosztályában állítjuk be, hiszen arra a megjelenítendő oldaltól, vagyis az elvégzendő műveletektől függetlenül szükségünk van.

```
class ApplicationController < ActionController::Base
  before_action :load_user

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

Ezt a példányváltozót hivatkozhatjuk meg a menüben.

```
<p>Hello <%= @user.name %>!</p>
```

Végül az útvonalakat tartalmazó `config/routes.rb` fájlban állítsuk át a webszerverünk gyökerét a hello nézetre.

```
root :to => "say#hello"
```

2. Modellek módosítása

Az előző gyakorlaton lemaradt a kapcsolat a felhasználók és a javítások között. Ezt most pótoljuk, a generált fájlon nem kell módosítanunk.

```
kovacsg@debian:~/gyakorlat# rails g migration AddUserToReviews
    user:references
```

```
      invoke  active_record
      create    db/migrate/20181106112024_add_user_to_reviews.rb
kovacsg@debian:~/gyakorlat/db# rails db:migrate
== 20181106112024 AddUserToReviews: migrating
   ==========================================
-- add_reference(:reviews, :user, {:foreign_key=>true})
   -> 0.0470s
== 20181106112024 AddUserToReviews: migrated (0.0471s)
   =================================
```

A két érintett modell osztályunkat is kiegészítjük ezzel a kapcsolattal:

```
class User < ApplicationRecord
  has_many :reviews
end
class Review < ApplicationRecord
  has_one :user
end
```

3. Tördelés

A következő dolgunk a feladatsorok nézet végleges változatának kialakítása. A feladatsorok listája (`quizzes/index.html.erb`) nézeten a feladatsor létrehozása, szerkesztése és törlése műveleteket csak az oktató típusú felhasználóknak engedélyezzük, a feladatsor kitöltése nézetet pedig csak a hallgató típusú felhasználók számára engedélyezzük. A kitöltés (`show`) és a szerkeszés `edit` nézetbe egyaránt beágyaztuk a feladatok listája nézetet, ezért majd ott is különbséget kell tennünk.

```
        <td><%= link_to 'Show', quiz unless is_lecturer? %></td>
        <td><%= link_to 'Edit', edit_quiz_path(quiz) if
            is_lecturer? %></td>
        <td><%= link_to 'Destroy', quiz, method: :delete, data: {
            confirm: 'Are you sure?' } if is_lecturer?%></td>
<%= link_to 'New Quiz', new_quiz_path if is_lecturer? %>
```

A hozzáférést korlátozó helpert, az `is_lecturer?`-t korábban elkészítettük, azonban a visszatérési értéke jelenleg `true`. Ez módosítanunk kell. Egy felhasználó akkor oktató, ha be van jelentkezve, és a típusa oktató. Az első feltétel ellenőrzésére már van helperünk.

```
module ApplicationHelper
  def is_lecturer?
    logged_in? && User.find(session[:user]).lecturer?
  end
end
```

A feladatsor szerkesztése nézeten egy formot látunk felül, alá pedig a feladatokat rendeztük egy táblázatba. Ezt most úgy módosítjuk, hogy a form csak az oktatók számára látható.

```
<%= render 'form', quiz: @quiz if is_lecturer? %>
<%= render 'tasks/index', tasks: @tasks %>
```

A feladatok `@tasks` azonosítójú példányváltozója még inicializálatlan, ezért a kontrollerben be kell állítanunk.

```
class QuizzesController < ApplicationController
  def show
    @tasks = @quiz.tasks
  end

  def edit
    @tasks = @quiz.tasks
  end
end
```

A feladatsor kitöltése nézeten ugyanazt a táblázatot látjuk, mint a feladatsor szerkesztése nézeten. Ez így nem jó, a feladatok adatait tartalmazó táblázat helyett magukat a feladatokat kellene itt látnunk.

Tehát a szerkesztendő fájlunk a feladatok beágyazó `tasks/_index.html.erb`. Először itt is különbséget teszük az oktatók és a hallgatók között. Az oktató a táblázatot látja, a hallgató pedig a feladat megoldását beadó formot.

```
<% if is_lecturer? %>
<table>
...
</table>
<% else %>
    <% @tasks.each do |task| %>
      <%= render 'tasks/solution', task: task %>
    <% end %>
<% end %>
```

A feladatot beadó űrlapunk (`tasks/_solution.html.erb`) nincs készen, hozzuk létre. Írjuk ki a feladat sorszámát, és szövegét egymás mellé vastag betűvel. Alatta pedig biztosítsuk egy beviteli mezőt a megoldás megadására.

```
<b><%= task.number %>. <%= task.text %></b><br/>
<%= form_tag(create_solution_path(task.quiz.id, task.id), method:
    :post ) do %>
  <%= text_field_tag :solution %>
  <%= submit_tag :send %>
<% end %>
```

Az űrlap eseménykezelője egy a `tasks` kontrollerben lévő akció lesz, amely létrehoz egy új `Solution` objektumot, amelyet a feladatsor egy feladatához fogunk rendelni. Az űrlapot csak bejelentkezett felhasználó töltheti ki, ezért a modell másik idegen kulcsának értéke is adott a `@user` példányváltozóban.

Mivel egy feladatsor korlátlan számú feladatból állhat, azok nem férnek el az oldan vertikálisan rendelkezésre álló területen. Ezen tördeléssel tudunk segíteni. Egy oldalon legyen egy feladat. 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 `tasks/_index.html.erb` nézeten felvesszük a lapozó linkeket az `@tasks` példányváltozóra.

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

Ez `page` nevű paramétert generál minden kéréshez, amelyet átadhatunk a `paginate` függvénynek. Az egy oldalon megjelenítendő események száma legyen 1. A feladatsor összes feladata helyett keressük elő a paraméternek megfelelő tördedéket. A keresőművelet logikus helye a feladatsor modell osztályban lesz.

```
class QuizzesController < ApplicationController
  def show
    @tasks = @quiz.get_tasks_page(params[:page])
  end
end
```

A feladatsor modell osztályt kiegészítjük a töredéket lekérdező függvénnyel, melynek egy paramétere van, a töredés sorszáma. A feladatsor feladatait a sorszámuk szerint növekvő sorrendbe rendezzük, majd lekérdezzük a töredéket úgy, hogy egy töredék pontosan 1 feladatot tartalmazzon.

```
class Quiz < ApplicationRecord
  def get_tasks_page(page)
    self.tasks.order(number: :asc).paginate(page: page, per_page:
        1)
  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 a `show` akcióra mutat, de tartalmazza a paramétert.

```
Rails.application.routes.draw do
  resources :quizzes do
    get 'page/:page', to: 'quizzes#show', on: :member # :
        collection
  end
end
```

4. Fájlok fel- és letöltése

A feladatok megoldásának beadása ne egy szövegdoboz kitöltésével, hanem a megoldás feltöltésével valósuljon meg. Ehhez fájlfeltöltést valósítunk meg, amelyhez szükségünk lesz egy új modellre, ami a feltöltött képekről tárol adatot. Egy csatolmányról megjegyezzük a MIME-típusát, azt a megoldást, amihez kapcsolódik, 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.

```
kovacsg@debian:~/gyakorlat# rails g model Attachment name:string
    mime:string size:integer path:string
      invoke  active_record
      create    db/migrate/20181106122450_create_attachments.rb
      create    app/models/attachment.rb
      invoke    test_unit
      create      test/models/attachment_test.rb
      create      test/fixtures/attachments.yml
kovacsg@debian:~/gyakorlat# rails db:migrate
== 20181106122450 CreateAttachments: migrating
   ===============================================
-- create_table(:attachments)
   -> 0.0230s
== 20181106122450 CreateAttachments: migrated (0.0233s)
   ======================================

kovacsg@debian:~/gyakorlat# rails g migration
    AddSolutionToAttachments solution:references
      invoke  active_record
      create    db/migrate/20181106122600
          _add_solution_to_attachments.rb
kovacsg@debian:~/gyakorlat# rails db:migrate
== 20181106122600 AddSolutionToAttachments: migrating
   ===============================================
-- add_reference(:attachments, :solution, {:foreign_key=>true})
   -> 0.0574s
```

```
== 20181106122600 AddSolutionToAttachments: migrated (0.0576s)
   ==========
```

Hajtsuk végre a migrációt, majd indítsuk újra a webszerverünket.

Egy megoldáshoz egy több fájl tartozhat, amelyek közül mindig az utolsó az aktuális. Ezért felvesszük ezt az egy-több relációt a két modellbe. A tulajdonos a megoldás modell.

```
class Solution < ApplicationRecord
  has_many :attachments
end
class Attachment < ApplicationRecord
  belongs_to :solution
end
```

Hozzuk először létre a fájlfeltöltő és -letöltő útvonalakat! A fájlfeltöltésnek két paramétere lesz, a feladatsor és a feladat azonosítói, a letöltésnek pedig egy, a megoldás azonosítója.

```
Rails.application.routes.draw do
  post 'upload/:quiz/:task', to: 'tasks#upload', as: '
      upload_solution'
  get 'download/:solution', to: 'tasks#download', as: '
      download_solution'
end
```

Módosítsuk a feladatot beadó űrlapunkat (`tasks/_solution.html.erb`), hogy az szövegbevitel helyett fájlfeltöltéssel működjék. Ehhez fel kell vennünk a `multipart` opciót a paraméterek közé

```
<b><%= task.number %>. <%= task.text %></b><br/>
<%= form_tag(upload_solution_path(task.quiz.id, task.id), method:
     :post, multipart: true ) do %>
  <%= text_field_tag :solution %>
  <%= submit_tag :send %>
<% end %>
```

Hozzuk létre a feltöltés eseménykezelőjét a feladatok kontrollerben. A megoldás objektum létrehozásához szükségünk van a feladatra, azt a paraméterül kapott azonosító alapján előkereshetjük az adatbázisból, a felhasználónk pedig rendelkezésre áll. A feltöltött fájlra a `solution` paraméterrel hivatkozhatunk. Ha az létezik, akkor mentsük el.

```
class TasksController < ApplicationController
  def upload
    @quiz = Quiz.find params[:quiz]
    @task = Task.find params[:task]
    file = params[:solution]
    unless params[:solution].nil?
```

```
      Attachment.saveFile params[:solution], @task, @user
    end
  end
end
```

A megoldáshoz rendelt 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(file, task, user)
    dir = Rails.root.join('public', 'data')
    unless File.exists? dir
      Dir.mkdir dir
    end
    fname = file.original_filename
    a = Attachment.new
    a.name = fname
    a.mime = file.content_type
    a.solution = Solution.create task: task, user: user
    a.path = ''
    a.save

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

A feladatra adott utolsó megoldást a `reviews/edit` nézetbe kell beágyaznunk. Ehhez szükség van a letöltés műveletre. Egy feladat javításakor rendelkezésre áll a feladat önmaga, és az arra született megoldások, amelyek mindegyike társítható egy hallgatóhoz. A feladat javítás nézeten a hallgatók nevét jelenítjük meg, amelyek közül kiválasztható egy kontkrét hallgató, és azáltal egy megoldás is. A beágyazást HTML objektum által valósítjuk meg, amelyhez szükségünk van egy URL-re. Az URL a letöltés útvonalunk adja,

amelynek egy paramétere van, a megoldás azonosítója, amely a feladat és a kiválasztott hallgató alapján kereshető.

A letöltés akciót a feladatok kontrollerbe helyezzük el. Az akció a paraméterük kapott megoldás azonosító alapján előkeresi a megoldást az adatbázisból, és a megoldáshoz tartozó csatolmány útvonal attribútuma által kijelölt fájlt visszaküldi közben jelzi a böngészőnek, hogy próbálja beágyazva megjeleníteni a MIME típusa alapján.

```
class TasksController < ApplicationController
  def download
    solution = Solution.find params[:solution]
    a = solution.attachment
    send_file a.path, type: a.mime, disposition: :inline
  end
end
```