Rails routing, kontrollerek
Gyakorlat

Kovács Gábor

2022. november 15.

1. Lapozás

A gyakorlat során az előző gyakorlatokon megkezdett feladat megoldását folytatjuk. A tantárgyak lisjáta képernyőn (`subjects/index.html.erb`) megjelenő tantárgyak számára nincs felső korlát, ez azonban hamar betöltheti a rendelkezésre álló függőleges teret, ezért bevezetjük ezek több lapra való tördelését. Egy oldalon legfeljebb egy tantárgy adatlapja fér el, ezért csak egyet jelenítünk meg. A lapozás megvalósításához felveszünk egy új függőséget[1] a `Gemfile`-ba:

```
gem 'will_paginate'
```

majd telepítjük a hiányzó függőségeinket:

```
bundle install
```

A tördelés megvalósítása ezután egyszerűvé válik. A `subjects/index.html.erb` nézeten felvesszük a lapozó linkeket a `@subjects` példányváltozóra.

```
<%= will_paginate @parties %>
```

A `@subjects` példányváltozót eddig az `index` metódusban állítottuk be, melynek törzsében a `page` paraméterhez tartozó töredéket vesszük elő az aktuális tantárgyak közül. Az effektív tördelést a tantárgy modell osztályában végezzük el egy osztálymetódusban.

```
class SubjectsController < ApplicationController
  def index
    #@subjects = Subject.all
    @subjects = Subject.get_subjects_page(params[:page])
  end
end
```

[1] `https://github.com/mislav/will_paginate`

Először konzolon nézzük meg, hogyan is történik a lapokra tördelés. Az alábbi kódrészlet 1. sorában az összes tárgyat lekérdezzük úgy, ahogy az most a kontrollerben van. A 2. sorban a tárgy neve szerint rendezzük őket. A 3. sorban e rendezett listából az első, egyelemű lapot vesszük elő, a 4. sorban pedig a második lapot.

```
irb(main):001:0> Subject.all
  Subject Load (0.9ms)  SELECT `subjects`.* FROM `subjects`
=>
[#<Subject:0x000055dc88fc2320
  id: 1,
  name: "Testnevelés",
  lecturer: "Futó Béla",
  pluto: "BMETT001",
  credit: 0,
  created_at: Tue, 11 Oct 2022 11:20:18.121945000 UTC +00:00,
  updated_at: Tue, 11 Oct 2022 11:20:18.121945000 UTC +00:00>,
 #<Subject:0x00007f171c47c540
  id: 2,
  name: "Fizika",
  lecturer: "Orosz László",
  pluto: "BMEFT001",
  credit: 5,
  created_at: Tue, 11 Oct 2022 11:21:29.181438000 UTC +00:00,
  updated_at: Tue, 11 Oct 2022 11:21:29.181438000 UTC +00:00>]
irb(main):002:0> Subject.all.order(name: :asc)
  Subject Load (3.1ms)  SELECT `subjects`.* FROM `subjects` ORDER
      BY `subjects`.`name` ASC
=>
[#<Subject:0x000055dc8865e388
  id: 2,
  name: "Fizika",
  lecturer: "Orosz László",
  pluto: "BMEFT001",
  credit: 5,
  created_at: Tue, 11 Oct 2022 11:21:29.181438000 UTC +00:00,
  updated_at: Tue, 11 Oct 2022 11:21:29.181438000 UTC +00:00>,
 #<Subject:0x000055dc8865e2c0
  id: 1,
  name: "Testnevelés",
  lecturer: "Futó Béla",
  pluto: "BMETT001",
  credit: 0,
  created_at: Tue, 11 Oct 2022 11:20:18.121945000 UTC +00:00,
  updated_at: Tue, 11 Oct 2022 11:20:18.121945000 UTC +00:00>]
irb(main):003:0> Subject.all.order(name: :asc).paginate(page: 1,
    per_page: 1)
  Subject Load (3.2ms)  SELECT `subjects`.* FROM `subjects` ORDER
      BY `subjects`.`name` ASC LIMIT 1 OFFSET 0
```

```
=>
[#<Subject:0x00007f171c4378c8
  id: 2,
  name: "Fizika",
  lecturer: "Orosz László",
  pluto: "BMEFT001",
  credit: 5,
  created_at: Tue, 11 Oct 2022 11:21:29.181438000 UTC +00:00,
  updated_at: Tue, 11 Oct 2022 11:21:29.181438000 UTC +00:00>]
irb(main):004:0> Subject.all.order(name: :asc).paginate(page: 2,
    per_page: 1)
  Subject Load (0.3ms)  SELECT `subjects`.* FROM `subjects` ORDER
      BY `subjects`.`name` ASC LIMIT 1 OFFSET 1
=>
[#<Subject:0x00007f171cd66610
  id: 1,
  name: "Testnevelés",
  lecturer: "Futó Béla",
  pluto: "BMETT001",
  credit: 0,
  created_at: Tue, 11 Oct 2022 11:20:18.121945000 UTC +00:00,
  updated_at: Tue, 11 Oct 2022 11:20:18.121945000 UTC +00:00>]
```

A tantárgy modell osztályt kiegészítjük a töredéket lekérdező osztálymetódussal, melynek egy paramétere van, a töredék sorszáma. A tantárgyak töredékét úgy kérdezzük le, hogy egy töredék pontosan 1 elemet tartalmazzon.

```
class Subject < ApplicationRecord
    def self.get_subjects_page(page)
        Subject.all.order(name: :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 az `index` akcióra mutat, de tartalmazza a `:page` paramétert. Mivel ez nem egy konkrét tantárgy példányra vonatkozik, az `on` opció értéke `collection` lesz, így az `:id` paraméter nem lesz része az útvonalnak (egyébként `member`-t használnánk).

```
Rails.application.routes.draw do
  resources :subjects do
    get 'page/:page', to: 'subjects#index', on: :collection
  end
end
```

2. Fájlok fel- és letöltése

Minden hallgatónak van egy avatarja, amely egy kép, és amelyet a profil oldalon fel, illetve le kell tudnunk tölteni. Léteznek erre kész API-k, mint a PaperClip, a CarrierWave vagy az ActiveStorage[2], most azonban fapados megoldást adunk rá.

A modellünket nevezzük `Attachment`-nek. Egy csatolmányról megjegyezzük a MIME-típusát, azt a felhasználót, amelyhez kapcsolódik, a feltöltött fájl elérési útvonalát a fájlrendszeren, az eredeti nevét és méretét. A csatolmány a fájlrendszeren a `public` könyvtárban létrehozott `avatars` alkönyvtárban tároljuk, mert bináris adatot az adatbázis sem tud a fájlrendszernél hatékonyabban tárolni.

```
kovacsg@debian:~/pluto/tmp/cache> rails g model Attachment
    filename:string path:string mime:string size:integer user:
    references
       invoke  active_record
       create    db/migrate/20221115114430_create_attachments.rb
       create    app/models/attachment.rb
       invoke    test_unit
       create      test/models/attachment_test.rb
       create      test/fixtures/attachments.yml
kovacsg@debian:~/pluto/tmp/cache> rails db:migrate
== 20221115114430 CreateAttachments: migrating
   ===============================================
-- create_table(:attachments)
   -> 0.0160s
== 20221115114430 CreateAttachments: migrated (0.0162s)
   ======================================
```

A csatolmány egy-egy kapcsolatban áll a felhasználóval, az `Attachment` modellben a kapcsolat automatikusan létrejött, a felhasználó modelljéhez azonban hozzá kell adnunk.

```
class User < ApplicationRecord
    has_one :attachment
end
```

A csatolány fel- és letöltésésének eseménykezelőjét a felhasználó kontrollerrel valósítjuk meg, amelyben két akciót értint a feltöltés (`upload`) és a letöltés (`avatar`). Ezen kívül legelőször a profil szerkesztése eseménykezetlőjét hozzáadjuk a kontrollerhez, amelyet a korábbi gyakorlatokon elmulasztottunk. Erre a `update` osztálymetódust használjuk, amelynek paramétere az a hash, amely a felhasználótől érkező form alapján áll elő, és megfelel a

[2] `https://edgeguides.rubyonrails.org/active_storage_overview.html`

`user_params` függvényben definiált fehérlistázás követelményeinek. Sikertelen módosítás esetén az előző oldalra megyünk vissza.

```
class UsersController < ApplicationController
  def update
    if User.update(user_params)
      redirect_to hello_path
    else
      flash[:notice] = 'Unsuccessful update'
      redirect_back fallback_location: edit_user_path(@user.id)
    end
  end
end
```

Módosítsuk a most felhasználók útvonalait fájlfeltöltő és -letöltő útvonalakat! Mind a fel-, mind a letöltésnél az `id` paraméter annak a felhasználónak az azonosítóját jelenti, akihez az új avatart rendelünk a csatolmány által.

```
Rails.application.routes.draw do
  post 'users/:id/upload', to: 'users#upload', as: 'upload'
  get 'users/:id/avatar', to: 'users#avatar', as: 'avatar'
end
```

A feltöltés helye a felhasználói profile képernyőn létrehozandó új form, amely egyetlen beviteli mezőből áll, egy fájlválasztóból, amelynek a neve legyen `file`. A formot bináris adatok átküldésére is alkalmassá kell tennünk. Ehhez fel kell vennünk a `multipart` opciót a paraméterek közé. A metódus HTTP POST, mert az előbb az útvonalak között azt adtuk meg.

```
    <fieldset>
        <legend>Set Avatar</legend>
        <%= form_tag upload_path(@user.id), multipart: true,
            method: :post do %>
            <%= label_tag 'file', 'Avatar' %>
            <%= file_field_tag 'file' %>
            <%= submit_tag 'Save' %>
        <% end %>
    </fieldset>
```

Hozzuk létre a feltöltés eseménykezelőjét a felhasználók kontrollerében. A csatolmányt akkor tudjuk hozzárendelni a felhasználóhoz, ha a felhasználó már létezik az adatbázisban, ezért a `user_params` függvényben a `params` hasht fehérlistázó sort nem kell módosítanunk. Ha van csatolmány a feltöltésben, vagyis a `file` kulcshoz tartozó paraméter létezik, akkor a feltöltött fájl objektumot elmentjük az `Attachment` modell osztály egy osztálymetódusával.

```
class UsersController < ApplicationController
```

```
  def upload
    unless params[:file].nil?
        Attachment.save_file(params[:file], @user)
    end
    redirect_back fallback_location: hello_path
  end
end
```

A Rails alkalmazáson belüli útvonalakra a fájlrendszeren a következőképp hivatkozhatunk:

```
irb(main):005:0> Rails.root
=> #<Pathname:/home/kovacsg/pluto>
irb(main):006:0> Rails.root.join('public', 'avatar')
=> #<Pathname:/home/kovacsg/pluto/public/avatar>
```

A csatolmány elmentésének logikus helye a csatolmány modell, mert több helyen használjuk, és így csak egy helyen kell karbantartanunk. Itt definiáljuk a fent hivatkozott osztálymetódust. Annak törzsében először definiáljuk a feltöltött csatolmányokat tartalmazó könyvtárt, és létrehozzuk, ha nem létezne. Ezután létrehozuk egy új csatolmány objektumot, amelynek beállíthatjuk a fájlnevét, a MIME típusát, és a felhasználót, akihez 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 :user

    def self.save_file(upload, user)
        return if upload.nil?
        dir = Rails.root.join("public", "avatars")
        unless File.exists? dir
            Dir.mkdir dir
        end
        fname = upload.original_filename
        a = Attachment.create filename: fname, mime: upload.
            content_type, user: user

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

A csatolmányokat az `img` HTML tagek `src` attribútumában használjuk például az `edit` nézetben, amely a felhasználói profilt rendeleli.

```
<img src="<%= avatar_path(@user.id) %>" height="80" width="80">
```

A letöltés akciót a felhasználók kontrollerbe helyezzük el. A csatolmány egy felhasználó `id`-val azonosított példányához tartozik, amelyet a paraméter alapján a `user` lokális változó beállítható. E felhasználtó objektumhoz asszociált logó objektumot előkeressük az adatbázisból, ha van ilyen, és a `send_file` művelettel visszaküldjük a felhasználónak. Ha nincs ilyen azonosítóval csatolmány, akkor az erőforrás nem létezik hibát adunk vissza.

```
class UsersController < ApplicationController
  def avatar
    user = User.find params[:id]
    unless user.nil?
      a = Attachment.where(user: user).take
      unless a.nil?
        send_file a.path, type: a.mime, disposition: :attachment
        return
      end
    end
    head 404
  end
end
```

Nézzük meg konzolon, hogy a webfelület való feltöltés sikeres volt-e.

```
kovacsg@debian:~/pluto/tmp/cache> rails c
Loading development environment (Rails 7.0.4)
irb(main):001:0> User.last.attachment
  User Load (0.3ms)  SELECT `users`.* FROM `users` ORDER BY `
      users`.`id` DESC LIMIT 1
  Attachment Load (2.4ms)  SELECT `attachments`.* FROM `
      attachments` WHERE `attachments`.`user_id` = 4 LIMIT 1
=>
#<Attachment:0x00007f89c44ce230
 id: 1,
 filename: "archivum_icon.png",
 path: "/home/kovacsg/pluto/public/avatars/1",
 mime: "image/png",
 size: 29084,
 user_id: 4,
 created_at: Tue, 15 Nov 2022 12:06:13.427829000 UTC +00:00,
 updated_at: Tue, 15 Nov 2022 12:06:13.434874000 UTC +00:00>
```

3. További nézetek és eseménykezelők hozzáadása

A bejelentkezett felhasználó menüjéből hiányzik két menüpont megvalósítása, a jelentkezésé és a naptáré, és menet közben rájöttünk, hogy jó lenne látni az adott félévre meghirdetett kurzusokat is.

Először hozzuk létre az útvonalakat. A kurzusok erőforrást ki kell bővítenünk egy, a meghirdetett kurzusokat listázó útvonallal, amely az index útvonalra hasonlít, és a `active` akcióra képezzük le, és egy, a kurzusra való jelentkezést lehetővé tevő útvonallal, amelyben, mivel konkrét kurzusról van szó, szerepelnie kell a kurzus azonosítójának. A menüben emellett szerepel a felhasználó naptára is, amelyben ugyan elhelyeztük a felhasználó azonosítótját, de arra nincs szükség, mert a session azonosítja a felhasználót. Ennek az eseménykezelője a felhasználók kontrollerének új, `calendar` nevű metódusa.

```
Rails.application.routes.draw do
  resources :courses do
    get 'register', to: 'courses#register', on: :member
    get 'active', to: 'courses#active', on: :collection
    end
  get 'users/calendar', to: 'users#calendar', as: 'calendar'
  end
```

A felhasználói menüt frissíthetjük az új helperekkel.

```
<%= link_to "Profile", edit_user_path(@user.id) %><br/>
<%= link_to "Subjects", subjects_path %><br/>
<%= link_to "Active courses", active_courses_path %><br>
<%= link_to "Application", subjects_path %><br/>
<%= link_to "Caledar", calendar_path %><br />
<%= link_to "Logout", logout_path%>
```

Az aktív kurzusok meghatározásához módosítanunk kell a szemeszter modellt megadva a szemeszter kezdetét és végét.

```
kovacsg@debian:~/pluto/app/views/users> rails g migration
    AddBeginAndEndsToSemester begins:date ends:date
       invoke  active_record
       create    db/migrate/20221115122524
            _add_begin_and_ends_to_semester.rb
kovacsg@debian:~/pluto/app/views/users> rails db:migrate
== 20221115122524 AddBeginAndEndsToSemester: migrating
   ========================
-- add_column(:semesters, :begins, :date)
   -> 0.0069s
-- add_column(:semesters, :ends, :date)
   -> 0.0036s
== 20221115122524 AddBeginAndEndsToSemester: migrated (0.0109s)
   ===============
```

Az adatbázisban állítsuk be ezeket a paramétereket a meglévő szemeszterekre (2-3. sor), és mivel csak egy kurzusunk van, hozzunk létre egy új kurzust a következő szemeszterre is (5.sor). Ezután azokat a kurzusokat keressük, ahol a szemeszter értéke egy olyan szemeszter, ahol a `begins` attribútum értéke a mai dátumnál nagyobb, és ezek közül hozzánk időben a legközelebb van.

```
kovacsg@debian:~/pluto/tmp/cache> rails c
Loading development environment (Rails 7.0.4)
irb(main):001:0> Semester.all
  Semester Load (0.2ms)  SELECT `semesters`.* FROM `semesters`
=>
[#<Semester:0x00007fe6d47e2cb0
  id: 1,
  name: "2022/2023_spring",
  year: 2023,
  season: "spring",
  created_at: Tue, 11 Oct 2022 11:25:45.696363000 UTC +00:00,
  updated_at: Tue, 11 Oct 2022 11:26:14.456019000 UTC +00:00,
  begins: nil,
  ends: nil>,
 #<Semester:0x00005566f27a0748
  id: 2,
  name: "2022/2023_fall",
  year: 2022,
  season: "fall",
  created_at: Tue, 11 Oct 2022 11:26:45.752992000 UTC +00:00,
  updated_at: Tue, 11 Oct 2022 11:26:45.752992000 UTC +00:00,
  begins: nil,
  ends: nil>]
irb(main):002:0> Semester.find(1).update(begins: Date.new
    (2022,9,5), ends: Date.new(2023,1,27))
  Semester Load (0.3ms)  SELECT `semesters`.* FROM `semesters`
      WHERE `semesters`.`id` = 1 LIMIT 1
  TRANSACTION (0.1ms)  BEGIN
  Semester Update (1.4ms)  UPDATE `semesters` SET `semesters`.`
      updated_at` = '2022-11-15 12:26:44.084315', `semesters`.`
      begins` = '2022-09-05', `semesters`.`ends` = '2023-01-27'
      WHERE `semesters`.`id` = 1
  TRANSACTION (0.5ms)  COMMIT
=> true
irb(main):003:0> Semester.find(2).update(begins: Date.new
    (2023,2,20), ends: Date.new(2023,7,14))
  Semester Load (0.3ms)  SELECT `semesters`.* FROM `semesters`
      WHERE `semesters`.`id` = 2 LIMIT 1
  TRANSACTION (0.1ms)  BEGIN
  Semester Update (1.5ms)  UPDATE `semesters` SET `semesters`.`
      updated_at` = '2022-11-15 12:27:18.040109', `semesters`.`
```

```
      begins` = '2023-02-20', `semesters`.`ends` = '2023-07-14'
      WHERE `semesters`.`id` = 2
  TRANSACTION (0.5ms)  COMMIT
=> true
irb(main):005:0> Course.create subject_id: 1, semester_id: 2, t:
    'practice', date1: Date.today, date2: Date.today+2.days,
    limit: 100, location1: 'Uszoda', location2: 'Uszoda'
  TRANSACTION (0.1ms)  BEGIN
  Subject Load (0.3ms)  SELECT `subjects`.* FROM `subjects` WHERE
       `subjects`.`id` = 1 LIMIT 1
  Semester Load (0.2ms)  SELECT `semesters`.* FROM `semesters`
      WHERE `semesters`.`id` = 2 LIMIT 1
  Course Create (1.8ms)  INSERT INTO `courses` (`subject_id`, `
      semester_id`, `t`, `date1`, `date2`, `location1`, `
      location2`, `limit`, `created_at`, `updated_at`) VALUES (1,
       2, 1, '2022-11-15 00:00:00', '2022-11-17 00:00:00', '
      Uszoda', 'Uszoda', 100, '2022-11-15 12:29:10.929037', '
      2022-11-15 12:29:10.929037')
  TRANSACTION (0.4ms)  COMMIT
=>
#<Course:0x00007fe6d489f590
 id: 2,
 subject_id: 1,
 semester_id: 2,
 t: "practice",
 date1: Tue, 15 Nov 2022 00:00:00.000000000 UTC +00:00,
 date2: Thu, 17 Nov 2022 00:00:00.000000000 UTC +00:00,
 location1: "Uszoda",
 location2: "Uszoda",
 limit: 100,
 created_at: Tue, 15 Nov 2022 12:29:10.929037000 UTC +00:00,
 updated_at: Tue, 15 Nov 2022 12:29:10.929037000 UTC +00:00>
irb(main):009:0> Semester.where('begins > ?', Date.today).first
  Semester Load (1.7ms)  SELECT `semesters`.* FROM `semesters`
      WHERE (begins > '2022-11-15') ORDER BY `semesters`.`id` ASC
       LIMIT 1
=>
#<Semester:0x00005566f243eff0
 id: 2,
 name: "2022/2023 fall",
 year: 2022,
 season: "fall",
 created_at: Tue, 11 Oct 2022 11:26:45.752992000 UTC +00:00,
 updated_at: Tue, 15 Nov 2022 12:27:18.040109000 UTC +00:00,
 begins: Mon, 20 Feb 2023,
 ends: Fri, 14 Jul 2023>
```

Az aktív kurzusok képernyő ugyanaz, mint a kurzusok képernyő, csak a megjelenítendő adatokban tér el, ezért újrahasznosítjuk a létező template-et

a `render` függvénnyel.

```
class CoursesController < ApplicationController
  # GET /courses or /courses.json
  def index
    @courses = Course.all
  end

  def active
    next_semester = Semester.where('begins > ?', Date.today).
        first
    @courses = Course.where(semester: next_semester)
    render template: 'courses/index'
  end
end
```

A kurzusra jelentkezés az összes aktív kurzus képernyőn lehetséges egy új linkre való kattintással, amely a kurzusra való regisztráció akcióra mutat, amelynek paramétere a kurzus, a felhasználót pedig a sessionből ismerjük.

```
<div id="courses">
  <% @courses.each do |course| %>
    <%= render course %>
    <p>
      <%= link_to "Show this course", course %>
      <% unless @user.courses.include? course %>
      <%= link_to "Register", register_course_path(course.id) %>
      <% end %>
    </p>
  <% end %>
  </div>
```

A kurzusok kontrollerében mivel a regisztráció művelet esetén a paraméterek között szerepel a kurzus azonosítója, módosítanunk kell a `before_action` listát, hogy a kurzus keresését e művelet előtt is végezze el. A regisztráció egyszerű, ha a felhasználó, kurzus páros még nem szerepel a kurzusok, felhasználók kapcsolótáblában, akkor hozzáadjuk. Ezt két irányból is megtehetjük, a felhasználó kurzusai közé vehetjük fel a kurzust, vagy a kurzus felhasználói közé a felhasználót, az eredmény azonos.

```
class CoursesController < ApplicationController
  before_action :set_course, only: %i[ show edit update destroy
      register ]

  def register
    unless @course.users.include? @user
        @course.users << @user
    end
    # @user.coursers << @course
```

```
    redirect_back fallback_location: active_courses_path
  end
end
```

A kurzusról való lejelenkezés hasonlóan történhez.

A órarendet egy külső API[3] segítségével oldjuk meg, ezért módosítjuk a `Gemfile`-t, majd futtatjuk a `bundle install` parancsot.

```
gem 'rails_calendar'
```

A felhasználók kontrollerben létrehozunk üres törzzsel egy `calendar` nevű akciót, és mivel üres a törzse, automatikusan renderelődni fog az azonos nevű template a nézetek közül. Hozzuk létre a `calendar.html.erb` nézetet az API dokumentációjában lévő példa alapján.

```
  <h1>Schedule</h1>
<%
  events = {
    '2022-11-15' => [ 'TODO_1', 'TODO_2' ],
    '2022-11-16' => [ 'TODO_3' ]
  }
%>

<%= rails_calendar(Date.new(2022, 11)) do |date| %>
  <% if events[date.to_s].present? %>
    <ul>
      <% events[date.to_s].each do |event| %>
        <li><%= event %></li>
      <% end %>
    </ul>
  <% end %>
<% end %>
```

A hátralévő teendőnk ezután a következő: a most a nézetben definiált `event` nevű lokális változót a kontroller akcióban inicializáljuk példányváltozóként a felhasználó adott havi eseményei alapján, illetve a stíluslapokat módosítva változtatnunk kell a megjelenésen.

[3] `https://github.com/rdiazv/rails_calendar`