Rails routing, kontrollerek
Gyakorlat

Kovács Gábor

2022. május 9.

1. Az adatmodell módosítása, kiegészítése

A felhasználói adatlapon szereplő nem attribútum enum deklarációját módosítjuk, hogy a férfi jelentse a hamis, a nő pedig az igaz értéket. A memóriában ezek után a férfi a `male`, a nő a `female` sztriggel lesz reprezentálva, és eközben az adatbázisban a 0 és az 1 értékek kerülnek be mint a bool típus értékeinek reprezentációi.

```
class Person < ApplicationRecord
  enum gender: {:male => false, :female => true }
end
```

Így a setterünk és getterünk már megfelelően működik.

```
irb(main):002:0> p = Person.first
  Person Load (0.3ms)  SELECT `people`.* FROM `people` ORDER BY `
      people`.`id` ASC LIMIT 1
=>
#<Person:0x000055fcdd2da840
...
irb(main):004:0> p.male?
=> false
irb(main):005:0> p.male!
  TRANSACTION (0.1ms)  BEGIN
  User Load (0.2ms)  SELECT `users`.* FROM `users` WHERE `users
      `.`id` = 2 LIMIT 1
  Person Update (8.3ms)  UPDATE `people` SET `people`.`gender` =
      FALSE, `people`.`updated_at` = '2023-05-09 10:16:12.609142'
       WHERE `people`.`id` = 2
  TRANSACTION (1.8ms)  COMMIT
=> true
irb(main):008:0> p.male?
=> true
```

```
irb(main):009:0> p
=>
#<Person:0x000055fcdd2da840
 id: 2,
 name: nil,
 birthdate: nil,
 gender: "male",
 into: nil,
 created_at: Tue, 02 May 2023 11:42:34.494717000 UTC +00:00,
 updated_at: Tue, 09 May 2023 10:16:12.609142000 UTC +00:00,
 user_id: 2>
irb(main):010:0>
```

```
kovacsg@debian:~/randi/app/models> rails db
MariaDB [randi_development]> select * from people;
+----+------+-----------+--------+------+---------------------
| id | name | birthdate | gender | into | created_at
                    | updated_at                  | user_id |
+----+------+-----------+--------+------+---------------------

|  2 | NULL | NULL      |      0 | NULL | 2023-05-02
   11:42:34.494717 | 2023-05-09 10:16:12.609142 |       2 |
+----+------+-----------+--------+------+---------------------

1 row in set (0.000 sec)
```

Az adatmodellünket az ismerős relációval bővítjük, amelyet felhasználó között valósítunk meg. Az alternatíva a felhasználói adatlapok közötti kapcsolat lett volna. A kapcsolatban az ismerős is egy felhasználó objektum, azonban ugyanarra a típusra két referenciát nem adhatunk meg. A megoldás, hogy egy kitalált, „barát” típusra hivatkozunk, amelyet a memóriában majd a felhasználó típus egy példányának jelölünk. Tehát egy kapcsolótáblára van szükségünk két idegen kulcssal. Ha okosan nevezzük el a migrációt, akkor a generált kódba nem kell belenyúlnunk.

```
kovacsg@debian:~/randi/app/models> rails g migration
    CreateJoinTableUsersFriends friend user
      invoke  active_record
      create    db/migrate/20230509101807
          _create_join_table_friends_users.rb
kovacsg@debian:~/randi/db/migrate> rails db:migrate
== 20230509101807 CreateJoinTableFriendsUsers: migrating
   ==========================
-- create_join_table(:friends, :users)
   -> 0.0244s
== 20230509101807 CreateJoinTableFriendsUsers: migrated (0.0247s)
    ============
```

```
kovacsg@debian:~/randi/app/models> rails db
MariaDB [randi_development]> desc friends_users
    -> ;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| friend_id | bigint(20)  | NO   |     | NULL    |       |
| user_id   | bigint(20)  | NO   |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.001 sec)
```

A felhasználó modell osztályában fel kell vennünk egy több-több kapcsolatot a felhasználók és az ismerős felhasználók között. A deklarációban jeleznünk kell, hogy mi a kapcsolótábla neve, mik az idegen kulcsok nevei, illetve azt, hogy az ismerős is ugyanennek a modellnek a példánya.

```
class User < ApplicationRecord
  has_and_belongs_to_many :friends, class_name: 'User',
      foreign_key: 'friend_id', join_table: 'friends_users',
      association_foreign_key:  'user_id'
end
```

Vegyünk fel pár felhasználót a `db/seeds.rb` fájlban, akik között létre tudjuk hozni az ismerős kapcsolatot.

```
for i in 1..10 do
  u = User.new username: "Valaki#{i}", email: "valaki#{i}@mail.
      bme.hu", password: 'titok', password_confirmation: 'titok'
  u.person = Person.new
  u.person.name = "Vala Ki #{i}"
  u.person.birthdate = Date.today
  u.person.female!
  end
```

Töltsük be az adatokat az adatbázisba, és nézzük meg, hogy létrejöttek-e.

```
kovacsg@debian:~/randi/db> rails db:seed
kovacsg@debian:~/randi/db> rails db
MariaDB [randi_development]> select * from users;
+----+----------+-----------------+-----------------------------------
| id | username | email           | encrypted_password
                          | created_at                  |
   updated_at                  | salt                        |
+----+----------+-----------------+-----------------------------------

|  2 | Senki    | senki@mail.bme.hu    | 2
   bdec43e317380b1f4bfcc5b057ca0c0291bbbff | 2023-05-02
   10:30:44.723526 | 2023-05-02 10:30:44.723526 |
   JRlZNX1fe19BIvP9lAnHXw== |
```

```
|  4 | Valaki    | valaki@mail.bme.hu   |
   b100ec40e67e680af6d35d34cf81f23a98d08eb9 | 2023-05-02
   11:28:40.864964 | 2023-05-02 11:28:40.864964 |
   ISqEaLXYzunEOj2hQ+98dQ== |
|  5 | Valaki1   | valaki1@mail.bme.hu  | 5020
   c89df2a3d6998a5834a37f2b768adc889ff5 | 2023-05-09
   10:32:15.627141 | 2023-05-09 10:32:15.627141 |
   RyHSc5OuMObPlRmlbHdAWQ== |
|  6 | Valaki2   | valaki2@mail.bme.hu  |
   f0ecc249843473f812463938e5c55784fb203ccf | 2023-05-09
   10:32:15.640011 | 2023-05-09 10:32:15.640011 |
   B8Wh1x4Gj1IrtIo2BlprqQ== |
|  7 | Valaki3   | valaki3@mail.bme.hu  |
   c3fca31d4fc38593af5c5a4d0ff5328243daa1a0 | 2023-05-09
   10:32:15.647549 | 2023-05-09 10:32:15.647549 | 6
   tHG0CfDUbEHAS2HXAeVjA== |
|  8 | Valaki4   | valaki4@mail.bme.hu  |
   fbc9eca2171f51976b7cdfb98f794a92785f930e | 2023-05-09
   10:32:15.653847 | 2023-05-09 10:32:15.653847 | 3rxl/2
   mpFi1HN8h605jomg== |
|  9 | Valaki5   | valaki5@mail.bme.hu  | 92
   b3608dfbdd66c522d1184549a64327a7650fe8 | 2023-05-09
   10:32:15.659941 | 2023-05-09 10:32:15.659941 |
   AUJeOXfefX6ocLn87XZHYA== |
| 10 | Valaki6   | valaki6@mail.bme.hu  | 32
   a1907fed5a4487aeb6e6acf21c8b489de7705a | 2023-05-09
   10:32:15.666336 | 2023-05-09 10:32:15.666336 |
   Fth3ofEzsu7bv5IRlQTtvA== |
| 11 | Valaki7   | valaki7@mail.bme.hu  |
   f81129c730e16794c589e82154c2f134791867f3 | 2023-05-09
   10:32:15.673677 | 2023-05-09 10:32:15.673677 |
   kDYnZIUc1xBNcv52z21vcg== |
| 12 | Valaki8   | valaki8@mail.bme.hu  | 5
   f1630c91246db48ab788597ba80393ccd25805c | 2023-05-09
   10:32:15.681946 | 2023-05-09 10:32:15.681946 | NrH/7
   acHrOVYzZ5Ryl//jg== |
| 13 | Valaki9   | valaki9@mail.bme.hu  |
   eec557dc9c2e61b751ec618f491eb8932b9362c6 | 2023-05-09
   10:32:15.689295 | 2023-05-09 10:32:15.689295 |
   ibw1jRZmKisMEGJD5zsYiQ== |
| 14 | Valaki10  | valaki10@mail.bme.hu | 608322
   b73d55c616bd87fb94220206676 1f5dc67 | 2023-05-09
   10:32:15.704612 | 2023-05-09 10:32:15.704612 | 2092
   ahsT0f3uBW0eoK+4mg== |
+----+-----------+----------------------+----------------------------------

12 rows in set (0.000 sec)

MariaDB [randi_development]> select * from people;
```

```
+----+-----------+---------------------+--------+------+---------------------+
| id | name      | birthdate           | gender | into | created_at          |
|                | updated_at                                 | user_id |
+----+-----------+---------------------+--------+------+---------------------+
|  2 | NULL      | NULL                |      0 | NULL | 2023-05-02
   11:42:34.494717 | 2023-05-09 10:16:12.609142 |       2 |
|  3 | Vala Ki 1 | 2023-05-09 |      1 | NULL | 2023-05-09
   10:32:15.630784 | 2023-05-09 10:32:15.630784 |       5 |
|  4 | Vala Ki 2 | 2023-05-09 |      1 | NULL | 2023-05-09
   10:32:15.642455 | 2023-05-09 10:32:15.642455 |       6 |
|  5 | Vala Ki 3 | 2023-05-09 |      1 | NULL | 2023-05-09
   10:32:15.649461 | 2023-05-09 10:32:15.649461 |       7 |
|  6 | Vala Ki 4 | 2023-05-09 |      1 | NULL | 2023-05-09
   10:32:15.655561 | 2023-05-09 10:32:15.655561 |       8 |
|  7 | Vala Ki 5 | 2023-05-09 |      1 | NULL | 2023-05-09
   10:32:15.661759 | 2023-05-09 10:32:15.661759 |       9 |
|  8 | Vala Ki 6 | 2023-05-09 |      1 | NULL | 2023-05-09
   10:32:15.668962 | 2023-05-09 10:32:15.668962 |      10 |
|  9 | Vala Ki 7 | 2023-05-09 |      1 | NULL | 2023-05-09
   10:32:15.675644 | 2023-05-09 10:32:15.675644 |      11 |
| 10 | Vala Ki 8 | 2023-05-09 |      1 | NULL | 2023-05-09
   10:32:15.684500 | 2023-05-09 10:32:15.684500 |      12 |
| 11 | Vala Ki 9 | 2023-05-09 |      1 | NULL | 2023-05-09
   10:32:15.699487 | 2023-05-09 10:32:15.699487 |      13 |
| 12 | Vala Ki 10 | 2023-05-09 |      1 | NULL | 2023-05-09
   10:32:15.707845 | 2023-05-09 10:32:15.707845 |      14 |
+----+-----------+---------------------+--------+------+---------------------+

11 rows in set (0.000 sec)
```

Végül nézzük meg, hogy mi lesz az a kód, amellyel az ismerősnek jelölés eseményt kezelő kontroller metódusban el kell helyeznünk. Keresünk magunknak egy felhasználót (1. sor), megnézzük az ismerőseinek listáját, amely üres (2. sor). A 3. sorban felveszünk ebbe a listába egy másik felhasználót. Ez a műveletet a másik irányban is el kell végezünk, vagyis a másik felhasználó ismerősei közé is fel kell vennünk a felhasználónkat. A 4. sorban látjuk, hogy a művelet sikerült. A 8. sorban megszüntetjük az ismerős kapcsolatot a felhasználó a tömbből való törlésével, majd a 9. sorban ellenőrizzük, hogy a művelet sikerült-e.

```
kovacsg@debian:~/randi/app/models$ rails c
Loading development environment (Rails 7.0.4.3)
irb(main):001:0> u = User.first
  User Load (0.2ms)  SELECT `users`.* FROM `users` ORDER BY `
      users`.`id` ASC LIMIT 1
=>
```

```
#<User:0x000055f180dded58
...
irb(main):002:0> u.friends
  User Load (0.9ms)  SELECT `users`.* FROM `users` INNER JOIN `
      friends_users` ON `users`.`id` = `friends_users`.`user_id`
      WHERE `friends_users`.`friend_id` = 2
=> []
irb(main):003:0> u.friends << User.last
  User Load (0.3ms)  SELECT `users`.* FROM `users` ORDER BY `
      users`.`id` DESC LIMIT 1
  TRANSACTION (0.1ms)  BEGIN
  User::HABTM_Friends Create (6.4ms)  INSERT INTO `friends_users`
       (`friend_id`, `user_id`) VALUES (2, 14)
  TRANSACTION (0.4ms)  COMMIT
=>
[#<User:0x000055f181118910
  id: 14,
  username: "Valaki10",
  email: "valaki10@mail.bme.hu",
  encrypted_password: "[FILTERED]",
  created_at: Tue, 09 May 2023 10:32:15.704612000 UTC +00:00,
  updated_at: Tue, 09 May 2023 10:32:15.704612000 UTC +00:00,
  salt: "[FILTERED]">]
irb(main):004:0> u.friends
=>
[#<User:0x000055f181118910
  id: 14,
  username: "Valaki10",
  email: "valaki10@mail.bme.hu",
  encrypted_password: "[FILTERED]",
  created_at: Tue, 09 May 2023 10:32:15.704612000 UTC +00:00,
  updated_at: Tue, 09 May 2023 10:32:15.704612000 UTC +00:00,
  salt: "[FILTERED]">]
irb(main):008:0> u.friends.delete(User.last)
  User Load (0.3ms)  SELECT `users`.* FROM `users` ORDER BY `
      users`.`id` DESC LIMIT 1
  TRANSACTION (0.1ms)  BEGIN
  User::HABTM_Friends Delete All (1.7ms)  DELETE FROM `
      friends_users` WHERE `friends_users`.`friend_id` = 2 AND `
      friends_users`.`user_id` = 14
  TRANSACTION (0.4ms)  COMMIT
=>
[#<User:0x000055f17f2e2090
  id: 14,
  username: "Valaki10",
  email: "valaki10@mail.bme.hu",
  encrypted_password: "[FILTERED]",
  created_at: Tue, 09 May 2023 10:32:15.704612000 UTC +00:00,
  updated_at: Tue, 09 May 2023 10:32:15.704612000 UTC +00:00,
```

```
  salt: "[FILTERED]">]
irb(main):009:0> u.friends
=> []
```

2. Útvonalak módosítása

Az útvonalak egy részét már az előző gyakorlatot módosítottuk. Az elfelejtett jelszó képernyő és annak eseménykezelőjéhez tartozó útvonakat is kiegészítjük az `as` opció által generált útvonal helperekkel. Kezdőoldalnak a hello képernyőt állítjuk be.

```
Rails.application.routes.draw do
  get 'users/forgotten', to: 'users#forgotten', as: 'forgotten'
  post 'users/send_forgotten', to: 'users#send_forgotten', as: '
      send_forgotten'
  root "say#hello"
end
```

A vendégfelhasználó és a bejelentkezett felhasználó menüiben ezután módosítjuk a képernyők kialakítása során beállított statikus értékeket az útvonal helperek által generált értékekre.

```
<%= link_to "Forgotten password", forgotten_path %>
```

Az elfejetett jelszó képernyő formjában az akciót módosítjuk, hogy azt az új útvonal helperrel generáljuk.

```
<%= form_tag send_forgotten_path, method: :post do %>
```

A belejentkezett felhasználó menüjének nézetében (`app/views/layouts/_user_menu.html.erb`) még paraméter nélkül hívjuk meg az `as` opció utáni útvonal helpert, ezért azt is módosítjuk. A `@user` példányváltozót az `ApplicationController` `find_user` függvényében inicializáltuk, így felhasználhatjuk, az `id` attribútum értékét hozzárendeljük az útvonal `:id` paraméteréhez.

```
Hello, <%= @user.username %>!

<p><%= flash[:notice] %></p>

<p><%= link_to 'Profile', edit_profile_path(@user.id) %></p>
<p><%= link_to 'Datasheet', person_path(@user.person.id) %></p>
<p><%= link_to "Edit datasheet", edit_person_path(@user.person.id
    ) %></p>
<p><%= link_to "Search partner", people_path %></p>
<p><%= link_to "Logout", logout_path %></p>
```

Az erőforrás alapú útvonalakat a `resources` függvény generálja, mégpedig hetet. Az erőforrás nevével megegyező útvonalhoz HTTP GET, illetve HTTP POST művelettel férhetünk hozzá. Az előbbi az erőforrás összes adatbázisbeli rekordját adja vissza egy listában, tömbben, az utóbbi pedig az erőforrás típusából egy új példányt hoz létre. Ha ehhez az útvonalhoz egy azonosítót adunk hozzá, akkor az erőforrás típusának az azonosító által azonosított példányán végezhetünk el műveleteket. A GET megmutatja az erőforrást, a DELETE törli az erőforrást, a PUT módosítja az erőforrást egy form adataival. Azt a képernyőt, ahol a módosítás elérhető szintén a GET művelettel érhetjük el, viszont ez ütközik az erőforrás megmutatásával, ezért hozzáteszünk az útvonalhoz egy `edit` szuffixet. Ehhez hasonlóan ütközés van, amikor azt a képernyőt akarjuk betölteni, ahol az erőforrás típusából új példányt létrehozó form van, az az összes erőforrás listájával üzközik, ezért ott egy `new` szuffixet adunk hozzá.

```
resources :people # only: [] vagy #except: []
# get 'people', to: 'parties#index', as: 'people'
# get 'people/:id', to: 'people#show', as: 'person'
# delete 'people/:id', to: 'people#destroy'
# get 'people/new', to: 'people#new', as: 'new_person'
# post 'people', to: 'people#create'
# put 'people/:id', to: 'people#update', as: 'update_person'
# get 'people/:id/edit, to: 'people#edit', as: 'edit_person'
```

3. Lapozás

A felhasználói adatlapok lisjáta képernyőn (`people/index.html.erb`) megjelenő adatlapok számára nincs felső korlát, ez azonban hamar betöltheti a rendelkezésre álló függőleges teret, ezért bevezetjük ezek több lapra való tördelését. Egy oldalon legfeljebb kettő, talán három felhasználó adatlapja fér el, ezért csak kettőt 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 `people/index.html.erb` nézeten felvesszük a lapozó linkeket a `@people` példányváltozóra.

```
<%= will_paginate @people %>
```

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

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

```
class PeopleController < ApplicationController
  def index
    @people = Person.get_page(params[:page])
  end
end
```

Az adatlap modell osztályt kiegészítjük a töredéket lekérdező osztálymetódussal, melynek egy paramétere van, a töredék sorszáma. Az adatlapok töredékét úgy kérdezzük le, hogy egy töredék pontosan 2 elemet tartalmazzon.

```
class Person < ApplicationRecord
  def Person.get_page(page)
    Person.all.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 `: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 :people do
    get 'page/:page', on: :collection, to: 'people#index', as: '
        page'
  endend
```

4. Fájlok fel- és letöltése

Minden felhasználónak 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 felhasználói adatlapot, amelyhez kapcsolódik, a

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

feltöltött fájl elérési útvonalát a fájlrendszeren, az eredeti nevét és méretét. A csatolmány a fájlrendszeren a `public` könyvtárban létrehozott `data` alkönyvtárban tároljuk, mert bináris adatot az adatbázis sem tud a fájlrendszernél hatékonyabban tárolni.

```
kovacsg@debian:~/randi/config> rails g model attachment path:
    string name:string mime:string size:integer person:references
       invoke  active_record
       create    db/migrate/20230509105829_create_attachments.rb
       create    app/models/attachment.rb
       invoke    test_unit
       create      test/models/attachment_test.rb
       create      test/fixtures/attachments.yml
kovacsg@debian:~/randi/db/migrate> rails db:migrate
== 20230509105829 CreateAttachments: migrating
   ===============================
-- create_table(:attachments)
   -> 0.0087s
== 20230509105829 CreateAttachments: migrated (0.0090s)
   ======================
```

A csatolmány egy-egy kapcsolatban áll a felhasználói adatlappal, a kapcsolat az `Attachment` modellben automatikusan létrejött, a felhasználói adatlap modelljéhez azonban hozzá kell adnunk. A csatolmány feltöltés után átmeneti tárolására felveszünk egy csak a memóriában használható attribútumot.

```
class Person < ApplicationRecord
  attr_accessor :uploaded_image
  has_one :attachment
end
```

A felhasználói adatlap nézetei között módosítjuk a fájlfeltöltés mezőt, hogy az ezt az `uploaded_image` mezőt használja.

```
  <div>
    <%= form.label :uploaded_image %>
    <%= form.file_field :uploaded_image %>
  </div>
```

A kontrollerben ezt a paramétert fehérlistára kell tennünk, hogy az bekerüljön a `params` hash-be.

```
class PeopleController < ApplicationController
  private
    # Only allow a list of trusted parameters through.
    def person_params
      params.require(:person).permit(:name, :birthdate, :gender,
          :into, :uploaded_image)
```

```
    end
end
```

A csatolány fel- és letöltésének eseménykezelőjét a felhasználói adatlap kontrollerével valósítjuk meg, amelyben három akciót értint a feltöltés: a létrehozás `create` nevű, a módosítás `update` nevű és a letöltés `download` nevű metódusa. A scaffolddal automatikusan generált kódban a `@person.save` illetve a `@person.update` metódusok meghívják a felhasználói adatlap mentés metódusát, amely előtt a feltöltött temporális fájlt el tudjuk tárolni azáltal, hogy meghívjuk a csatolmány modell egy osztálymetódusát a temporális fájl objektumával és az adatlappal magával mint paraméterrel.

```
class Person < ApplicationRecord
  before_save :a
  def a
    Attachment.save_file self.uploaded_image, self
  end
end
```

A Rails alkalmazáson belüli útvonalakra a fájlrendszeren a következőképp hivatkozhatunk:

```
kovacsg@debian:~/randi/app/models> rails c
Loading development environment (Rails 7.0.4.3)
irb(main):001:0> Rails.root
=> #<Pathname:/home/kovacsg/randi>
irb(main):002:0> Rails.root.join("public")
=> #<Pathname:/home/kovacsg/randi/public>
irb(main):003:0> Rails.root.join("public").join('data')
=> #<Pathname:/home/kovacsg/randi/public/data>
irb(main):004:0> Rails.root.join("public", 'data')
=> #<Pathname:/home/kovacsg/randi/public/data>
```

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 :person

```

```
  def self.save_file(upload, p)
    return if upload.nil?
    dir = Rails.root.join('public', 'data')
    unless File.exists? dir
      Dir.mkdir dir
    end
    fname = upload.original_filename
    a = Attachment.new name: fname, mime: upload.content_type,
        person: p
    a.save
    path = File.join(dir, p.id.to_s)
    File.open(path, 'wb') do |f| f.write(upload.read) end
    a.update path: path, size: File.size(path)
  end
end
```

Hozzuk létre a most felhasználói adatlapok avaratjait letöltő útvonalat! Ez egy konkrét felhasználó objektumra vonatkozik, amelynek van `id` attribútuma, ezért az `on` opció értékének a `:member` szimbólumot adjuk meg.

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

A csatolmányokat az `img` HTML tagek `src` attribútumában használjuk például az `people/_person.html.erb` nézetben, amely a felhasználói profilt rendeleli.

```
<img src="<%= download_person_path(person.id) %>" width="160"
    height='160'>
```

A letöltés akciót a felhasználói adatlapok kontrollerbe helyezzük el. A csatolmány egy felhasználói adalap `id`-val azonosított példányához tartozik, ezért bővítjük a `before_action` listáját az új `download` metódusunkkal, hogy annak elején ne kelljen a keresést elvégeznünk. E felhasználói adatlap objektumhoz asszociált avatar 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 PeopleController < ApplicationController
  before_action :set_person, only: %i[ show edit update destroy
      download ]
```

```
  def download
    a = @person.attachment
    unless a.nil?
      send_file a.path, type: a.mime, filename: a.name,
          disposition: :inline
      return
    end
    head 404
  end
end
```

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

```
kovacsg@debian:~/randi/public/data$ rails c
Loading development environment (Rails 7.0.4.3)
irb(main):004:0> p = Person.find 12
  Person Load (0.4ms)  SELECT `people`.* FROM `people` WHERE `
      people`.`id` = 12 LIMIT 1
=>
#<Person:0x0000564ac2fa6550
...
irb(main):005:0> p.attachment
  Attachment Load (0.4ms)  SELECT `attachments`.* FROM `
      attachments` WHERE `attachments`.`person_id` = 12 LIMIT 1
=>
#<Attachment:0x0000564ac2e75078
 id: 1,
 path: "/home/kovacsg/randi/public/data/1",
 name: "rails_hello.png",
 mime: "image/png",
 size: 29154,
 person_id: 12,
 created_at: Tue, 09 May 2023 11:20:08.961231000 UTC +00:00,
 updated_at: Tue, 09 May 2023 11:20:08.974074000 UTC +00:00>
irb(main):006:0>
```

5. További nézetek és eseménykezelők hozzáadása

A felhasználói adatlapon a nem attribútumot újraértelmeztük, azonban ez még nem tükröződik a formban. A jelölőnégyzetet rádiógombokra cseréljük le.

```
  <div>
    <%= form.label :gender, 'male', style: "display:␣block" %>
    <%= form.radio_button :gender, "male" %>
```

```
    <%= form.label :gender, 'female', style: "display:␣block" %>
    <%= form.radio_button :gender, "female" %>
  </div>
```

A felhasználói adatlapon az ismerősnek jelölés, illetve az ismerésnek jelölés törlése műveletek útvonalait és eseménykezelőit fel kell vennünk. Kezdjük az útvonalakkal. Ismerőse egy bejelentkezett felhasználónak lehet, akinek van `id`-val rendelkező adatlapja, ezért mindkét útvonal `:member` típusú lesz.

```
Rails.application.routes.draw do
  resources :people do
    get 'page/:page', on: :collection, to: 'people#index', as: '
        page'
    get 'download', on: :member, to: 'people#download', as: '
        download'
    get 'add_friend', on: :member, to: 'people#add_friend', as: '
        add_friend'
    get 'remove_friend', on: :member, to: 'people#remove_friend',
         as: 'remove_friend'
  end
end
```

A kontrollerbe ezután fel kell vennünk a `add_friend` és a `remove_friend` metódusokat, amelyeket szintén felveszünk a `before_action` listájába, merthogy szükségünk van az adatlap objektumára. A két akció közül egyik sem rendelkezik önálló képernyővel, az előző képernyőre megyünk vissza. A metódusok törzse pedig a korábban konzolon már vizsgált tömbhöz hozzáadás, illetve tömbből eltávolítás műveletek lesznek.

```
class PeopleController < ApplicationController
  before_action :set_person, only: %i[ show edit update destroy
      download add_friend remove_friend ]

  def add_friend
    my_user = @person.user
    if @user != my_user
      @user.friends << my_user
      my_user.friends << @user
    end
    redirect_back fallback_location: hello_path
  end

  def remove_friend
    @user.friends.delete(@person.user)
    @person.user.friends.delete(@user)
    redirect_back fallback_location: hello_path
  end
end
```