HTML és Rails
Gyakorlat

Kovács Gábor

2022. március 29.

A gyakorlat célja, hogy kialakítsuk a félév során megoldandó feladat képernyőit HTML-ben, ahol lehet Rails metódusok felhasználásával.

Az első lépés a webalkalmazásunk keretének kialakítása ezt a nézetek között az alkalmazásszintű nézetben, vagyis a `layouts/application.html.erb`-ben tehetjük meg. Rendezzük el úgy az oldalunkat, hogy legyen benne egy fejrész, egy központi rész, amely bal oldalon egy keskeny menüsávból áll, jobb oldalon a tartalomból, és egy lábrész. Az elrendezést `div`-ekkel valósítjuk meg, mindegyikhez egyedi `id`-t rendelve.

```
<div id="page">
  <div id="header"></div>
  <div id="main">
    <div id="menu">
    </div>
    <div id="content">
      <%= yield %>
    </div>
  </div>
  <div id="footer">
    RoR, 2022
  </div>
</div>
```

Második lépésként készítsük el az oldal stíluslapját, amivel ezek a helyükre kerülnek, és helyezzünk el benne minimális mennyiségű formázási információt. Az oldal legyen 800 pixel széles. A fejrész legyen világosszürke és 100 képpont magas, és helyezzünk el benn egy banner képet. Az oldal központi része legyen 600 pixel magas. A menüsávot a központi részben, a fejléc alatt helyezzük el, és az vízszintesen a szélesség 24%-át foglalja el, magasságát a központi rész magassága definiálja. Az oldal tartalmi része világosszürke

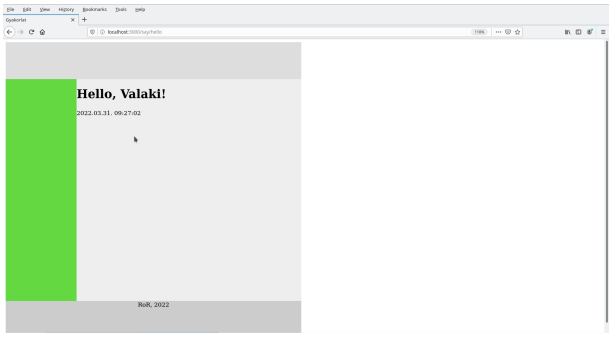
háttérrel rendelkezzen, és a menütől jobbra helyezkedjen el a vízszintesen a szélesség 76%-át elfoglalva. A lábrészben a szöveget igazítsuk középre, és legyen az is 100 pixel magas, valamint a fejlécnél világosabb szürke színű.

```
div#page {
        width: 800px;
}
div#header {
        height: 100px;
        background-color: #ddd;
}
div#footer {
        height: 100px;
        background-color: #ccc;
        text-align: center;
        clear: both;
}
div#main {
        height: 600px;
}
div#menu {
        float: left;
        width: 24%;
        height: 100%;
        background-color: #0d0;
}
div#content {
        float: left;
        width: 76%;
        height: 100%;
        background-color: #eee;
}
```

Az így kialakított elrendezést például az 1. ábra mutatja.

Két felhasználótípusra készülünk fel egyelőre, egy látogatóra és egy bejelentkezett felhasználóra, az utóbbiak korábban keresztülmentek egy regisztrációs folyamaton. A látogató csak böngészhet, bejelentkezhet és jelszóemlékeztetőt kérhet. A bejelentkezett felhasználók számára több funkciót is elérhetővé teszünk.

Kezdjük a látogató menüjével. Helyezzünk el a menüben egy a belépést lehetővé tevő formot! Ezt részleges rendereléssel tesszük meg. A formot tartalmazó fájl alkalmazás szinten kezeljük, ezért a `layouts` könyvtárban helyezzük el, így a be nem jelentkezett felhasználó bármelyik oldalon bejelent-



1. ábra. Az oldal elrendezésének kialakítása

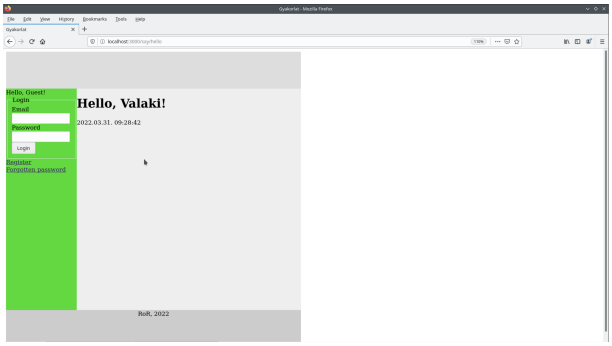
kezhet. A Rails konvenció szerint a részlegesen renderelt állományok neve aláhúzásjellel kezdődik. Legyen a fájlunk neve ezért `_guest.html.erb`! A form tartalmazzon egy felhasználónévre utaló címkét és egy szövegbeviteli mezőt, valamint egy a jelszóra utaló címkét és jelszóbeviteli mezőt, továbbá egy Login feliratú nyomógombot. A formot a `form_tag` Rails helperrel valósítjuk meg, aminek első paramétere a formot kezelő URL, vagyis a form `action` attribútuma, illetve adjuk meg, hogy HTTP POST-tal kívánjuk elküldeni. A form mezőit rendre a `label_tag`, `text_field_tag`, `password_field_tag` és `submit_tag` helperekkel hozzuk létre, és a beviteli mezőket 14 karakter hosszúra korlátozzuk. A be nem jelentkezett felhasználónak tegyük lehetővé az elfelejtett jelszó visszaszerzését, ezt egy link hozzáadásával tesszük meg.

```
Hello, Guest!

<fieldset>
  <legend>Login</legend>
  <%= form_tag '/sessions/create', method: :post do %>
    <%= label_tag 'email', 'Email' %>
    <%= text_field_tag 'email', '', size: 14 %>
    <%= label_tag 'password', 'Password' %>
    <%= password_field_tag 'password', '', size: 14 %>
    <%= submit_tag 'Login' %>
  <% end %>
</fieldset>

<%= link_to "Register", '/users/new' %><br/>
<%= link_to "Forgotten password", '/users/forgotten' %>
```

Ezután a `menu` azonosítójú `div`-ben meghivatkozhatjuk ezt az oldalt. A Rails konvenció szerint az aláhúzásjelet el kell hagynunk.

```
<div id="menu">
    <%= render '/layouts/guest_menu' %>
</div>
```

A vendégfelhasználó menüjének megvalósítását a 2. ábra mutatja.



2. ábra. A vendégfelhasználó menüje

Modellezzük azt az esetet is, amikor egy felhasználó már bejelentkezett. Ezt egy, a `helpers/application_helper` állományban elhelyezendő saját helper metódussal tesszük meg `logged_in?`. Itt egyelőre manuálisan állítjuk, hogy be van-e jelentkezve a felhasználó. A metódus értelemszerűen boolean visszatérési értékű.

```
module ApplicationHelper
  def logged_in?
    true
  end
end
```

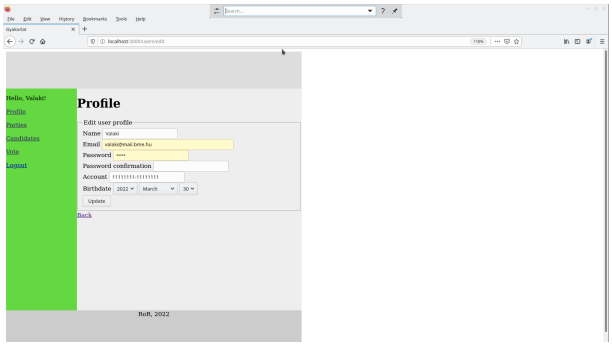
Ezt visszavezetve a keretbe a menüt megvalósító `menu`-ban a következő módosítást végezzük el. Így a helper módosításával be, illetve ki tudunk lépni az oldalról. „Jelentkeztessük” be és ki a felhasználót, hogy ellenőrizhessük, hogy a megfelelő menü jelenik-e meg a vendég és a felhasználó számára.

```
<div id="menu">
  <% if logged_in? %>
    <%= render 'layouts/user_menu' %>
  <% else %>
    <%= render 'layouts/guest_menu' %>
  <% end %>
</div>
```

A bejelentkezett hallgató felhasználó menüjét a vendéghez hasonlóan beágyazott nézettel hozzuk létre (`_user_menu.html.erb`). Egyelőre öt akciót definiálunk: a profiloldal megtekintését, a pártlisták, a jelöltek listájának megtekintését, a szavazást, valamint a kijelentkezést.

```
<p>Hello, user!</p>

<%= link_to "Profile", '/users/edit' %><br/>
<%= link_to "Add new bee pasture", new_beepasture_path %><
    br/>
<%= link_to "All beepastures", beepastures_path %><br/>
<%= link_to 'Logout', '/sessions/destroy' %>
```

A bejelentkezett hallgató felhasználó menüjének megvalósítását a 3. ábra mutatja.



3. ábra. A bejelentkezett felhasználó menüje

Nézzük meg a be nem lépett felhasználó létrehozásának folyamatát! Az előző gyakorlat alkalmával már létrehoztuk a felhasználó modellünk kezdetleges változtatás, így arra már tudhatunk hivatkozni egy Rails formban, amely az MVC tervezési minta szerint szorosan kapcsolódik a nézethez. Hozzuk létre a felhasználó nézetét és fontosabb akcióit a következő paranccsal:

```
kovacsg@debian:~/gyakorlat/app/views> rails g controller
    users new forgotten edit
Running via Spring preloader in process 17987
      create  app/controllers/users_controller.rb
       route  get 'users/new'
get 'users/forgotten'
get 'users/edit'
      invoke  erb
      create    app/views/users
```

```
      create    app/views/users/new.html.erb
      create    app/views/users/forgotten.html.erb
      create    app/views/users/edit.html.erb
      invoke  test_unit
      create    test/controllers/users_controller_test.rb
      invoke  helper
      create    app/helpers/users_helper.rb
      invoke    test_unit
      invoke  assets
      invoke    scss
      create      app/assets/stylesheets/users.scss
```

A parancs futtatásával létrejött az `users` kontroller és a hozzá kapcsolódó nézetek köztük az új felhasználó létrehozását lehetővé tevő `new`, a felhasználói profil szerkesztését megvalósító `edit`, és az elfelejtett jelszó esetén az email címet elkérő `forgotten` nézet. Az tervezői kérdés, hogy az elfelejtett jelszó kezelését a felhasználók kontrollere részének tekintjük, vagy önálló kontrollert hozunk létre számára. A gyakorlaton amellett döntöttünk, hogy az elfelejtett jelszó kerüljön a felhasználók kontrollerébe.

Hozzunk mindjárt létre a regisztrációs nézetet! Legyen egy címsorunk, amely elmondja a felhasználónak, hogy melyik oldalon van. Az esetleges hibaüzeneteknek tartsunk fenn helyet. Ezután egy `fieldset`-ben definiáljuk egy formot, amely ez esetben egy konkrét, létező modellhez van kötve. Ezt a `form_for` Rails helperrel tehetjük meg. Ennek első paramétere egy modell objektum vagy annak neve szimbólum formájában, második paramétere a formhoz kötött akció, amely legyen a `users` kontroller (ezt nem kell leírnunk, mert az új felhasználó létrehozása akció kontrollere ugyanaz) `create` akciója, a harmadik paramétere a HTTP metódus, ami POST. A metódus blokkjának van egy paramétere a `form`, amin kereszül definiáljuk fogjuk az űrlap elemeit. Legyen a hat elem rendre a következő: egy 20 karakter széles, a felhasználnévre vonatkozó szövegbeviteli mező a hozzá kapcsolódó címkével, egy 20 karakter széles, a felhasználó email címére vonatkozó szövegbeviteli mező a hozzá kapcsolódó címkével, két darab 20 karakter széles jelszóbeviteli mező a hozzájuk kapcsolódó címkével, egy a bankszámlaszámra vonatkozó szövegbeviteli mező a hozzá tartozó címkével, és egy HTML alapon készült, három legördülő menüből álló dátumbeviteli mező, amely később leváltható lesz JavaScript alapú dátumválasztóra. A két jelszómező eltérő azonosítóval rendelkezzék, az egyik prefixe `_confirmation`-re végződjék.

Ha a felhasználó meggondolná magát, és megsem kívánná regisztrálni magát, egy `Back` feliratú linkkel biztosítjuk számára a lehetőséget az előző oldalra való visszatérésre.

```
<h1>Registration</h1>

<fieldset>
  <legend>Register a new user</legend>
  <%= form_for @user, url: create_user_path, method: :post
      do |form| %>
    <div>
      <%= form.label :name %>
      <%= form.text_field :name, size: 20 %>
    </div>
    <div>
      <%= form.label :email %>
      <%= form.text_field :email, size: 40 %>
    </div>
    <div>
      <%= form.label :password %>
      <%= form.password_field :password, size: 20 %>
    </div>
    <div>
      <%= form.label :password_confirmation %>
      <%= form.password_field :password_confirmation, size:
          20 %>
    </div>
    <div>
      <%= form.label :account %>
      <%= form.text_field :account, size: 20 %>
    </div>
    <div>
      <%= form.label :birthdate %>
      <%= form.date_select :birthdate %>
    </div>
    <%= form.submit "Register" %>
  <% end %>
</fieldset>

<%= link_to "Back", :back %>
```

A form helper paraméterei között URL-nek egy függvénynevet adtunk meg, amely generálja a felhasználó létrehozási eseményét kezelő URL-t. Ezt a `config/routes.rb` fájlban adjuk meg. A `get`, `post`, `put` a HTTP műveletre vonatkozik, utána az útvonalra vonatkozó sztringet látjuk. A `to` opció azt határozza meg, hogy ez a sztring mely kontroller, mely akcióját kell, hogy kiváltsa beérkező HTTP kérés esetén, a `#` karakter előtt a kontroller neve,

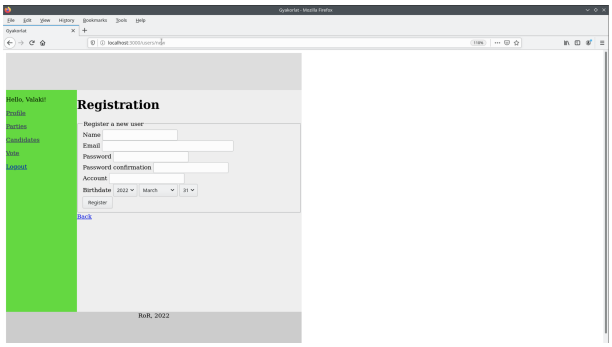
utána az akció neve kell, hogy álljon. Az `as` opció az útvonal elnevezése a karbantarthatóság javítása végett, hogy csak ebben a fájlban kelljen módosítani az útvonalat, ha az szükséges lenne, a többi fájlban ne. Az opció értéke két függvényt generál, a megadott név `_path`, illetve `_url` szuffixszel ellátott azonosítóval, amelyek az URL útvonal részét, illetve a teljes URL-t generálják.

```
Rails.application.routes.draw do
  get 'users/new', to: 'users#new', as: 'register'
  post 'users/create', to: 'users#create', as: 'create_user
      '.
  get 'users/edit', to: 'users#edit', as: 'profile'
  put 'users/update', to: 'users#update', as: '
      update_profile'
  get 'users/forgotten', to: 'users#forgotten', as: '
      forgotten'
  post 'users/send_forgotten', to: 'users#send_forgotten',
      as: 'send_forgotten'
end
```

Ahhoz, hogy az űrlap megjelenjen, a kontrollerben inicializánunk kell a `@user` példányváltozót.

```
class UsersController < ApplicationController
  def new
    @user = User.new
  end
end
```

A felhasználói regisztráció nézetét a 4. ábra mutatja.



4. ábra. A regisztráció nézete

A létrejött oldal HTML forrását tekintve a következőt látjuk. A formok

mezőinek `name` és `id` attribútuma tartalmazza a modell nevét és a mező nevét. A név attribútum Ruby hash mintájára készült el, a modell nevének hash-ére hivatkozik a mező Rails forrásban megadott neve. Az általunk megadott mezőkön kívül létrejött két hidden mező is, amelyek a form használójának hitelesítését hivatottak ellenőrizni. A visszalépés itt JavaScripttel valósul meg. A forrást megtekintve láthatjuk, hogy a `:password_confirmation` szimbólumból a Rails automatikusan a *Password confirmation* szöveget állította elő. A stringek és a szimbólumok így ezen elv mentén felcserélhetők a form helperek argumentumlistájában.

A form eseményét a Rails konvenció szerint a `create` kontroller metódus fogja kezelni. Ez még nem létezik, ezért definiáljuk azt egyelőre üres törzzsel.

A felhasználói profil szerkesztésének nézetében (`edit.html.erb`) található form szinte teljesen megegyezik az új felhasználót létrehozó formmal. Az email cím módosítását inaktívvá tehetjük, illetve az eseménykezelő kontroller akciót kell módosítanunk. A form eseményét a Rails konvenció szerint a `update` kontroller metódus fogja kezelni, erre létrehozzuk az útvonalat. Ez még nem létezik, ezért definiáljuk ezt is egyelőre szintén üres törzzsel. Ezen kívül a nézetben a feliratokat kell még átírnunk regisztrációról profil szerkesztésére.

```
<h1>Profile</h1>

<fieldset>
  <legend>Edit user profile</legend>
  <%= form_for @user, url: update_profile_path, method: :
      put do |form| %>
    <div>
      <%= form.label :name %>
      <%= form.text_field :name, size: 20 %>
    </div>
    <div>
      <%= form.label :email %>
      <%= form.text_field :email, size: 40 %>
    </div>
    <div>
      <%= form.label :password %>
      <%= form.password_field :password, size: 20 %>
    </div>
    <div>
      <%= form.label :password_confirmation %>
      <%= form.password_field :password_confirmation, size:
          20 %>
    </div>
```

```
    <div>
      <%= form.label :account %>
      <%= form.text_field :account, size: 20 %>
    </div>
    <div>
      <%= form.label :birthdate %>
      <%= form.date_select :birthdate %>
    </div>
    <%= form.submit "Update" %>
  <% end %>
</fieldset>

<%= link_to "Back", :back %>
```

Mivel a `form_for` Rails helper metódust használtuk a form létrehozására a `new` és az `edit` nézetekben, ezért szükséges a megfelelő kontroller akciókban a `@user` példányváltozó inicializálása. Ezeket egyelőre ne adatbázisból tegyük meg, hanem statikus tartalommal töltsük fel. Míg a `new` esetén a felhasználó még nem lézetik az adatbázisban, attribútumai inicializálatlanok, ezért elégséges egy frissen létrehozott példány használata, addig az `edit` esetén már ki kell töltenünk a struktúra mezőit beleértve az adatbázisbeli azonosító `id` attribútumot is. A `show` akcióban is inicializáljuk a `@user` példányváltozót, a `index` akcióban pedig a `@users` példányváltozót, amely egy User típusú objektumokat tartalmazó tömb.

```
class UsersController < ApplicationController
  def edit
    @user = User.new
    @user.id = 1
    @user.username = 'senki'
    @user.email = 'senki@mail.bme.hu'
    @user.password = 'titok'
  end

  def forgotten
    @user = User.new
  end
end
```

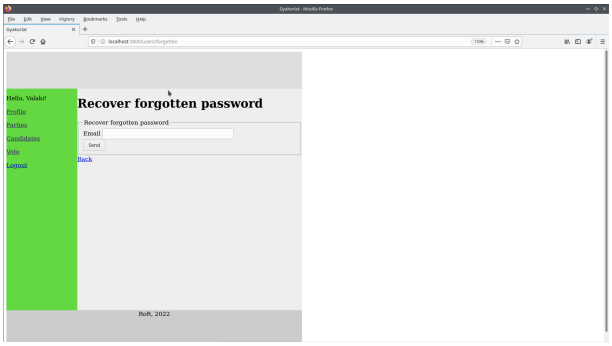
A felhasználói profiloldal szerkesztésének nézetét az **??**. ábra mutatja. Láthatjuk, hogy a Rails automatikusan inicializálta a form mezőit, ahol a hozzájuk tartozó érték elérhető volt – a jelszó mezők kivételével.

Ezután alakítsuk ki az elfejetett jelszó oldalt is. Itt egyszerűbb a formunk a beléptetésnél, csak az email címet tartalmazza.

```
<h1>Forgotten password</h1>
<fieldset>
  <legend>Please give your email address</legend>
  <%= form_for @user, url: '/users/send_forgotten', method:
      :post do |form| %>
    <div>
      <%= form.label :email %>
      <%= form.text_field :email, size: 20 %>
    </div>
    <%= form.submit "Send" %>
  <% end %>
</fieldset>

<%= link_to "Back", :back %>
```

Az elfelejtett jelszó kiküldését a form eseményét kezelő kontroller akció, a `send_forgotten` teszi majd meg, amit fel kell vennünk a kontroller osztályába egyelőre üres törzzsel.

Az elfelejtett jelszó nézetét a 5. ábra mutatja.



5. ábra. Az elfelejtett jelszó nézete

Az előző gyakorlaton már létrehoztuk a pártok modelljét, és automatikusan generált nézeteit. Most az `index`, illetve a `show` képernyőket újraírjuk. Mivel a Rails adatbáziskezelését még nem ismerjük, előbb cseréljük le a `PartiesController`-ben az automatikusan generált adatbázisműveleteket tartalmazó kódrészleteket statikus adatokra. Ezt két helyen kell megtennünk, az `index` akcióban, ahol a `@parties` példányváltozó méhlegelő példányokat tartalmazó tömbjét hozzuk létre, illetve a `before_action` helper függvény által kijelölt `set_party` azonosítójú metódusban, amely lefut mind a `show`, mind az `edit` nézet betöltésekor, itt csak a `@party` példányváltozóhoz kell

egy topik példányt rendelnünk.

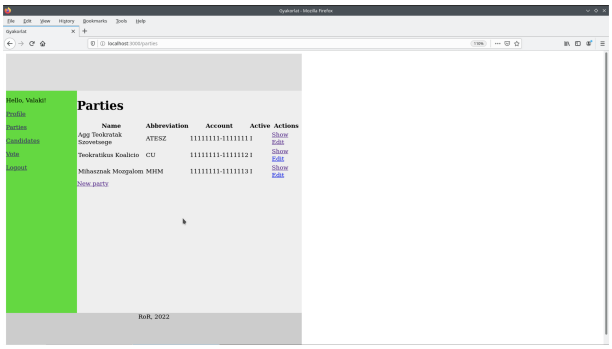
```
class PartiesController < ApplicationController
  before_action :set_party, only: %i[ show edit update
      destroy ]

  # GET /parties or /parties.json
  def index
    #@parties = Party.all
    @p1 = Party.new name: "Agg␣Teokratak␣Szovetsege",
        abbreviation: "ATESZ", account: '11111111-1111111',
        active: true, id: 1
    @p2 = Party.new name: "Teokratikus␣Koalicio",
        abbreviation: "CU", account: '11111111-1111112',
        active: true, id: 2
    @p3 = Party.new name: "Mihasznak␣Mozgalom",
        abbreviation: "MHM", account: '11111111-1111113',
        active: true, id: 3
    @parties = [ @p1, @p2, @p3 ]
  end
  private
    # Use callbacks to share common setup or constraints
        between actions.
    def set_party
      #@party = Party.find(params[:id])
      @party = Party.new name: "Agg␣Teokratak␣Szovetsege",
          abbreviation: "ATESZ", account: '11111111-1111111'
          , active: true, id: 1
    end
end
```

A `index` nézetben eredetileg pártokhoz tartozó töredékeket ágyaztunk be, azonok egy `for` ciklussal végigszaladva a kontroller `index` akciójában definiált `@parties` példányváltozó elemein. Most ezt táblázatos nézetre cseréljük le a táblázat oszlopaiban megjelenítve a párt nevét, rövidítését, bankszámlaszámát, és azt, hogy működik-e.

```
<h1>Parties</h1>

<table id="parties">
  <thead>
    <th>Name</th>
    <th>Abbreviation</th>
    <th>Account</th>
    <th>Active</th>
```

```
    <th>Actions</th>....
  </thead>
  <tbody>
  <% @parties.each do |party| %>
    <%#= render party %>
    <tr>
      <td><%= party.name %></td>
      <td><%= party.abbreviation %></td>
      <td><%= party.account %></td>
      <td><%= party.active ? "I" : "N" %></td>
      <td>
      <%= link_to "Show", party %>
      <%= link_to "Edit", edit_party_path(party.id) %>
      </td>
    </tr>
  <% end %>
  </tbody>
</table>
```

A pártok listájának nézetét a 6. ábra mutatja.



6. ábra. A pártok listájának nézete

Hasonlóan járunk el a jelöltekkel is. Először létrehozzuk a jelöltek modelljék, és REST felületét. A jelöltek nevét és párthovatartozását ismerjük, az utóbbi a `references` típussal jelöljük, amely azt jelenti, hogy a `party` attribútum minden esetben egy `Party` típusú objektumot jelöl. Emellett létrejön egy `party_id` nevű attribútum is, amely `Party` típusú objektumok `id` attribútumának értékét veheti fel.

```
kovacsg@debian:~/gyakorlat/app/views/parties$ rails g
   scaffold candidates name:string party:references
```

```
[WARNING] The model name 'candidates' was recognized as a
    plural, using the singular 'candidate' instead. Override
     with --force-plural or setup custom inflection rules
    for this noun before running the generator.
      invoke  active_record
      create    db/migrate/20220329112120_create_candidates
          .rb
      create    app/models/candidate.rb
      invoke    test_unit
      create      test/models/candidate_test.rb
      create      test/fixtures/candidates.yml
      invoke  resource_route
       route    resources :candidates
      invoke  scaffold_controller
      create    app/controllers/candidates_controller.rb
      invoke    erb
      create      app/views/candidates
      create      app/views/candidates/index.html.erb
      create      app/views/candidates/edit.html.erb
      create      app/views/candidates/show.html.erb
      create      app/views/candidates/new.html.erb
      create      app/views/candidates/_form.html.erb
      create      app/views/candidates/_candidate.html.erb
      invoke    resource_route
      invoke    test_unit
      create      test/controllers/
          candidates_controller_test.rb
      create      test/system/candidates_test.rb
      invoke    helper
      create      app/helpers/candidates_helper.rb
      invoke      test_unit
      invoke    jbuilder
      create      app/views/candidates/index.json.jbuilder
      create      app/views/candidates/show.json.jbuilder
      create      app/views/candidates/_candidate.json.
          jbuilder
kovacsg@debian:~/gyakorlat/app/views/parties$
kovacsg@debian:~/gyakorlat/app/views$ rails db:migrate
== 20220329112120 CreateCandidates: migrating
   ==============================================
-- create_table(:candidates)
   -> 0.0254s
== 20220329112120 CreateCandidates: migrated (0.0258s)
```

```
=========================
```

Hozzunk létre egy jelöltet, és közben nézzük meg a referencia, vagyis az idegen kulcs működését (16., 17., 19. és 20. sorok), valamint az adatbázisbeli rekordok különböző keresési módjait. A `where` függvény a modell példányának attribútumaira illeszt mintát a paraméterként megadott hash objektum alapján (22. sor). Ez minden esetben egy felsorolás típust, pontosabban egy tömböt ad vissza, amely tartalmazza a mintára illeszkedő rekordokhoz tartozó objektumokat. Ha a tömbből egy objektumra vagyunk kíváncsiak, akkor azt a `last` (23. sor), `first` (26. sor) vagy a `take` (25. sor) metódusokkal tudjuk kivenni. Ha az eredményhalmazból több objektumra van szükségünk, akkor azt ezeknek a függvényeknek átadott egész típusú paraméterrel adjatjuk meg, a 24. sorban az utolsó két rekordot kérdezzük le. Ha csak egy, a mintára illeszkedő objektumra van szükségünk, akkor használhatjuk a `find_by` függvényt (27. sor).

```
irb(main):015:0> c = Candidate.new
=> #<Candidate:0x0000561d484efca8 id: nil, name: nil,
    party_id: nil, created_at: nil, updated_at: nil>
irb(main):016:0> c.party_id = 1
=> 1
irb(main):017:0> c.party
  Party Load (5.1ms)  SELECT `parties`.* FROM `parties`
    WHERE `parties`.`id` = 1 LIMIT 1
=>
#<Party:0x00007f02508195d0
 id: 1,
 name: "Agg␣Teokratak␣Szovetsege",
 abbreviation: "ATESZ",
 account: "11111111-11111112",
 active: true,
 created_at: Tue, 22 Mar 2022 12:44:32.448586000 UTC
    +00:00,
 updated_at: Tue, 22 Mar 2022 12:44:32.448586000 UTC
    +00:00>
irb(main):018:0> c.name = 'Claire␣Underwood'
=> "Claire␣Underwood"
irb(main):019:0> c.party = Party.find(2)
  Party Load (0.3ms)  SELECT `parties`.* FROM `parties`
    WHERE `parties`.`id` = 2 LIMIT 1
=>
#<Party:0x0000561d46b90988
...
```

```
irb(main):020:0> c.party
=>
#<Party:0x0000561d46b90988
 id: 2,
 name: "Teokratikus Koalicio",
 abbreviation: "CU",
 account: "11111111-11111113",
 active: true,
 created_at: Tue, 29 Mar 2022 11:10:11.712572000 UTC
     +00:00,
 updated_at: Tue, 29 Mar 2022 11:10:11.712572000 UTC
     +00:00>
irb(main):021:0> c.save
  TRANSACTION (0.1ms)  BEGIN
  Candidate Create (4.9ms)  INSERT INTO `candidates` (`name
      `, `party_id`, `created_at`, `updated_at`) VALUES ('
      Claire Underwood', 2, '2022-03-31 07:12:19.069156', '
      2022-03-31 07:12:19.069156')
  TRANSACTION (1.9ms)  COMMIT
=> true
irb(main):022:0> Candidate.where(party_id:1)
  Candidate Load (3.1ms)  SELECT `candidates`.* FROM `
      candidates` WHERE `candidates`.`party_id` = 1
=> []
irb(main):023:0> Candidate.where(party_id:2).last
  Candidate Load (0.4ms)  SELECT `candidates`.* FROM `
      candidates` WHERE `candidates`.`party_id` = 2 ORDER BY
       `candidates`.`id` DESC LIMIT 1
=> #<Candidate:0x0000561d485a8708 id: 2, name: "Claire
    Underwood", party_id: 2, created_at: Thu, 31 Mar 2022
    07:12:19.069156000 UTC +00:00, updated_at: Thu, 31 Mar
    2022 07:12:19.069156000 UTC +00:00>
irb(main):024:0> Candidate.where(party_id:2).last(2)
  Candidate Load (0.3ms)  SELECT `candidates`.* FROM `
      candidates` WHERE `candidates`.`party_id` = 2 ORDER BY
       `candidates`.`id` DESC LIMIT 2
=>
[#<Candidate:0x00007f02503d1748 id: 1, name: "Claire
    Underwood", party_id: 2, created_at: Tue, 29 Mar 2022
    11:40:34.752439000 UTC +00:00, updated_at: Tue, 29 Mar
    2022 11:40:34.752439000 UTC +00:00>,
 #<Candidate:0x00007f02503d1810 id: 2, name: "Claire
     Underwood", party_id: 2, created_at: Thu, 31 Mar 2022
```

```
    07:12:19.069156000 UTC +00:00, updated_at: Thu, 31 Mar
    2022 07:12:19.069156000 UTC +00:00>]
irb(main):025:0> Candidate.where(party_id:2).take
  Candidate Load (0.3ms)  SELECT `candidates`.* FROM `
    candidates` WHERE `candidates`.`party_id` = 2 LIMIT 1
=> #<Candidate:0x0000561d48621d10 id: 1, name: "Claire
    Underwood", party_id: 2, created_at: Tue, 29 Mar 2022
    11:40:34.752439000 UTC +00:00, updated_at: Tue, 29 Mar
    2022 11:40:34.752439000 UTC +00:00>
irb(main):026:0> Candidate.where(party_id:2).first
  Candidate Load (0.4ms)  SELECT `candidates`.* FROM `
    candidates` WHERE `candidates`.`party_id` = 2 ORDER BY
     `candidates`.`id` ASC LIMIT 1
=> #<Candidate:0x00007f02509ca640 id: 1, name: "Claire
    Underwood", party_id: 2, created_at: Tue, 29 Mar 2022
    11:40:34.752439000 UTC +00:00, updated_at: Tue, 29 Mar
    2022 11:40:34.752439000 UTC +00:00>
irb(main):027:0> Candidate.find_by party_id: 2
  Candidate Load (0.4ms)  SELECT `candidates`.* FROM `
    candidates` WHERE `candidates`.`party_id` = 2 LIMIT 1
=> #<Candidate:0x00007f0250082f10 id: 1, name: "Claire
    Underwood", party_id: 2, created_at: Tue, 29 Mar 2022
    11:40:34.752439000 UTC +00:00, updated_at: Tue, 29 Mar
    2022 11:40:34.752439000 UTC +00:00>
```

A kontrollerben ismételten lecseréljük az adatbázis-műveleteket statikus tartalomra.

```
class CandidatesController < ApplicationController
  before_action :set_candidate, only: %i[ show edit update
     destroy ]

  # GET /candidates or /candidates.json
  def index
    #@candidates = Candidate.all
    @c1 = Candidate.new name: 'I. Szent Viktor', party_id:
       1, id: 1
    @c2 = Candidate.new name: 'Frank Underwood', party_id:
       2, id: 1
    @c3 = Candidate.new name: 'Claire Underwood', party_id:
        2, id: 1
    @c4 = Candidate.new name: 'Bajszos Laszlo', party_id:
       1, id: 1
    @c5 = Candidate.new name: 'Kuruc Elod', party_id: 3, id
```

```
        : 1
    @c6 = Candidate.new name: 'Kuruc Dora', party_id: 3, id
        : 1
    @candidates = [@c1, @c2, @c3, @c4, @c5, @c6]
  end
  private
    # Use callbacks to share common setup or constraints
      between actions.
    def set_candidate
      #@candidate = Candidate.find(params[:id])
      @candite = Candidate.new name: 'I. Szent Viktor',
          party_id: 1, id: 1
    end
end
```