HTML és Rails
Gyakorlat

Kovács Gábor

2022. október 11.

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. Kétféle felhasználóval számolunk egyelőre egy vendégfelhasználóval, és egy bejelentkezett felhasználóval. Az admin felhasználó képernyői bár létezni fognak, úgy teszünk, mintha nem léteznének, mert a bejelentkezett felhasználó nem férhet hozzá.

Először a vendégfelhasználó képernyőinek elrendezését hozzuk létre. Neki három képernyője van: a bejelenkezés, a regisztráció és az elfejeltett jelszó. Ezek egy-egy formot tartalmaznak, melyeket az oldal közepére rendezünk. Hozzunk létre egy új fájl a layoutok között, és egy a bejelentkezést megjelenítő képernyőt:

```
kovacsg@debian:~/pluto> touch app/views/layouts/guest.html.
    erb
kovacsg@debian:~/pluto> rails g controller home login
      create  app/controllers/home_controller.rb
       route  get 'home/login'
      invoke  erb
      create    app/views/home
      create    app/views/home/login.html.erb
      invoke  test_unit
      create    test/controllers/home_controller_test.rb
      invoke  helper
      create    app/helpers/home_helper.rb
      invoke    test_unit
```

A `layouts/guest.html.erb` fájlt az `layouts/application.html.erb` alapján hozzuk létre, a `<body>` elemben térnek el. A törzs tartalmazzon egy

`div`-et, amely a `guest_body` osztályba tartozik.

```
<body>
  <div class="guest_body">
      <%= yield %>
  </div>
</body>
```

A `home/login.html.erb` tartalma a `yield` helyére kerül, és egy formot tartalmaz, amely egy `loginbox` osztályú `div`-ben tartalmazzon egy Pluto-kódra 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 6, illetve 18 karakter hosszúra korlátozzuk. A be nem jelentkezett felhasználónak tegyük lehetővé regisztrációt, valamint az elfelejtett jelszó visszaszerzését, ezeket egy-egy link hozzáadásával tesszük meg.

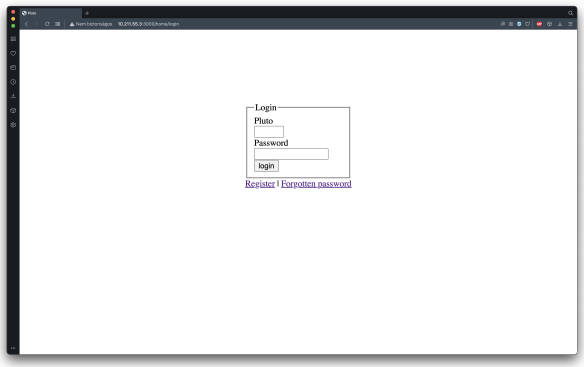
```
<div class="loginbox">
<fieldset>
    <legend>Login</legend>
    <%= form_tag '/session/create', method: :post do %>
        <%= label_tag :pluto %><br>
        <%= text_field_tag :pluto, '', size: 6 %><br>
        <%= label_tag :password %><br>
        <%= text_field_tag :password, '', size: 18 %><br>
        <%= submit_tag :login %>
    <% end %>
</fieldset>
<%= link_to "Register", '/users/new' %> | <%= link_to "
   Forgotten password", '/users/forgotten' %>
</div>
```

A középre rendezést a stíluslapok között tesszük meg, az `app/assets/stylesheets/application.css` fájlban, ahol a `guest_body` és a `loginbox` osztályokhoz rendelünk formázási direktívákat.

```
.guest_body {
    position: relative;
    height: 400px;
}
.guest_body div.loginbox {
```

```
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}
```

A vendégfelhasználók számára kialakított elrendezést például az 1. ábra mutatja.



1. ábra. A vendégfelhasználó képernyők elrendezésének kialakítása

A bejelentkezett felhasználó számára 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 `class`-t rendelve.

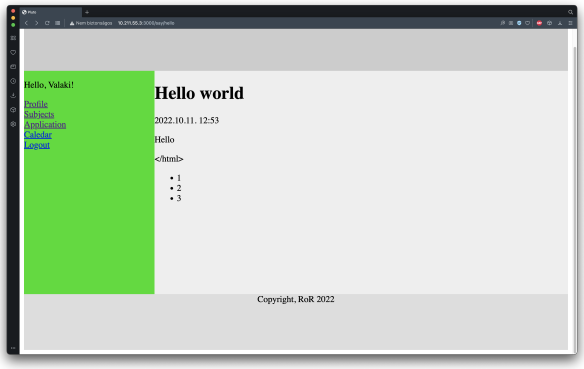
```
<body>
  <div class="body">
      <div class="header"></div>
      <div class="main">
          <div class="menu">
          </div>
          <div class="content">
              <%= yield %>
          </div>
      </div>
      <div class="footer">
          Copyright, RoR 2022
      </div>
  </div>
</body>
```

Következő lépésként készítsük el az oldal stíluslapját, amivel ezek a helyükre kerülnek, és helyezzünk el benne minimális mennyiségű formázási információt. Az oldal legyen 600 pixel magas. A fejrész legyen világosszürke és 100 képpont magas, és helyezzünk el benn egy banner képet. Az oldal központi része legyen 400 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.body {
    height: 600px;
}
div.header {
    height: 100px;
    background-color: #cccccc;
}
div.footer {
    height: 100px;
    background-color: #dddddd;
    text-align: center;
    clear: both;
}
div.main {
    height: 400px;
}
div.menu {
    float: left;
    width: 24%;
    height: 100%;
    background-color: #0d0;
}
div.content {
    float: left;
    width: 76% !important;
    height: 100%;
    background-color: #eee;
  }
```

Az így kialakított elrendezést például az 2. ábra mutatja.

A bejelentkezett felhasználó menüjét a menü részbe tesszük töredékként,



2. ábra. Az oldal elrendezésének kialakítása

amelyet a `render` függvénnyel teszünk meg. Ez azt feltételezi, hogy létezik egy `_menu.html.erb` nevű fájl a `layouts` könyvtárban.

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

Mivel nincs ilyen fájlunk, hozzuk létre a bejelentkezett felhasználó menüjét tartalmazó fájt.

```
kovacsg@debian:~/pluto> touch app/views/layouts/_menu.html.
    erb
```

A bejelentkezett hallgató felhasználó menüjében öt akciót definiálunk: a profiloldal megtekintését, a tárgyak listáját, a kurzusra jelentkezést, az órarendez, valamint a kijelentkezést.

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

<%= link_to "Profile", '/users/edit'%><br/>
<%= link_to "Subjects", '/subjects' %><br/>
<%= link_to "Application", '/subjects' %><br/>
<%= link_to "Caledar", '/calendar' %><br />
<%= link_to "Logout", '/sessions/destroy'%>
```

A menü tetején üdvözöljük a felhasználót a nevével. Mivel minden kontroller esetén szükségünk van erre, a bejelentkezett felhasználó nevének előkeresését az `ApplicationController` kontrollerben tesszük meg. Definiálunk egy privát függvény, amely minden kontroller akció előtt lefut, és ott beállítjuk a `@user` példányváltozót, és így a felhasználó nevét.

```
class ApplicationController < ActionController::Base
    before_action :find_user
    private
        def find_user
            @user = User.new
            @user.name = 'Valaki'
        end
end
```

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:~/pluto> rails g controller users new edit
   forgotten
      create  app/controllers/users_controller.rb
       route  get 'users/new'
              get 'users/edit'
              get 'users/forgotten'
      invoke  erb
      create    app/views/users
      create    app/views/users/new.html.erb
      create    app/views/users/edit.html.erb
      create    app/views/users/forgotten.html.erb
      invoke  test_unit
      create    test/controllers/users_controller_test.rb
      invoke  helper
      create    app/helpers/users_helper.rb
      invoke    test_unit
```

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, továbbá átugorhat az elfejeltett jelszó oldara is.

```
<div class="loginbox">
    <fieldset>
        <legend>Register a new user</legend>
        <%= form_for @user, url: { action: :create },
            method: :post do |form| %>
            <div>
                <%= form.label :name %>
                <%= form.text_field :name, size: 20 %>
            </div>
            <div>
                <%= form.label :email %>
                <%= form.text_field :email, size: 30 %>
            </div>
            <div>
                <%= form.label :password %>
                <%= form.password_field :password, size: 20
                    %>
            </div>
            <div>
                <%= form.label :password_confiration %>
                <%= form.password_field :
                    password_confirmation, size: 20 %>
            </div>
            <%= form.submit 'Register' %>
```

```
        <% end %>
    </fielset>
    <%= link_to "Back", :back %> | <%= link_to "Forgotten␣
        password", '/users/forgotten' %>
</div>
```

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.

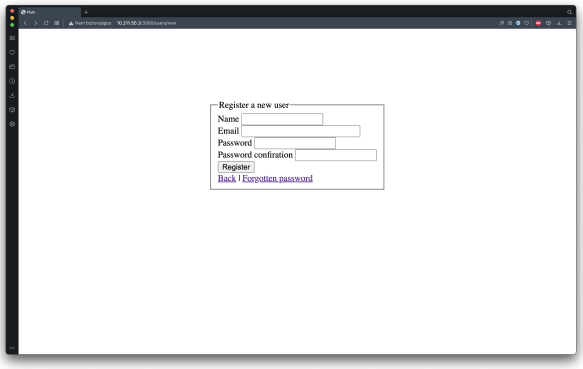
```
Rails.application.routes.draw do
  get 'users/new'
  post 'users/create'
  get 'users/edit'
  put 'users/update'
  get 'users/forgotten'
  post 'users/send_forgotten'
end
```

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

```
class UsersController < ApplicationController
  layout 'guest', only: [:new, :forgotten]
  def new
    @user = User.new
  end
end
```

A regisztráció és az elfelejtett jelszó akciók a vendégfelhasználókra vonatkoznak, ezért csakis ezekre vonatkozóan beállítjuk a `guest` elrendezésts. A felhasználói regisztráció nézetét a 3. ábra mutatja.

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.



3. ábra. A regisztráció nézete

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. A különbség annyi, hogy a felhasználó a létrehozás után már rendelkezik Pluto-kóddal, amely viszont nem módosítható, ezért a 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.

```
<div class="loginbox">
    <fieldset>
        <legend>User profile</legend>
        <%= form_for @user, url: { action: :update },
            method: :put do |form| %>
            <div>
                <%= form.label :pluto %>
                <%= form.password_field :pluto, size: 6,
                    disabled: true %>
            </div>
            <div>
                <%= form.label :name %>
                <%= form.text_field :name, size: 20 %>
            </div>
            <div>
                <%= form.label :email %>
```

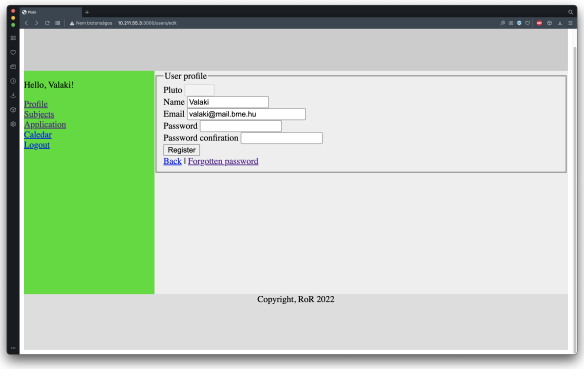
```
                <%= form.text_field :email, size: 30 %>
            </div>
            <div>
                <%= form.label :password %>
                <%= form.password_field :password, size: 20
                    %>
            </div>
            <div>
                <%= form.label :password_confiration %>
                <%= form.password_field :
                    password_confirmation, size: 20 %>
            </div>
            <%= form.submit 'Register' %>
        <% end %>
    </fielset>
    <%= link_to "Back", :back %> | <%= link_to "Forgotten␣
        password", '/users/forgotten' %>
</div>
```

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, amely már megtörtént a `ApplicationController`-ben, viszont néhány attribútum értéke még hiányzik. Ezeket egyelőre nem az adatbázisból tesszük meg, hanem statikus tartalommal töltjük 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.

```
class UsersController < ApplicationController
  layout 'guest', only: [:new, :forgotten]
  def edit
    @user.pluto = 'AAAAAA'
    @user.email = 'valaki@mail.bme.hu'
  end

  def forgotten
  end
end
```

A felhasználói profiloldal szerkesztésének nézetét az 4. ábra mutatja. Itt már a bejelentkezett felhasználó elrendezését látjuk. Láthatjuk továbbá, hogy a Rails automatikusan inicializálta a form mezőit, ahol a hozzájuk tartozó érték elérhető volt – a jelszó mezők kivételével.



4. ábra. A profiloldal nézete

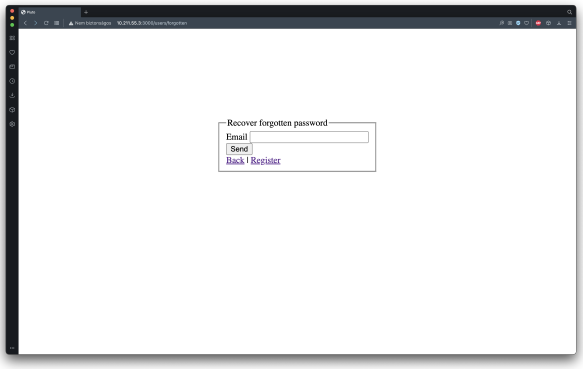
Ezután alakítsuk ki az elfejetett jelszó oldalt is. Itt egyszerűbb a formunk a beléptetésnél, csak az email címet tartalmazza.

```
<div class="loginbox">
    <fieldset>
        <legend>Recover forgotten password</legend>
        <%= form_for @user, url: { action: :send_forgotten
            }, method: :post do |form| %>
            <div>
                <%= form.label :email %>
                <%= form.text_field :email, size: 30 %>
            </div>
            <%= form.submit 'Send' %>
        <% end %>
    </fieldset>
    <%= link_to "Back", :back %> | <%= link_to "Register",
        '/users/new' %>
</div>
```

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.

A bejelentkezett felhasználó menüjében tovább haladva a tárgyak listáját látjuk. A tárgyakat a felhasználó nem szerkesztheti közvetlenül, csakis az adminisztrátor. Hozzunk létre egyetlen paranccsal REST interfésszel rendelkező, a tárgyak adatait karbantartani képes képernyőket hozzájuk tartozó kontroller akciókkal, és a tárgyak adatmodelljét.



5. ábra. Az elfelejtett jelszó nézete

A tárgyak rendelkezzenek névvel, előadóval, Pluto-kóddal és kreditszámmal.

```
kovacsg@debian:~/pluto> rails g scaffold subject name:
    string lecturer:string pluto:string credit:integer{1}
      invoke  active_record
      create    db/migrate/20221011111456_create_subjects.
          rb
      create    app/models/subject.rb
      invoke    test_unit
      create      test/models/subject_test.rb
      create      test/fixtures/subjects.yml
      invoke  resource_route
       route    resources :subjects
      invoke  scaffold_controller
      create    app/controllers/subjects_controller.rb
      invoke    erb
      create      app/views/subjects
      create      app/views/subjects/index.html.erb
      create      app/views/subjects/edit.html.erb
      create      app/views/subjects/show.html.erb
      create      app/views/subjects/new.html.erb
      create      app/views/subjects/_form.html.erb
      create      app/views/subjects/_subject.html.erb
      invoke    resource_route
      invoke    test_unit
      create      test/controllers/subjects_controller_test
          .rb
      create      test/system/subjects_test.rb
```

```
      invoke    helper
      create      app/helpers/subjects_helper.rb
      invoke      test_unit
      invoke    jbuilder
      create      app/views/subjects/index.json.jbuilder
      create      app/views/subjects/show.json.jbuilder
      create      app/views/subjects/_subject.json.jbuilder
```

Mivel modellt is létrehoztunk ezzel a paranccsal, létrejött egy migráció is. Hajtsuk végre, majd nézzük meg, mit generáltunk automatikusan.

```
kovacsg@debian:~/pluto> rails db:migrate
== 20221011111456 CreateSubjects: migrating
   ===================================
-- create_table(:subjects)
   -> 0.0258s
== 20221011111456 CreateSubjects: migrated (0.0260s)
   ==========================
```

A webfelületen a `/subjects` útvonalat megnyitva látjuk a tárgyak listáját, amely jelenleg üres. Az új tárgy létrehozása link egy űrlapot nyit meg, amellyel új tárgyat hozhatunk létre a beviteli mezők kitöltésével. A mentés gombra kattintás után a tárgy adatlapjára kerülünk, ahol a tárgyat az `id` attribútumával azonosítjuk. Innen átmehetünk a tárgy adatainak szerkesztése képernyőre, amely pontosan ugyanaz a form, mint a tárgy létrehozása form azzal a különbséggel, hogy itt a tárgy példányának az `id` attribútuma már beállított, és a beviteli mezőkben megjelennek a beállított értékek. A lista oldalra visszamenve láthatjuk a már létrehozott tárgyak listáját.

Konzolon, és adatbázis-konzolon is meggyőződhetünk arról, hogy a felületen összekattintot tárgyak valóban létrejöttek.

```
kovacsg@debian:~/pluto> rails c
Loading development environment (Rails 7.0.4)
irb(main):004:0> Subject.all
  Subject Load (0.3ms)  SELECT `subjects`.* FROM `subjects`
=>
[#<Subject:0x00007f59d828d708
  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:0x00007f59d828d618
  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):005:0>
kovacsg@debian:~/pluto$ rails db

MariaDB [pluto_development]> select * from subjects;
+----+--------------+--------------+----------+--------+
| id | name         | lecturer     | pluto    | credit |
  created_at                 | updated_at
  |
+----+--------------+--------------+----------+--------+

|  1 | Testnevelés  | Futó Béla    | BMETT001 |      0 |
  2022-10-11 11:20:18.121945 | 2022-10-11 11:20:18.121945
  |
|  2 | Fizika       | Orosz László | BMEFT001 |      5 |
  2022-10-11 11:21:29.181438 | 2022-10-11 11:21:29.181438
  |
+----+--------------+--------------+----------+--------+

2 rows in set (0.000 sec)
```

A tárgyaknak szemeszterenként vannak kurzusai, ezért a tárgyak mintájára létrehozzuk előbb a szemeszterek modellt, majd a kurzusok modellt. Kezdjük a szemeszterekkel. Egy szemeszernek van neve, éve, amely négy digiten ábrázolható egész szám, és évszaka, amelyet egy digites egész számmal ábrázolunk. Hajtsuk egyből végre a migrációt.

```
kovacsg@debian:~/pluto> rails g scaffold semester name:
   string year:integer{4} season:integer{1}
      invoke  active_record
      create    db/migrate/20221011112331_create_semesters.
         rb
```

```
      create    app/models/semester.rb
      invoke    test_unit
      create      test/models/semester_test.rb
      create      test/fixtures/semesters.yml
      invoke  resource_route
       route    resources :semesters
      invoke  scaffold_controller
      create    app/controllers/semesters_controller.rb
      invoke    erb
      create      app/views/semesters
      create      app/views/semesters/index.html.erb
      create      app/views/semesters/edit.html.erb
      create      app/views/semesters/show.html.erb
      create      app/views/semesters/new.html.erb
      create      app/views/semesters/_form.html.erb
      create      app/views/semesters/_semester.html.erb
      invoke    resource_route
      invoke    test_unit
      create      test/controllers/
         semesters_controller_test.rb
      create      test/system/semesters_test.rb
      invoke    helper
      create      app/helpers/semesters_helper.rb
      invoke      test_unit
      invoke    jbuilder
      create      app/views/semesters/index.json.jbuilder
      create      app/views/semesters/show.json.jbuilder
      create      app/views/semesters/_semester.json.
         jbuilder
kovacsg@debian:~/pluto> rails db:migrate
== 20221011112331 CreateSemesters: migrating
   =============================================
-- create_table(:semesters)
   -> 0.0082s
== 20221011112331 CreateSemesters: migrated (0.0085s)
   ====================================
```

Módosítjuk a modell osztályunkat, hogy az évszakot a 0, 1 számok helyett a `spring` és a `fall` karaktersorozatokkal állíthassuk. Ezt `enum` deklarációjával tesszük meg, azt mondjuk, hogy a `season` attibútum egész értéke mint index jelölje ki a megadott tömb megfelelő értékét.

```
class Semester < ApplicationRecord
    enum :season, [:spring, :fall]
```

```
end
```

A felhasználói felületet is érinti a módosítás, a `app/views/semester/_form.html.erb` fájlban a season beviteli mezőt számválasztóról legördülő menüre cseréljük. A második paraméter egy kétdimenziós tömb, ahol a belső tömbök első eleme a legördülő menüben megjelenő szöveg, a második eleme pedig annak az értéke, amely majd az season attribútumhoz rendelődik az opció kiválasztása esetén.

```
  <div>
    <%= form.label :season, style: "display: block" %>
    <%= form.select :season, [['Spring', 'spring'], ['Fall
        ', 'fall']] %>
  </div>
```

Most ne a webfelületen, hanem konzolon hozzunk létre két szemesztert, hogy lássuk az enumot működés közben. Az 1. sorban létrehozunk egy új szemesztert, a 2. sorban beállítjuk, hogy ez egy tavaszi szemeszter, és egyből elmentjük az adatbázisba, erről a 3. sorban meggyőződünk. A 4-5. sorban beállítjuk az évet és a szemeszter nevét, a 6. sorban módosítjuk a rekord adatait. A 7-10. sorban létrehozunk egy másik példányt. A 11. sorban megnézzük az adatbázisunkban lévő szemeszter példányokat.

```
kovacsg@debian:~/pluto> rails c
Loading development environment (Rails 7.0.4)
irb(main):001:0> s = Semester.new
=> #<Semester:0x00005583710b03c0 id: nil, name: nil, year:
    nil, season: nil, created_at: nil, updated_at...
irb(main):002:0> s.spring!
  TRANSACTION (0.1ms)  BEGIN
  Semester Create (2.7ms)  INSERT INTO `semesters` (`name`,
       `year`, `season`, `created_at`, `updated_at`) VALUES
      (NULL, NULL, 0, '2022-10-11 11:25:45.696363',
      '2022-10-11 11:25:45.696363')
  TRANSACTION (0.6ms)  COMMIT
=> true
irb(main):003:0> s
=>
#<Semester:0x00005583710b03c0
 id: 1,
 name: nil,
 year: nil,
 season: "spring",
 created_at: Tue, 11 Oct 2022 11:25:45.696363000 UTC
```

```
    +00:00,
 updated_at: Tue, 11 Oct 2022 11:25:45.696363000 UTC +00:00
    >
irb(main):004:0> s.year = 2023
=> 2023
irb(main):005:0> s.name = '2022/2023 spring'
=> "2022/2023 spring"
irb(main):006:0> s.save
  TRANSACTION (0.1ms)  BEGIN
  Semester Update (6.2ms)  UPDATE `semesters` SET `
      semesters`.`name` = '2022/2023 spring', `semesters`.`
      year` = 2023, `semesters`.`updated_at` = '2022-10-11
      11:26:14.456019' WHERE `semesters`.`id` = 1
  TRANSACTION (0.5ms)  COMMIT
=> true
irb(main):007:0> s = Semester.new
=> #<Semester:0x00005583715feb60 id: nil, name: nil, year:
    nil, season: nil, created_at: nil, updated_at...
irb(main):008:0> s.year = 2022
=> 2022
irb(main):009:0> s.name = '2022/2023 fall'
=> "2022/2023 fall"
irb(main):010:0> s.fall!
  TRANSACTION (0.1ms)  BEGIN
  Semester Create (0.6ms)  INSERT INTO `semesters` (`name`,
       `year`, `season`, `created_at`, `updated_at`) VALUES
      ('2022/2023 fall', 2022, 1, '2022-10-11
      11:26:45.752992', '2022-10-11 11:26:45.752992')
  TRANSACTION (1.9ms)  COMMIT
=> true
irb(main):011:0> Semester.all
  Semester Load (0.3ms)  SELECT `semesters`.* FROM `
      semesters`
=>
[#<Semester:0x00005583716915f0
  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>,
```

```
#<Semester:0x0000558371691500
  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>]
```

A kurzus egy tárgy egy szemeszterben induló példánya, ezért a kurzusnak hivatkozia kell egy-egy tárgy, illetve szemeszter példányra. Ezen kívül a kurzusnak van típusa, amely lehet előadás, gyakorlat vagy laboratórium, ezt `t` névvel nevezzük, mert a `type` név foglalt. A kurzusnak két alkalma, ahol mindkettőnek van időpontja és helye. Az időpont dátum típusú, amiből csak a hét napja, az óra és a perc mezőket fogjuk majd használni. A kurzusnak ezen kívül van egy létszámkorlátja is, amely egy három digiten ábrázolható egész szám.

```
kovacsg@debian:~/pluto> rails g scaffold course subject:
   references semester:references t:integer{1} date1:
   datetime date2:datetime location1:string location2:
   string limit:integer{3}
      invoke  active_record
      create    db/migrate/20221011113239_create_courses.rb
      create    app/models/course.rb
      invoke    test_unit
      create      test/models/course_test.rb
      create      test/fixtures/courses.yml
      invoke  resource_route
       route    resources :courses
      invoke  scaffold_controller
      create    app/controllers/courses_controller.rb
      invoke    erb
      create      app/views/courses
      create      app/views/courses/index.html.erb
      create      app/views/courses/edit.html.erb
      create      app/views/courses/show.html.erb
      create      app/views/courses/new.html.erb
      create      app/views/courses/_form.html.erb
      create      app/views/courses/_course.html.erb
      invoke    resource_route
      invoke    test_unit
      create      test/controllers/courses_controller_test.
```

```
          rb
       create    test/system/courses_test.rb
       invoke  helper
       create    app/helpers/courses_helper.rb
       invoke    test_unit
       invoke  jbuilder
       create    app/views/courses/index.json.jbuilder
       create    app/views/courses/show.json.jbuilder
       create    app/views/courses/_course.json.jbuilder
kovacsg@debian:~/pluto> rails db:migrate
== 20221011113239 CreateCourses: migrating
   ====================================
-- create_table(:courses)
   -> 0.0169s
== 20221011113239 CreateCourses: migrated (0.0171s)
   ===========================
```

Hasonlóan a szemeszterhez a kurzus típusát is enum felhasználásával oldjuk meg.

```
class Course < ApplicationRecord
  enum :t, [:lecture, :practice, :laboratory]
end
```

A kurzus formját, `app/views/course/_form.hmtl.erb` három helyen is módosítanunk kell. A tárgyakra, illetve a szemeszterekre való hivatkozást, továbbá a kurzus típusának megadását is legördülő menükkel oldjuk meg. Először nézzük meg, hogyan tudunk a legördülő menükhöz az opciókat tartalmazó kétdimenziós tömböt egy adatbázis-művelettel lekérdezni.

```
kovacsg@debian:~/pluto> rails c
Loading development environment (Rails 7.0.4)
irb(main):001:0> Subject.all.map do |x| [x.name, x.id] end
  Subject Load (0.2ms)  SELECT `subjects`.* FROM `subjects`
=> [["Testnevelés", 1], ["Fizika", 2]]
```

Ezt felhasználva módosítjuk a formunkat.

```
  <div>
    <%= form.label :subject, style: "display:␣block" %>
    <%= form.select :subject_id, options_for_select(Subject
        .all.map do |x| [x.name, x.id] end) %>
  </div>

  <div>
```

```
    <%= form.label :semester, style: "display: block" %>
    <%= form.select :semester_id, options_for_select(
        Semester.all.map do |x| [x.name, x.id] end) %>
  </div>

  <div>
    <%= form.label :t, style: "display: block" %>
    <%= form.select :t, [['Lecture', 'lecture'], ['Practice
        ', 'practice'], ['Laboratoty', 'laboratory']] %>
  </div>
```