# HTML és Rails
# Gyakorlat

Kovács Gábor

2021. március 16.

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">
     <% if logged_in? %>
       <%= render 'layouts/user' %>
     <% else %>
       <%= render 'layouts/guest' %>
     <% end %>
   </div>
   <div id="content">
     <%= yield %>
   </div>
 </div>
 <div id="footer"> RoR, 2021</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. Az oldal központi része legyen 600 pixel magas. A

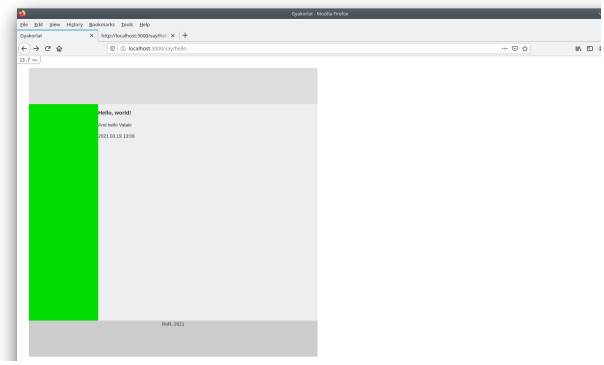
menüsávot a központi részben, a fejléc alatt helyezzük el, és az vízszintesen a szélesség 24%-át foglalja el, magasságát a központi rész magassága definiálja. Az oldal tartalmi része világosszürke háttérrel rendelkezzen, és a menütől jobbra helyezkedjen el a vízszintesen a szélesség 76%-át elfoglalva. 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, egyfelhasználóra, az utóbbi korábban keresztülment 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



1. ábra. Az oldal elrendezésének kialakítása

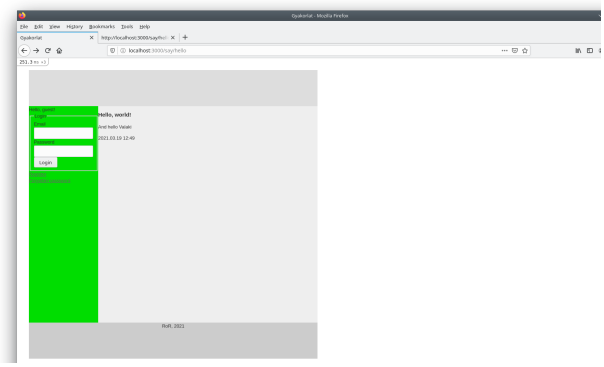
tartalmazó fájl alkalmazás szinten kezeljük, ezért a `layouts` könyvtárban helyezzük el, így a be nem jelentkezett felhasználó bármelyik oldalon bejelentkezhet. A Rails konvenció szerint a részlegesen renderelt állományok neve aláhúzásjellel kezdődik. Legyen a fájlunk neve ezért `_guest.html.erb`! A 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 18 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 '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/user' %>
</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.

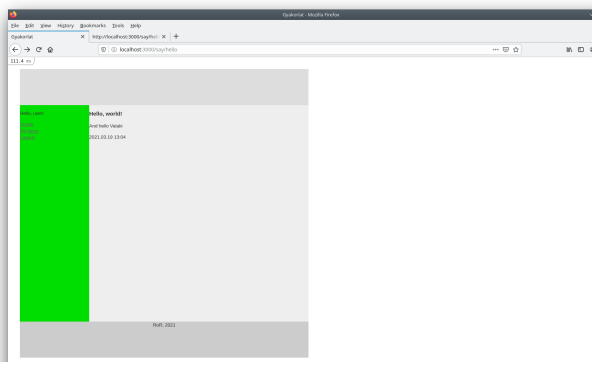
```
<div id="menu">
  <% if logged_in? %>
    <%#= render '/layouts/user' %>
```

```
  <% else %>
    <%#= render '/layouts/guest' %>
  <% end %>
</div>
```

A bejelentkezett felhasználó menüjét a vendéghez hasonlóan beágyazott nézettel hozzuk létre (`_user.html.erb`). Egyelőre három akciót definiálunk: a profiloldal megtekintését, az összes projekt megtkintését, valamint a kijelentkezést.

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

<%= link_to "Profile", '/users/edit' %><br/>
<%= link_to "My items", '/items' %><br/>
<%= link_to "Logout", '/sessions/destroy' %><br/>
```

A bejelentkezett 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/layouts$ rails g
    controller users new edit show forgotten
Running via Spring preloader in process 4585
      create  app/controllers/users_controller.rb
       route  get 'users/new'
get 'users/edit'
```

```
get 'users/show'
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/show.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
      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`, a felhasználó adatlapját megmutató `show`, 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 az öt elem rendre a következő: egy 20 karakter széles, a felhasználó nevére 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, 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.

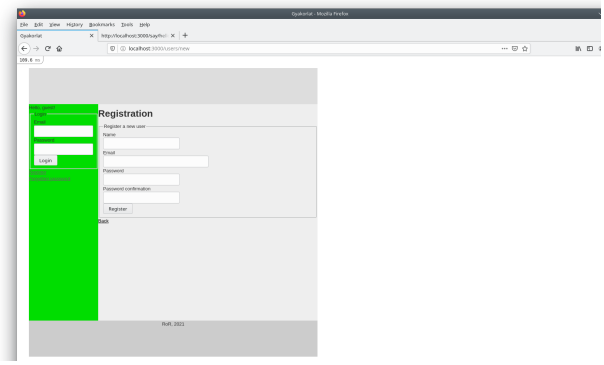
```
<h1>Registration</h1>

<fieldset>
  <legend>Register a new user</legend>
  <%= form_for @user, url: '/users/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_confirmation %>
      <%= form.password_field :password_confirmation, size:
          20 %>
    </div>
    <%= form.submit "Register" %>
  <% end %>
</fieldset>

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

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.

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-




4. ábra. A regisztráció nézete

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 formok eseménykezelőihez a generate controller parancsunk nem generált útvonalakat, ezért azokat felvesszük a `config/routes.rb`-be. Jelenleg a nézetek kialakításához szingleton példányokkal dolgozunk, vagyis egyetlen felhasználó adatait vagyunk képesek megjeleníteni. Később szükségünk lesz egy `:id` szimbólumra, ami képessé tesz minket a felhasználók megkülönböztetésére, és azonos nevű paraméterként jelentkezik majd a kontrollerben.

```
get 'users/new'
post 'users/create'
get 'users/edit'
put 'users/update'
get 'users/show'
get 'users/forgotten'
post 'users/send_forgotten'

get 'say/hello'
```

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 kontrol-

ler 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: '/users/update', method: :put 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_confirmation %>
      <%= form.password_field :password_confirmation, size:
          20 %>
    </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,

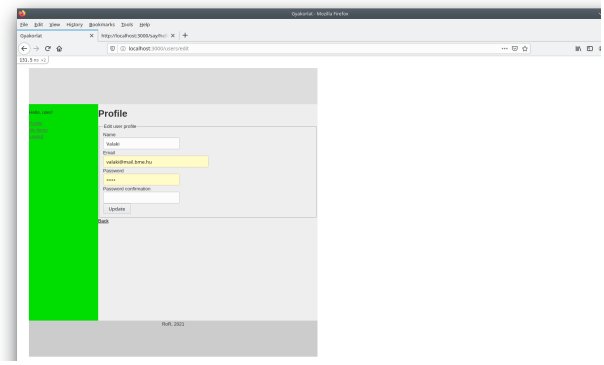
a `index` akcióban pedig a `@users` példányváltozót, amely egy User típusú objektumokat tartalmazó tömb.

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

  def create
  end

  def edit
    @user = User.new name: 'Valaki', email: 'valaki@mail.
       bme.hu', password: ''
  end

  def update
  end

  def show
    @user = User.new name: 'Valaki', email: 'valaki@mail.
       bme.hu', password: ''
  end

  def forgotten
  end

  def send_forgotten
  end
end
```

A felhasználói profiloldal szerkesztésének nézetét az 5. 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 to recover
     password</legend>
  <%= form_tag '/users/send_forgotten', method: :post do %>
    <div>
      <%= label_tag :email, "Email" %>
```

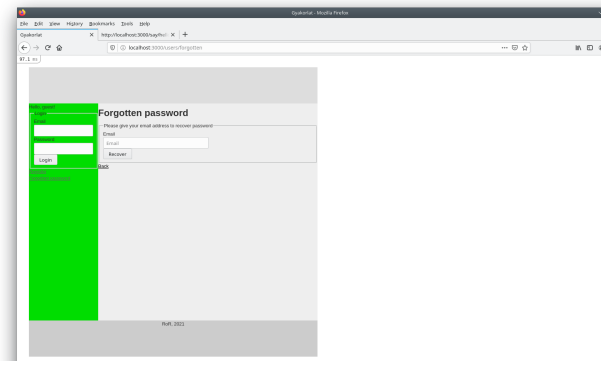



5. ábra. A profiloldal nézete

```
        <%= text_field :email, '', placeholder: 'Email', size
            : 30 %>
      </div>
      <%= submit_tag "Recover" %>
    <% 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 6. ábra mutatja.




6. ábra. Az elfelejtett jelszó nézete

Hozzuk létre az erőforrásmegosztó megosztó portálunk projektjeihez (Item

néven az előző gyakorlaton létrehozott scaffold) tartozó erőforrásokra (fájlok és metaadataik) vonatkozó modellünket és a hozzá tartozó kontrollert egy paranccsal. A `Resource` modellünkben legyen két string típusú, `name`, illetve `version` nevű, és egy a tulajdonos projektre vonatkozó referencia. Az erőforrások rendelkezni fog magát az fájt tartalmazó állomány modellel, ezt a következő gyakorlatokon rendeljük majd hozzá.

```
kovacsg@debian:~/gyakorlat$ rails g scaffold resource name:
   string item:references version:string
Running via Spring preloader in process 9489
      invoke  active_record
      create    db/migrate/20210316121736_create_resources.
         rb
      create    app/models/resource.rb
      invoke    test_unit
      create      test/models/resource_test.rb
      create      test/fixtures/resources.yml
      invoke  resource_route
       route    resources :resources
      invoke  scaffold_controller
      create    app/controllers/resources_controller.rb
      invoke    erb
      create      app/views/resources
      create      app/views/resources/index.html.erb
      create      app/views/resources/edit.html.erb
      create      app/views/resources/show.html.erb
      create      app/views/resources/new.html.erb
      create      app/views/resources/_form.html.erb
      invoke    resource_route
      invoke    test_unit
      create      test/controllers/
         resources_controller_test.rb
      create      test/system/resources_test.rb
      invoke    helper
      create      app/helpers/resources_helper.rb
      invoke      test_unit
      invoke    jbuilder
      create      app/views/resources/index.json.jbuilder
      create      app/views/resources/show.json.jbuilder
      create      app/views/resources/_resource.json.
         jbuilder
      invoke  assets
      invoke    scss
```

```
      create      app/assets/stylesheets/resources.scss
      invoke  scss
    identical    app/assets/stylesheets/scaffolds.scss
```

Létrejött egy `Resource` modell, egy `ResourcesController` kontroller és a kapcsolódó nézetek: `new` és `edit` egy-egy nézet, amelyek a közös `_form` töredékben lévő formot használják az erőforrás adatainak létrehozására, illetve módosítására. A `show` nézet az erőforrás adatlapját mutatja, a `index` nézet pedig az elérhető erőforrásokat mutatja egy táblázatban.

Észrevehetjük, hogy az erőforrás létrehozása és szerkesztése hasonló művelek, így egy formmal kezelhető. A scaffolddal létrehozott form rendelkezik egy felhasználóbarát hibamegjelenítő résszel, amely nem megfelelő adat megadása esetén kiemeli a hibás beviteli mezőket a Rails beépített stílusával (`app/assets/stylesheets/scaffolds.css`).

Hajtsuk végre a scaffold létrehozása során keletkezett migrációt.

```
kovacsg@debian:~/gyakorlat/config$ rails db:migrate
== 20210316121736 CreateResources: migrating
   ==========================================
-- create_table(:resources)
   -> 0.0182s
== 20210316121736 CreateResources: migrated (0.0185s)
   =================================
```

Ezután a böngészőben nyissuk meg a projektek (items) nézetet (`http://localhost:3000/items`), próbáljuk ki a projekt létrehozását, törlését, listázását. Nézzük meg, hogy létrejött-e a rekord az adatbázisban, nyissuk meg az adatbázis konzolt:

```
kovacsg@debian:~/gyakorlat/db/migrate# rails db
MariaDB [gyakorlat_development]> desc items;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| name        | varchar(255) | YES  |     | NULL    |                |
| description | text         | YES  |     | NULL    |                |
```

```
| created_at   | datetime(6)   | NO   |      | NULL     |
                |
| updated_at   | datetime(6)   | NO   |      | NULL     |
                |
+--------------+---------------+------+------+----------+-----------------+

5 rows in set (0.008 sec)


MariaDB [gyakorlat_development]> select * from items;
+----+------------+-------------+---------------------------+---------------------------+
| id | name       | description | created_at
                  | updated_at                |
+----+------------+-------------+---------------------------+---------------------------+
|  1 | My project | My project  | 2021-03-16
  12:14:18.818988 | 2021-03-16 12:14:18.818988 |
+----+------------+-------------+---------------------------+---------------------------+

1 row in set (0.000 sec)
```

Mivel a Rails adatbáziskezelését még nem ismerjük, az `ItemsController`-ben cseréljük le az automatikusan generált adatbázisműveleteket tartalmazó kódrészleteket statikus adatokra. Ezt két helyen kell megtennünk, az `index` akcióban, ahol az `@items` példányváltozó projekt példányokat tartalmazó tömbjét hozzuk létre, illetve a `before_action` helper függvény által kijelölt `set_item` azonosítójú metódusban, amely lefut mind a `show`, mind az `edit` nézet betöltésekor, itt csak az `@item` példányváltozóhoz kell egy projekt példányt rendelnünk. Egy projekthez több erőforrás is tartozhat, tároljuk azokat most a `resources` tömbben.

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

  # GET /items or /items.json
  def index
    #@items = Item.all
    @item1 = Item.new name: "Project 1", description: "
       Project 1", id: 1
    @item2 = Item.new name: "Project 2", description: "
       Project 2", id: 2
    @items = []
```

```
    @items << @item1
    @items << @item2
  end

  private
    def set_item
      #@item = Item.find(params[:id])
      @item = Item.new name: "Project 1", description: "
          Project 1", id: 1
      @resources = []
      @resource1 = Resource.new name: 'File1', item: @item,
          version: '1.0', id: 1
      @resource2 = Resource.new name: 'File2', item: @item,
          version: '1.0', id: 2

      @resources << @resource1
      @resources << @resource2
    end
end
```

A projekthez rendelt erőforrások megjeleníthetősége végett a `show` akció számára a kontrollerben inicializáltuk az erőforrások példányváltozóját, azokat meg tudjuk jeleníteni. A projekt adatlapján, ahol egy felsorolásba linkként beágyazzuk a kapcsolódó erőforrásokat.

```
<ul>
  <% for resource in @resources do %>
    <li>
    <%= link_to "#{resource.name} #{resource.version}", "/
        resources/"+resource.id.to_s %>
    </li>
  <% end %>
</ul>
```

A link az erőforrás adatlapjára visz, ahol feltölthető az erőforráshoz tartozó állomány. Ezt a későbbiekben, a feltöltés megvalósításakor még módosítani fogjuk, valamint létrehozzuk hozzá az eseménykezelőt.

```
<fieldset>
  <legend>Upload resource</legend>
  <%= form_tag '/resource/upload', method: :post do %>
    <%= label_tag 'File' %>
    <%= file_field_tag :attachment %>
    <%= submit_tag 'Upload' %>
  <% end %>
```

```
</fieldset>
```