Rails MVC, modell, session
Gyakorlat

Kovács Gábor

2017. október 24.

Az előző gyakorlaton megkezdett példát folytatjuk a felhasználói sessionök kezelésének megvalósításával, illetve az adatmodell kialakításával. Az előző alkalommak két modellt hoztunk létre, a felhasználók `User` nevű modelljét és a sportszerek `Item` nevű modelljét. A felhasználók modellje a következőképp néz ki az adatbázisban.

```
MariaDB [gyakorlat_development]> desc users;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| email      | varchar(255) | YES  |     | NULL    |                |
| password   | varchar(255) | YES  |     | NULL    |                |
| name       | varchar(255) | YES  |     | NULL    |                |
| phone      | varchar(255) | YES  |     | NULL    |                |
| created_at | datetime     | NO   |     | NULL    |                |
| updated_at | datetime     | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)

MariaDB [gyakorlat_development]> select * from users;
+----+-----------------+----------+---------+--------------+---------------------+---------------------+
| id | email           | password | name    | phone        | created_at          | updated_at          |
+----+-----------------+----------+---------+--------------+---------------------+---------------------+
|  1 | valaki@mail.bme.hu | titok    | Vala Ki | +3611234567 | 2017-09-26 11:12:08 | 2017-09-26 11:12:08 |
+----+-----------------+----------+---------+--------------+---------------------+---------------------+
1 row in set (0.00 sec)
```

Először növeljük portálunk biztonságát azzal, hogy a jelszavakat nem szövegesen, hanem titkosítva tároljuk. Ez a jelszó mező átnevezéséből és egy új, a titkosítás során a felhasználó számára egyedi attribútum felvételéből áll. A Rails az új attribútum felvételéhez képes automatikusan invertálható migrációt generálni jól definiált migrációnév esetén. A hozzáadott attribútumnak

ilyen esetben a migráció neve után kell szerepelnie. Azonban az attribútum átnevezését magunknak kell majd hozzáadnunk a migrációhoz.

```
kovacsg@debian:~/gyakorlat/db/migrate> rails g migration AddSaltToUser salt:
    string
      invoke  active_record
      create    db/migrate/20171024101520_add_salt_to_user.rb
```

Mivel az attribútum átnevezése nem invertálható, vagy a `change` metódusban használjuk a `reversible` függvényt, vagy a `change` helyett két függvényt definiálunk `up`, illetve `down` néven. Most ez utóbbit választjuk. Az automatikusan generált invertálható műveletről el kell feledkeznünk, magunknak kell kettéválasztanunk a műveletet. Mivel az adatbázisban már van adat, fel irányú migráció esetén gondoskodunk kell azok új attribútumainak inicializálásáról.

```
class AddSaltToUser < ActiveRecord::Migration[5.1]
  def up
    add_column :users, :salt, :string
    rename_column :users, :password, :encrypted_password
  end

  def down
    remove_column :users, :salt
    rename_column :users, :encrypted_password, :password
  end

#  def change
#    reversible do |dir|
#      dir.up { }
#      dir.down { }
#    end
#  end
end
```

Hajtsuk végre a migrációt érvényre juttatandó az adatmodellünk változásait!

```
kovacsg@debian:~/gyakorlat/db/migrate> rails db:migrate
== 20171024101520 AddSaltToUser: migrating ====================================
-- add_column(:users, :salt, :string)
   -> 0.0353s
-- rename_column(:users, :password, :encrypted_password)
   -> 0.0073s
== 20171024101520 AddSaltToUser: migrated (0.0428s) ===========================
```

A `new_record?` függvény azt állapítja meg egy ActiveRecord objektumról, hogy azt elmentettük-e már az adatbázisba, ezt az `id` attribútum ellenőrzésével végzi el, ami alapértelmezés szerint `nil`. A memóriában létrehozott `ActiveRercord` objektumok az első mentés műveletig új rekordnak számítanak, az `id` attribútumuk csak annak hatására inicializálódik.

```
kovacsg@debian:~/gyakorlat/app/controllers> rails c
```

```
Loading development environment (Rails 5.0.1)
irb(main):001:0> User.new
=> #<User id: nil, name: nil, encrypted_password: nil, email: nil,
    created_at: nil, updated_at: nil, salt: nil>
irb(main):002:0> u = User.new
=> #<User id: nil, name: nil, encrypted_password: nil, email: nil,
    created_at: nil, updated_at: nil, salt: nil>
irb(main):003:0> u.new_record?
=> true
```

```
kovacsg@debian:~/gyakorlat/db/migrate> rails db
MariaDB [gyakorlat_development]> desc users;
+-----------------------+--------------+------+-----+---------+----------------+
| Field                 | Type         | Null | Key | Default | Extra          |
+-----------------------+--------------+------+-----+---------+----------------+
| id                    | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| email                 | varchar(255) | YES  |     | NULL    |                |
| encrypted_password    | varchar(255) | YES  |     | NULL    |                |
| name                  | varchar(255) | YES  |     | NULL    |                |
| phone                 | varchar(255) | YES  |     | NULL    |                |
| created_at            | datetime     | NO   |     | NULL    |                |
| updated_at            | datetime     | NO   |     | NULL    |                |
| salt                  | varchar(255) | YES  |     | NULL    |                |
+-----------------------+--------------+------+-----+---------+----------------+

8 rows in set (0.00 sec)
```

Nézzük meg, hogy milyen elemi műveletekkel férhetünk egy vagy több adatbázisbeli rekordhoz hozzá Rails-ből. A `first` (1. sor), `last` (2. sor) és `take` (3. sor) rendre az elsődleges kulcs szerinti első, utolsó, és az alapértelmezett rendezés szerinti első rekordot adják vissza. Ezeknek egész paramétert adva a rendezés után visszaadott rekordok számát adhatjuk meg (4. sor), a visszatérési érték `ActiveRecord::Base` objektumról `ActiveRecord::Relation` objektumra változik, ami tömbként működik, és kaszkádosan végrehajthatók rá a keresési műveletek. Az `all` függvény az modell összes rekordját visszaadja egy tömbben.

```
kovacsg@debian:~/gyakorlat/app/models> rails c
Loading development environment (Rails 5.1.4)
irb(main):001:0> User.first
   (0.2ms)  SET NAMES utf8,  @@SESSION.sql_mode = CONCAT(CONCAT(@@sql_mode,
       ',STRICT_ALL_TABLES'), ',NO_AUTO_VALUE_ON_ZERO'),  @@SESSION.
       sql_auto_is_null = 0, @@SESSION.wait_timeout = 2147483
  User Load (0.2ms)  SELECT  `users`.* FROM `users` ORDER BY `users`.`id`
      ASC LIMIT 1
```

```
=> #<User id: 1, email: "valaki@mail.bme.hu", encrypted_password: "titok",
    name: "Vala Ki", phone: "+3611234567", created_at: "2017-09-26
    11:12:08", updated_at: "2017-09-26 11:12:08", salt: nil>
irb(main):002:0> User.last
  User Load (0.4ms)  SELECT  `users`.* FROM `users` ORDER BY `users`.`id`
     DESC LIMIT 1
=> #<User id: 1, email: "valaki@mail.bme.hu", encrypted_password: "titok",
    name: "Vala Ki", phone: "+3611234567", created_at: "2017-09-26
    11:12:08", updated_at: "2017-09-26 11:12:08", salt: nil>
irb(main):003:0> User.take
  User Load (0.8ms)  SELECT  `users`.* FROM `users` LIMIT 1
=> #<User id: 1, email: "valaki@mail.bme.hu", encrypted_password: "titok",
    name: "Vala Ki", phone: "+3611234567", created_at: "2017-09-26
    11:12:08", updated_at: "2017-09-26 11:12:08", salt: nil>
irb(main):004:0> User.take(2)
  User Load (0.3ms)  SELECT  `users`.* FROM `users` LIMIT 2
=> [#<User id: 1, email: "valaki@mail.bme.hu", encrypted_password: "titok",
    name: "Vala Ki", phone: "+3611234567", created_at: "2017-09-26
    11:12:08", updated_at: "2017-09-26 11:12:08", salt: nil>]
irb(main):006:0> User.take(2)[0]
  User Load (0.3ms)  SELECT  `users`.* FROM `users` LIMIT 2
=> #<User id: 1, email: "valaki@mail.bme.hu", encrypted_password: "titok",
    name: "Vala Ki", phone: "+3611234567", created_at: "2017-09-26
    11:12:08", updated_at: "2017-09-26 11:12:08", salt: nil>
```

Elsődleges kulcs alapján, amely jellemzően a weboldalon megjelenő azonosító önmaga, a `find` metódussal kereshetünk egy objektumot. Valamely attribútum értéke alapján való keresést többféleképpen végezhetünk. Ezek a metódusok minden esetben egy Ruby tömböt adnak vissza, amely azonban lehet akár üres is, vagy tartalmazhat egyetlen objektumot is. A Rails 4-es verziójától a `where` keresőmetódus (8-9. sor) a javasolt. Ezzel nem csak paraméter egyezésére kereshetünk, hanem egy string paraméterrel általánosabb feltételt is megadhatunk, például egy attribútum értéke és egy szám összehasonlítását. A keresőmetódusok tetszőleges számú alkalommal egymás után láncolhatók, a `where` által visszaadott tömbből jellemzően a `take` metódussal vesszük ki a keresés eredményét, ha tudjuk, hogy pontosan egy lesz.

```
irb(main):007:0> User.find 1
  User Load (0.4ms)  SELECT  `users`.* FROM `users` WHERE `users`.`id` = 1
     LIMIT 1
=> #<User id: 1, email: "valaki@mail.bme.hu", encrypted_password: "titok",
    name: "Vala Ki", phone: "+3611234567", created_at: "2017-09-26
    11:12:08", updated_at: "2017-09-26 11:12:08", salt: nil>
irb(main):008:0> User.where(email: 'valaki@mail.bme.hu')
  User Load (0.4ms)  SELECT  `users`.* FROM `users` WHERE `users`.`email` =
     'valaki@mail.bme.hu' LIMIT 11
=> #<ActiveRecord::Relation [#<User id: 1, email: "valaki@mail.bme.hu",
    encrypted_password: "titok", name: "Vala Ki", phone: "+3611234567",
    created_at: "2017-09-26 11:12:08", updated_at: "2017-09-26 11:12:08",
    salt: nil>]>
irb(main):009:0> User.where(email: 'valaki@mail.bme.hu').take
  User Load (0.7ms)  SELECT  `users`.* FROM `users` WHERE `users`.`email` =
     'valaki@mail.bme.hu' LIMIT 1
=> #<User id: 1, email: "valaki@mail.bme.hu", encrypted_password: "titok",
    name: "Vala Ki", phone: "+3611234567", created_at: "2017-09-26
    11:12:08", updated_at: "2017-09-26 11:12:08", salt: nil>
```

```
irb(main):010:0> u = User.new email:'vki@mail.bme.hu', password: 'titok',
    name: 'V Ki', phone: '+3612345678'
=> #<User id: nil, email: "vki@mail.bme.hu", encrypted_password: nil, name:
    "V Ki", phone: "+3612345678", created_at: nil, updated_at: nil, salt:
    nil>
irb(main):011:0> u.password
=> "titok"
irb(main):012:0> u.save
   (0.2ms)  BEGIN
  SQL (0.7ms)  INSERT INTO `users` (`email`, `encrypted_password`, `name`, `
      phone`, `created_at`, `updated_at`, `salt`) VALUES ('vki@mail.bme.hu',
       '749f9e65a50b2a160d3b8a73742bc7ca3ccc6234', 'V Ki', '+3612345678', '
      2017-10-24 10:30:40', '2017-10-24 10:30:40', '58
      f31f69d2c529f0533f9d325d697004')
   (7.6ms)  COMMIT
=> true
irb(main):013:0> u
=> #<User id: 2, email: "vki@mail.bme.hu", encrypted_password: "749
    f9e65a50b2a160d3b8a73742bc7ca3ccc6234", name: "V Ki", phone:
    "+3612345678", created_at: "2017-10-24 10:30:40", updated_at:
    "2017-10-24 10:30:40", salt: "58f31f69d2c529f0533f9d325d697004">
irb(main):014:0>
```

Következő lépésként tegyük rendbe a felhasználói session kezelését, ami a `menu` akcióinak megvalósítását, a jelszó titkosítását, és a titkosított jelszó adatbázisban való eltárolását jelenti első körben. Másodszor pedig a felhasználó regisztráció folyamatát érinti.

Nézzük először meg, milyen egyéb módunk van hash kulcsként használható hexadecimális vagy Base64 string előállítására Railsben (3-6. sor).

```
kovacsg@debian:~/gyakorlat/app/models> rails c
irb(main):001:0> SecureRandom.hex(16)
=> "a64cb6f8be1d6669527d484fe51c5250"
irb(main):002:0> SecureRandom.base64(16)
=> "W8bNp1W9d82mL9LK5v6bww=="
irb(main):003:0> Digest::SHA1.hexdigest('lalala')
=> "df2efa060e335f97628ca39c9fef5469ab3cb837"
irb(main):004:0> Digest::SHA1.hexdigest('lalala')
=> "df2efa060e335f97628ca39c9fef5469ab3cb837"
```

A migrációban minden egyes felhasználói rekordhoz hozzáadtunk egy egyéni a jelszó titkosításához használt random kulcs tárololására használt attribútumot és egy a típust tároló attribútumot, továbbá a jelszó attribútumot pedig átnevezzük, így az titkosítatlanul nem kerülhet bele az adatbázisba.

A jelszó attribútumot átneveztük, viszont a felületen továbbra is használjuk, ezért csak a modell osztályra korlátozva elérhetővé újra tesszük.

```
class User < ApplicationRecord
  attr_accessor :password
end
```

Bejelentkezéskor a megadott jelszót már az adatbázisban található titkosított jelszóval kell összevetnünk, ezért a `User` modell példányának mentésekor (regisztráció során vagy a felhasználói jelszó módosításakor) a `password`

példányváltozót `encrypted_password` attribútummá kell transzformálnunk. Ezt a következőképp tesszük meg. Definiálunk egy `encrypt` azonosítójú osztálymetódust, amely a `password` és `salt` példányváltozók alapján egy hash függvénnyel egy kódolt karaktersorozatot hoz létre. Definiálunk továbbá egy `encrypt_password` azonosítójú metódust, amely az összes nem üres jelszó esetére elvégzi a titkosítást, illetve új, még el nem mentett rekord esetén inicializálja a `salt` attribútum értékét egy véletlen számmal. Végül a `before_save` metódussal jelezzük, hogy a `save` metódus minden egyes meghívása előtt hívódjék meg a `encrypt_password` metódus.

```
class User < ApplicationRecord
  before_save :encrypt_password

  def self.encrypt(pass, salt)
    Digest::SHA1.hexdigest(pass+salt)
  end

  def encrypt_password
    return if password.blank?
    if new_record?
      self.salt = SecureRandom.hex(16)
    end
    self.encrypted_password = User.encrypt self.password, self.salt
  end
end
```

Ezután rátérhetünk a felhasználói session megvalósítására. Ehhez létrehozzuk a `sessions` kontrollert, amelynek `create` és `destroy` metódusai léptetik be, illetve ki a felhasználót.

```
kovacsg@debian:~/gyakorlat/app/helpers> rails g controller sessions create
    destroy
      create  app/controllers/sessions_controller.rb
       route  get 'sessions/destroy'
       route  get 'sessions/create'
      invoke  erb
      create    app/views/sessions
      create    app/views/sessions/create.html.erb
      create    app/views/sessions/destroy.html.erb
      invoke  test_unit
      create    test/controllers/sessions_controller_test.rb
      invoke  helper
      create    app/helpers/sessions_helper.rb
      invoke    test_unit
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/sessions.coffee
      invoke    scss
      create      app/assets/stylesheets/sessions.scss
```

A létrehozott `create` és `destroy` nézetekre nincs szükségünk, azokat töröljük. A `sessions` kontrollerhez ezek után nem tartozik nézet, a `login`, illetve a `logout` link eseményeit kezeli le. Ellenőrizzük, hogy a menüben, vagyis a `layouts/_menu.html.erb`-ben a form akciója a `/sessions/create`-re

mutat-e, illetve a belépett felhasználó menüjében a `Logout` link a `/sessions/destroy`-ra mutat-e. Belépéskor, illetve kilépéskor, megpróbálunk az aktuális oldalon maradni. A `routes.rb` konfigurációs fájlhoz adjuk hozzá a `post 'sessions/ create'` és a `get 'sessions/ destroy'`, ugyanis bejelentkezéskor HTTP POST üzenetben adatokat is küldünk a szerver felé, kijelentkezéskor pedig HTTP GET üzenet elég.

A következő lépés a felhasználó hitelesítésének megvalósítása, amit a `User` modellben teszünk meg egy osztálymetódussal. A hitelesítés két argumentummal rendelkezik egy felhasználói email címmel és egy jelszóval, és a sikeresen hitelesített felhasználó objektumával vagy `nil`-lel tér vissza. Először megkeresi a rekordok között a felhasználó azonosítójának megfelelő rekordot, majd elvégzi a hitelesítést. Bármelyik sikertelensége esetén a visszatérési érték `nil`. A hitelesítés (`authenticated?` metódus) azt ellenőrzi, hogy a titkosított jelszó attribútum megegyezik-e a jelszó titkosítása által visszaadott értékkel.

```
class User < ApplicationRecord
  def self.authenticate(email, pass)
    #user = find_by_email email
    user = User.where(email: email).take
    user && user.authenticated?(pass) ? user : nil
  end

  def authenticated?(pass)
    self.encrypted_password == User.encrypt(pass, self.salt)
  end
end
```

A kontrollerünk ezek után a következőképp néz ki. A hitelesítés imént megírt metódusának visszatérési értékét a `user` kontroller példányváltozóhoz rendeljük. Az email címet és a jelszót a `params` hash-ből vesszük ki a `menu`-ben megadott név alapján. A `params` hash alábbi használata veszélyes lehet, éles rendszerben ne használjuk közvetlenül! A SQL injection támadásokat elkerülendő az aposztrófokat escape-elnünk kell!

Ha a hitelesített felhasználó értéke nem `nil`, akkor a `session` hash `:user` szimbólummal hivatkozott értékének beállítjuk a felhasználó `id` attribútumának értékét, majd visszairányítjuk a felhasználót az előző oldalra. Ellenkező esetben egy hibaüzenetet küldünk a következő oldalnak a `flash` hash-en keresztül, és ugyancsak visszairányítjuk a felhasználót az előző oldalra. Kilépéskor töröljük a `session` hash tartalmát, és egy `flash` üzenettel visszairányítjuk a felhasználót az előző oldalra.

```
class SessionsController < ApplicationController
  def create
    @user = User.authenticate params[:email], params[:password]
    if @user
      session[:user] = @user.id
    else
```

```
      flash[:notice] = 'Invalid email address or password'
    end
    redirect_back fallback_location: hello_path
  end

  def destroy
    reset_session
    flash[:notice] = "Logged out successfully"
    redirect_back fallback_location: hello_path
  end
end
```

A flash hashen keresztül értéket adhatunk át a következő HTTP kérésre adott válasz számára. A megjelenítendő üzenet helye legyen az központi nézetben és a `_menu.html.erb`-ben.

```
<%= flash[:notice] %><br />
```

Ezek után már el tudjuk dönteni, hogy egy felhasználó mikor van bejelentkezve az előző alkalommal írt alkalmazás szintű helperben. Ha a `session :user` szimbólumhoz tartozó értéke nem üres, akkor a felhasználó be van jelentkezve.

```
module ApplicationHelper
  def logged_in?
    session[:user]
  end
end
```

Felhasználó létrehozásához és adatainak módosításához szükséges nézeteket már létrehoztuk, valósítsuk meg a formokat kezelő kontroller akciókat. A regisztrációhoz a `users` kontroller `create` akciója tartozik, a profil módosításához pedig az `update` akció tartozik. Takarítsuk ki az előző gyakorlaton bedrótozott értékeket a kontrollerből! A rekordok biztonsága végett a HTTP kérés paramétereinek lehetséges kulcsait korlátozzunk, és az alapján hozunk létre új felhasználót. Az ellenőrzést a `user_params` privát metódus végzi el.

```
class UsersController < ApplicationController
  def create
    @user = User.new user_params
    if @user.save
      session[:user] = @user.id
      flash[:notice] = 'Successful registration'
      redirect_back fallback_location: hello_path
    else
      flash[:notice] = @user.errors.messages
      redirect_back fallback_location: hello_path
    end
  end

  def update
    @user.update user_params
    redirect_back
  end
```

```
  private
    def user_params
      params.require(:user).permit(:name, :email, :phone, :password, :
          password_confirmation)
    end
```

Több felhasználó számára is elérhetővé szeretnénk tenni a portálunkat, ez lehetséges, mert a felhasználók adatait most már az adatbázisból vesszük elő. Ha egy konkrét felhasználó adait szeretnénk megnézni, szerkeszteni vagy módosítani, akkor a HTTP kérés paramétereként át kell adnunk a felhasználó adatbázisbeli azonosítóját is, hogy a megfelelő felhasználó adatai jelenlenek meg, módosuljanak. Az első módosítás a HTTP kérés paraméterezhetővé tétele, amit a `routes.rb` konfiguráció állományban teszünk meg – magyarázat a következő előadáson.

```
  post 'sessions/create', to: 'sessions#create', as: 'login'
  match 'sessions/destroy', to: 'sessions#destroy', as: 'logout', via: [:get
      , :delete]

  match 'users/new', to: 'users#new', via: :get, as: 'register'
  post 'users/create'

  get 'users/show/:id', to: 'users#show', as: 'profile'

  get 'users/edit/:id', to: 'users#edit', as: 'edit_profile'
  put 'users/update/:id', to: 'users#update', as: 'update_profile'

  get 'users/forgotten'
  post 'users/send_forgotten'
  get 'say/hello', to: 'say#hello', as: 'hello'
```

A felhasználó `id` attribútumának értéke, akárcsak az HTTP kérés összes egyéb paramétere bekerül a `params` hash-be, ahonnan kikereshetjük, ha szükségünk van rá. Ha épp nem új felhasználót hozunk létre, akkor ezt meg kell tennünk. Bejelentkezett felhasználó esetén ugyanerre a célra felhasználhatjuk a sessionbel tárolt értéket it. Mivel több függvény előtt ugyanazt a keresést kellene elvégeznünk, egy privát callback függvényt definiálunk a `before_filter` segítségével, ami minden kontroller esetén lefut, mert azt a kontrollerek ősosztályában, az `ApplicationController`-ben definiáljuk. A keresést csak akkor végezzük el, ha van aktív felhasználói session.

```
class ApplicationController < ActionController::Base
  before_action :find_user

  private
    def find_user
      @user = User.find session[:user] if session[:user]
    end
end
```

A regisztrációkor az elmentendő felhasználót még az elmentés előtt validáljuk, az emailcím attribútumnak nemüresnek kell lennie (`:presence`), és

egyedinek (`:uniqueness`) kell lennie , és ha ez nem teljesül, akkor visszajelzünk, hogy nem megfelelő felhasználónévről van szó. A név mezőt kötelező megadni a regisztráció során, és nem lehet üres. A jelszónak és annak ismétlésének meg kell egyeznie (`:confirmation`), ha az elmentett jelszó nem üres (`password_required?` metódussal vizsgálva). A `confirmation` opció létrehozza a modell objektumban a `_confirmation` szuffixű settert és gettert, így a kontroller hozzá tudja rendelni ahhoz a formból érkező adatokat. Ezeket az ellenőrzéseket a modell osztályban validációs helper metódusokkal tesszük meg.

```
class User < ActiveRecord::Base
  validates :email, {
    presence: true, uniqueness: true
  }
  validates :name, presence: true
  validates :password, confirmation: true, if: :password_required?

  def password_required?
    self.new_record? || !password.blank?
  end
end
```

Konzolon és a webfelületen ellenőrizhetjük, hogy elmenthető-e üres névvel névvel és létező email címmel egy új rekord! Az `ActiveRecord` példányokról a `valid?` metódussal kérdezhetjük meg, hogy átmennek-e az osztályában definiált validációkon. Ha nem, akkor a hibaüzeneteket az `errors` példányváltozóban érhetjük el, amiket kivezethetünk a nézetekre.

```
irb(main):001:0> u = User.new
   (0.2ms)  SET NAMES utf8,  @@SESSION.sql_mode = CONCAT(CONCAT(@@sql_mode,
       ',STRICT_ALL_TABLES'), ',NO_AUTO_VALUE_ON_ZERO'),  @@SESSION.
       sql_auto_is_null = 0, @@SESSION.wait_timeout = 2147483
=> #<User id: nil, email: nil, encrypted_password: nil, name: nil, phone:
    nil, created_at: nil, updated_at: nil, salt: nil>
irb(main):002:0> u.save
   (0.2ms)  BEGIN
  User Exists (0.3ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email`
      IS NULL LIMIT 1
   (0.1ms)  ROLLBACK
=> false
irb(main):003:0> u.errors.messages
=> {:email=>["can't be blank"], :name=>["can't be blank"]}
irb(main):004:0> u = User.new name: 'a', email: 'valaki@mail.bme.hu'
=> #<User id: nil, email: "valaki@mail.bme.hu", encrypted_password: nil,
    name: "a", phone: nil, created_at: nil, updated_at: nil, salt: nil>
irb(main):005:0> u.save
   (0.7ms)  BEGIN
  User Exists (0.4ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email` =
      BINARY 'valaki@mail.bme.hu' LIMIT 1
   (0.1ms)  ROLLBACK
=> false
irb(main):006:0> u.errors.messages
=> {:email=>["has already been taken"]}
```

A felhasználói regisztáció nézetén (`new.html.erb`) a hibaüzeneteket egyszerűen megjeleníthetjük a következő kódrészlet segítségével:

```
<h1>Registration</h1>
<%= flash[:notice] %>

  <% if @user.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@user.errors.count, "error") %> prohibited this user
          from being saved:</h2>

      <ul>
      <% @user.errors.full_messages.each do |message| %>
        <li><%= message %></li>
      <% end %>
      </ul>
    </div>
  <% end %>
```

Próbáljunk meg ezután felhasználót felvenni a webfelületen hibás adatokkal, és nézzük meg a hibaüzeneteket! Láthatjuk, hogy ezek a műveletek nem hajtódnak végre, és a Rails megjelöli a hibás beviteli mezőket. Ezután a validációs üzenet is megváltozik, amit konzolon ellenőrizhetünk.

A hibaüzenetet testreszabhatjuk a lokalizációs beállításokban:

```
  activerecord:
    errors:
      models:
        user:
          attributes:
            mail:
              blank: 'Empty email
              taken: 'Email already used'
```

Az előző gyakorlaton létrehoztuk a sportszerek modellt. A sportszereket felhasználók kölcsönözhetik ki, így e két modell között kapcsolatnak kell fennállnia, a kölcsönzésnek. A kölcsönzés eseményről megjegyezzünk, hogy mikor történt, erre az alapértelmezés szerint generálódó időpecsétet fogjuk használni. Továbbá szükségünk van az eszköz visszaadásának időpontjára, és a visszaadát utáni állapotra. Az állapot egy egy karakteren ábrázolható érték, ami lehet elérhető, kölcsönzött, illetve elveszett. Ezt egy Rails `enum`-mal hozzuk létre. A `references` típusúként deklarált attribútumok esetén az idegen kulcsokhoz automatikusan létrejöttek az ActiveRecord asszociációk. A sportszerek és a felhasználók közötti kapcsolatot a kölcsönzések modellben az egész típusú `user_id` és `item_id` attribútumok adják, amelyek megfelel a Rails alapértelmezett idegen kulcs elnevezési konvenciójának, és így minden módosítás nélkül használhatuk.

```
kovacsg@debian:~/gyakorlat/config> rails g model borrow user:references item
    :references status:integer{1} return_time:datetime
      invoke  active_record
      create    db/migrate/20171024112323_create_borrows.rb
      create    app/models/borrow.rb
```

```
      invoke    test_unit
      create      test/models/borrow_test.rb
      create      test/fixtures/borrows.yml
```

A relációink a következők. Egy felhasználó sok kölcsönzéssel rendelkezhet, amelyek minden esetben egy-egy sportszerre vonatkoznak. Fordítva egy sportszert az idő folyamán sok felhasználó is kölcsönözheti, de egyszerre csak egy kölcsönzésben szerepelhet, ami az utolsó kölcsönzés.

```
class Borrow < ApplicationRecord
  enum status: [:borrowed, :available, :lost]
  belongs_to :user
  belongs_to :item
end
class Item < ApplicationRecord
  has_many :borrows
  has_many :users, through: :borrows
end
class User < ApplicationRecord
  has_many :borrows
  has_many :items, through: :borrows
end
```

Hajtsuk végre a kölcsönzések modell migrációját.

```
kovacsg@debian:~/gyakorlat/db/migrate> rails db:migrate
== 20171024112323 CreateBorrows: migrating
   ===================================
-- create_table(:borrows)
   -> 0.0246s
== 20171024112323 CreateBorrows: migrated (0.0249s)
   ==========================
```

Az enumhoz kapcsolt értéket az enum értékeivel nevesítve kérdezhetjük le és állíthatjuk be.

```
irb(main):001:0> Borrow.statuses
=> {"borrowed"=>0, "available"=>1, "lost"=>2}
irb(main):007:0> b = Borrow.new(user_id:5, item_id:3)
=> #<Borrow id: nil, user_id: 5, item_id: 3, status: nil, return_time: nil,
    created_at: nil, updated_at: nil>
irb(main):008:0> b.borrowed!
   (0.3ms)  BEGIN
  User Load (0.6ms)  SELECT  `users`.* FROM `users` WHERE `users`.`id` = 5
      LIMIT 1
  Item Load (0.1ms)  SELECT  `items`.* FROM `items` WHERE `items`.`id` = 3
      LIMIT 1
  SQL (1.4ms)  INSERT INTO `borrows` (`user_id`, `item_id`, `status`, `
      created_at`, `updated_at`) VALUES (5, 3, 0, '2017-10-27 08:40:08', '
      2017-10-27 08:40:08')
   (8.8ms)  COMMIT
=> true
irb(main):009:0> b.borrowed?
=> true
```

Hozzunk létre konzolon néhány kölcsönzést! Először felhasználókra van szükségünk, vegyünk fel vezeték- és keresztneveket, majd generáljuk felhasználókat ezek random kombinációiból.

```
kovacsg@debian:~/gyakorlat/app> rails c
Loading development environment (Rails 5.1.4)
irb(main):001:0> first_name = ['Adam', 'Bob', 'Cecile', "David"]
irb(main):005:0> last_name = ['Brown', 'Pink', 'Orange', 'Purple']
=> ["Brown", "Pink", "Orange", "Purple"]
irb(main):023:0> for i in 1..10 do
irb(main):024:1* fn = first_name.sample(1)[0]
irb(main):025:1> ln = last_name.sample(1)[0]
irb(main):026:1> u = User.new name: fn+' '+ln , email: fn+'_'+ln+"@mail.bme.
    hu", phone: '+3611111111', password: 'titok', password_confirmation: '
    titok'
irb(main):027:1> u.save
irb(main):028:1> end
   (0.6ms)  SET NAMES utf8 ,   @@SESSION.sql_mode = CONCAT(CONCAT(@@sql_mode,
    ',STRICT_ALL_TABLES'), ',NO_AUTO_VALUE_ON_ZERO'),   @@SESSION.
    sql_auto_is_null = 0, @@SESSION.wait_timeout = 2147483
   (0.1ms)  BEGIN
  User Exists (0.2ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email` =
    BINARY 'Bob_Pink@mail.bme.hu' LIMIT 1
  SQL (0.2ms)  INSERT INTO `users` (`email`, `encrypted_password`, `name`, `
    phone`, `created_at`, `updated_at`, `salt`) VALUES ('Bob_Pink@mail.bme.
    hu', 'e7773864faa1d67a8fbfe506d7372dc29db26f10', 'Bob Pink',
    '+3611111111', '2017-10-24 11:35:17', '2017-10-24 11:35:17', '
    c68bb30e8ece7025bdb65ab7044f9dcf')
   (7.3ms)  COMMIT
   (0.2ms)  BEGIN
  User Exists (0.3ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email` =
    BINARY 'David_Pink@mail.bme.hu' LIMIT 1
  SQL (0.4ms)  INSERT INTO `users` (`email`, `encrypted_password`, `name`, `
    phone`, `created_at`, `updated_at`, `salt`) VALUES ('David_Pink@mail.bme
    .hu', 'accd3875e3face2c1461fa5d26f9dc387a0e974d', 'David Pink',
    '+3611111111', '2017-10-24 11:35:17', '2017-10-24 11:35:17', '
    b2f5a2b2021ac4dbf252cb50a9964d2e')
   (9.2ms)  COMMIT
   (0.2ms)  BEGIN
  User Exists (0.4ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email` =
    BINARY 'Adam_Purple@mail.bme.hu' LIMIT 1
  SQL (0.2ms)  INSERT INTO `users` (`email`, `encrypted_password`, `name`, `
    phone`, `created_at`, `updated_at`, `salt`) VALUES ('Adam_Purple@mail.
    bme.hu', '153e20eb259d7c51a061d34cd5edba299e8f8f32', 'Adam Purple',
    '+3611111111', '2017-10-24 11:35:17', '2017-10-24 11:35:17', '9961
    e5c5120909158ed990d0748d0fd8')
   (1.8ms)  COMMIT
   (0.1ms)  BEGIN
  User Exists (1.0ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email` =
    BINARY 'David_Orange@mail.bme.hu' LIMIT 1
  SQL (0.2ms)  INSERT INTO `users` (`email`, `encrypted_password`, `name`, `
    phone`, `created_at`, `updated_at`, `salt`) VALUES ('David_Orange@mail.
    bme.hu', '7ae505d292fc4797877d4c618f776d7394e5720b', 'David Orange',
    '+3611111111', '2017-10-24 11:35:17', '2017-10-24 11:35:17', '777
    edeebb672f77fd51b70a2aa8b0498')
   (2.1ms)  COMMIT
   (0.1ms)  BEGIN
  User Exists (0.2ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email` =
    BINARY 'Adam_Pink@mail.bme.hu' LIMIT 1
  SQL (0.2ms)  INSERT INTO `users` (`email`, `encrypted_password`, `name`, `
    phone`, `created_at`, `updated_at`, `salt`) VALUES ('Adam_Pink@mail.bme.
    hu', '8dcaa3c079cb81668733eca0bd3fd0c3b8b6cb1a', 'Adam Pink',
    '+3611111111', '2017-10-24 11:35:17', '2017-10-24 11:35:17', '618
    c373b0078d633a88305850cbaf203')
   (1.3ms)  COMMIT
```

```
   (0.1ms)  BEGIN
  User Exists (0.2ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email` =
     BINARY 'David_Purple@mail.bme.hu' LIMIT 1
  SQL (0.1ms)  INSERT INTO `users` (`email`, `encrypted_password`, `name`, `
     phone`, `created_at`, `updated_at`, `salt`) VALUES ('David_Purple@mail.
     bme.hu', 'ce6cf8ffaf6c9db38a2c7272ada8bd308c6e2b90', 'David Purple',
     '+3611111111', '2017-10-24 11:35:17', '2017-10-24 11:35:17', '
     ec0b41d7bd6b32c4d4ab6583edd80097')
   (1.9ms)  COMMIT
   (0.1ms)  BEGIN
  User Exists (0.2ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email` =
     BINARY 'Cecile_Brown@mail.bme.hu' LIMIT 1
  SQL (0.2ms)  INSERT INTO `users` (`email`, `encrypted_password`, `name`, `
     phone`, `created_at`, `updated_at`, `salt`) VALUES ('Cecile_Brown@mail.
     bme.hu', '33a1860e1e37e9353a7ea51b57e4cd20a1a24619', 'Cecile Brown',
     '+3611111111', '2017-10-24 11:35:17', '2017-10-24 11:35:17', '
     f4666dbb4bb3d7be3e8c4ce239774361')
   (1.8ms)  COMMIT
   (0.1ms)  BEGIN
  User Exists (0.5ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email` =
     BINARY 'Adam_Purple@mail.bme.hu' LIMIT 1
   (0.2ms)  ROLLBACK
   (0.1ms)  BEGIN
  User Exists (0.2ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email` =
     BINARY 'Cecile_Pink@mail.bme.hu' LIMIT 1
  SQL (0.1ms)  INSERT INTO `users` (`email`, `encrypted_password`, `name`, `
     phone`, `created_at`, `updated_at`, `salt`) VALUES ('Cecile_Pink@mail.
     bme.hu', '273ab2ae4b5bde2b6eac94ed243c71c13c9dce38', 'Cecile Pink',
     '+3611111111', '2017-10-24 11:35:17', '2017-10-24 11:35:17', '941
     a010d28dce09faa8450118bbb5473')
   (2.0ms)  COMMIT
   (0.1ms)  BEGIN
  User Exists (0.5ms)  SELECT  1 AS one FROM `users` WHERE `users`.`email` =
     BINARY 'Cecile_Pink@mail.bme.hu' LIMIT 1
   (0.1ms)  ROLLBACK
=> 1..10
irb(main):030:0> User.count
   (0.4ms)  SELECT COUNT(*) FROM `users`
=> 11
```

Hasonlóan generáljunk kezdeti adatokat a sportszer raktárkészletünkbe.

```
irb(main):032:0> items = ['Labda', "Teniszuto", "Zsamoly", "Hullahopp", "
    Ugrokotel"]
=> ["Labda", "Teniszuto", "Zsamoly", "Hullahopp", "Ugrokotel"]
irb(main):033:0> for i in 1..100 do
irb(main):034:1* Item.create name: items.sample(1)[0], barcode: Time.now.
    to_i, price: i * 100
irb(main):035:1> end
   (0.7ms)  BEGIN
  SQL (0.5ms)  INSERT INTO `items` (`name`, `barcode`, `price`, `created_at
     `, `updated_at`) VALUES ('Zsamoly', 1508845070, 100, '2017-10-24
     11:37:50', '2017-10-24 11:37:50')
   (4.1ms)  COMMIT
   (0.1ms)  BEGIN
  SQL (0.2ms)  INSERT INTO `items` (`name`, `barcode`, `price`, `created_at
     `, `updated_at`) VALUES ('Labda', 1508845070, 200, '2017-10-24
     11:37:50', '2017-10-24 11:37:50')
   (1.8ms)  COMMIT
...
```

Végül random felhasználók kölcsönözzenek ki random sporteszközöket. A felsorolás típusú objektumra meghívott Ruby `collect` metódus egy blokkot vár paraméterül, és a blokk visszatérési értékéből tömböt épít. A szokásos `do..end` blokk helyett a felhasználók modell `id` getter metódusa lesz a blokk.

```
irb(main):039:0> User.all.collect(&:id)
  User Load (0.8ms)  SELECT `users`.* FROM `users`
=> [1, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13]
irb(main):040:0> User.all.collect(&:id).sample(1)[0]
  User Load (0.2ms)  SELECT `users`.* FROM `users`
=> 1
irb(main):041:0> User.all.collect(&:id).sample(1)[0]
  User Load (0.3ms)  SELECT `users`.* FROM `users`
=> 10
irb(main):051:0> for i in 1..50 do
irb(main):052:1* Borrow.create user_id: User.all.collect(&:id).sample(1)[0],
     item_id: Item.all.collect(&:id).sample(1)[0], status: Borrow.statuses.
    to_a.sample(1)[0][1]
irb(main):053:1> end
  User Load (0.2ms)  SELECT `users`.* FROM `users`
  Item Load (0.1ms)  SELECT `items`.* FROM `items`
   (0.1ms)  BEGIN
  User Load (0.3ms)  SELECT  `users`.* FROM `users` WHERE `users`.`id` = 6
      LIMIT 1
  Item Load (0.3ms)  SELECT  `items`.* FROM `items` WHERE `items`.`id` = 12
      LIMIT 1
  SQL (0.2ms)  INSERT INTO `borrows` (`user_id`, `item_id`, `status`, `
      created_at`, `updated_at`) VALUES (6, 12, 2, '2017-10-24 11:42:23', '
      2017-10-24 11:42:23')
   (1.7ms)  COMMIT
  User Load (0.2ms)  SELECT `users`.* FROM `users`
  Item Load (0.1ms)  SELECT `items`.* FROM `items`
   (0.1ms)  BEGIN
  User Load (0.1ms)  SELECT  `users`.* FROM `users` WHERE `users`.`id` = 6
      LIMIT 1
  Item Load (0.2ms)  SELECT  `items`.* FROM `items` WHERE `items`.`id` = 8
      LIMIT 1
  SQL (0.4ms)  INSERT INTO `borrows` (`user_id`, `item_id`, `status`, `
      created_at`, `updated_at`) VALUES (6, 8, 1, '2017-10-24 11:42:23', '
      2017-10-24 11:42:23')
   (3.3ms)  COMMIT
...
```

Nézzük meg, hogy az így létrehozott relációkat hogyan használhatjuk. Keressünk egy felhasználót (55. sor), és kérdezzük le a kölcsönzéseit (56. sor), és azok számosságát (57. sor). Rendezzük a felhasználó kölcsönzéseit létrehozási időpont szerint csökkenő (58. sor), növekvő (59. sor), illetve eszközazonosító szerint növekvő sorrendbe (60. sor). Végül nézzük meg, hogy a felhasználó milyen eszközöket kölcsönzött ki e kölcsönzések folyamán.

```
irb(main):055:0> u = User.find 11
  User Load (0.7ms)  SELECT  `users`.* FROM `users` WHERE `users`.`id` = 11
      LIMIT 1
=> #<User id: 11, email: "David_Purple@mail.bme.hu", encrypted_password: "
    ce6cf8ffaf6c9db38a2c7272ada8bd308c6e2b90", name: "David Purple", phone:
    "+3611111111", created_at: "2017-10-24 11:35:17", updated_at:
    "2017-10-24 11:35:17", salt: "ec0b41d7bd6b32c4d4ab6583edd80097">
irb(main):056:0> u.borrows
```

```
  Borrow Load (0.4ms)  SELECT  `borrows`.* FROM `borrows` WHERE `borrows`.`
      user_id` = 11 LIMIT 11
=> #<ActiveRecord::Associations::CollectionProxy [#<Borrow id: 30, user_id:
    11, item_id: 39, status: "borrowed", return_time: nil, created_at:
    "2017-10-24 11:42:23", updated_at: "2017-10-24 11:42:23">, #<Borrow id:
    36, user_id: 11, item_id: 21, status: "lost", return_time: nil,
    created_at: "2017-10-24 11:42:23", updated_at: "2017-10-24 11:42:23">,
    #<Borrow id: 37, user_id: 11, item_id: 79, status: "available",
    return_time: nil, created_at: "2017-10-24 11:42:23", updated_at:
    "2017-10-24 11:42:23">]>
irb(main):057:0> u.borrows.count
   (0.8ms)  SELECT COUNT(*) FROM `borrows` WHERE `borrows`.`user_id` = 11
=> 3
irb(main):058:0> u.borrows.order(created_at: :desc)
  Borrow Load (6.9ms)  SELECT  `borrows`.* FROM `borrows` WHERE `borrows`.`
      user_id` = 11 ORDER BY `borrows`.`created_at` DESC LIMIT 11
=> #<ActiveRecord::AssociationRelation [#<Borrow id: 30, user_id: 11,
    item_id: 39, status: "borrowed", return_time: nil, created_at:
    "2017-10-24 11:42:23", updated_at: "2017-10-24 11:42:23">, #<Borrow id:
    36, user_id: 11, item_id: 21, status: "lost", return_time: nil,
    created_at: "2017-10-24 11:42:23", updated_at: "2017-10-24 11:42:23">,
    #<Borrow id: 37, user_id: 11, item_id: 79, status: "available",
    return_time: nil, created_at: "2017-10-24 11:42:23", updated_at:
    "2017-10-24 11:42:23">]>
irb(main):059:0> u.borrows.order(created_at: :asc)
  Borrow Load (0.5ms)  SELECT  `borrows`.* FROM `borrows` WHERE `borrows`.`
      user_id` = 11 ORDER BY `borrows`.`created_at` ASC LIMIT 11
=> #<ActiveRecord::AssociationRelation [#<Borrow id: 30, user_id: 11,
    item_id: 39, status: "borrowed", return_time: nil, created_at:
    "2017-10-24 11:42:23", updated_at: "2017-10-24 11:42:23">, #<Borrow id:
    36, user_id: 11, item_id: 21, status: "lost", return_time: nil,
    created_at: "2017-10-24 11:42:23", updated_at: "2017-10-24 11:42:23">,
    #<Borrow id: 37, user_id: 11, item_id: 79, status: "available",
    return_time: nil, created_at: "2017-10-24 11:42:23", updated_at:
    "2017-10-24 11:42:23">]>
irb(main):060:0> u.borrows.order(item_id: :asc)
  Borrow Load (0.8ms)  SELECT  `borrows`.* FROM `borrows` WHERE `borrows`.`
      user_id` = 11 ORDER BY `borrows`.`item_id` ASC LIMIT 11
=> #<ActiveRecord::AssociationRelation [#<Borrow id: 36, user_id: 11,
    item_id: 21, status: "lost", return_time: nil, created_at: "2017-10-24
    11:42:23", updated_at: "2017-10-24 11:42:23">, #<Borrow id: 30, user_id:
     11, item_id: 39, status: "borrowed", return_time: nil, created_at:
    "2017-10-24 11:42:23", updated_at: "2017-10-24 11:42:23">, #<Borrow id:
    37, user_id: 11, item_id: 79, status: "available", return_time: nil,
    created_at: "2017-10-24 11:42:23", updated_at: "2017-10-24 11:42:23">]>
irb(main):061:0> u.items
  Item Load (0.4ms)  SELECT  `items`.* FROM `items` INNER JOIN `borrows` ON
      `items`.`id` = `borrows`.`item_id` WHERE `borrows`.`user_id` = 11
      LIMIT 11
=> #<ActiveRecord::Associations::CollectionProxy [#<Item id: 39, name: "
    Labda", barcode: 1508845070, price: 3600, created_at: "2017-10-24
    11:37:50", updated_at: "2017-10-24 11:37:50">, #<Item id: 21, name: "
    Labda", barcode: 1508845070, price: 1800, created_at: "2017-10-24
    11:37:50", updated_at: "2017-10-24 11:37:50">, #<Item id: 79, name: "
    Labda", barcode: 1508845071, price: 7600, created_at: "2017-10-24
    11:37:51", updated_at: "2017-10-24 11:37:51">]>
irb(main):062:0>
```

Az adatbázisunkba kezdeti adatokat reprodukálható módon is bevihetünk anélkül, hogy a Rails konzolt használnánk.A `db/seeds.rb` fájlban elhelye-

zett Ruby metódushívásokkal tudunk adatokat megfelvenni. Itt pontosan azon utasításoknak kell szerepelniük, amiket a konzolon kiadnánk adatok rögzítésekor. Ha az adatokat ebben a fájban adjuk meg, akkor az az adatbázis újrainicializálása után mindig helyreállítható lesz. Vegyünk ezen kívül fel még adatokat, amik a nézetek fejlesztésésben játszanak majd szerepet. Az adatokatokat ezek megadása után a következő paranccsal tudjuk betölteni az adatbázisba:

```
rake db:seed
```

Az előző gyakorlaton adatbázis nélkül inicializáltuk ezeket a nézeteket. Most már ezt helyre tehetjük a kontrollerben. Töröljük ki a múltkor hozzáadott kódrészleteket, és állítsuk vissza a kikommentezett viselkedést. Ez a `items_controller` fájl metódusait érinti.