Rails MVC, modell, session
Gyakorlat

Kovács Gábor

2013. árpilis 1.

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, az elvégzendő házimunkák `Task` nevű modelljét, amelyek a következő táblákat hozták létre az adatbázisban.

```
mysql> describe users;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      | NO   | PRI | NULL    | auto_increment |
| username    | varchar(255) | YES  |     | NULL    |                |
| password    | varchar(255) | YES  |     | NULL    |                |
| first_name  | varchar(255) | YES  |     | NULL    |                |
| family_name | varchar(255) | YES  |     | NULL    |                |
| email       | varchar(255) | YES  |     | NULL    |                |
| last_login  | datetime     | YES  |     | NULL    |                |
| created_at  | datetime     | YES  |     | NULL    |                |
| updated_at  | datetime     | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)

mysql> describe tasks;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      | NO   | PRI | NULL    | auto_increment |
| deadline    | datetime     | YES  |     | NULL    |                |
| title       | varchar(255) | YES  |     | NULL    |                |
| user_id     | int(11)      | YES  |     | NULL    |                |
| state       | int(11)      | YES  |     | NULL    |                |
| description | text         | YES  |     | NULL    |                |
| created_at  | datetime     | YES  |     | NULL    |                |
| updated_at  | datetime     | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

Első lépésként tegyük rendbe a felhasználói session kezelését, ami a `loginform` kontroller akcióinak megvalósítását, a jelszó titkosítását, és a titkosított jel-

szó adatbázisban való eltárolását jelenti első körben. Másodszor pedig a felhasználó regisztráció folyamatát érinti.

Módosítsuk a `User` modellt, hogy az ne a titkosítatlan, hanem a már titkosított jelszót tárolja el. Ezt egy migráció létrehozásával és alkalmazásával, valamint a modell osztály módosításával tesszük meg, és ha már módosítjuk a modellünkek, akkor vegyünk fel egy egész típusú attribútumot, ami az jelzi az 1 értékével, hogy a felhasználó rendszergazda, vagyis van joga felhasználók felvételére, törlésére, és minden más érték, ami praktikusan a 0 értéket jelenti, a standard felhasználók típusát jelöli.

```
rails generate migration AddSaltToUser salt:string type
  :integer
```

A migrációban minden egyes felhasználói rekordhoz hozzáadunk 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. Mivel az átnevezés migráció nem reverzibilis, az automatikusan generált `change` metódust szétválaszjuk egy `up` és egy `down` metódusra.

```
class AddSaltToUser < ActiveRecord::Migration
  def up
    add_column :users, :salt, :string
    # 0: user, 1: admin
    add_column :users, :type, :integer
    rename_column :users, :password, :encrypted_password
  end

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

A jelszó attribútumot átneveztük, viszont a felületen továbbra is használjuk, ezért csak a modell osztályra korlátozva elérhetővé újra tesszük, és egyúttal kibővítjük a modell osztály elérhető attribútumait a jelszó példányváltozójának különböző változataival[1].

```
class User < ActiveRecord::Base
  attr_accessible :passwd, :passwd_confirmation, :
    ecrypted_passwd
```

[1]Rails 4-eben már nem kell

```
  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 < ActiveRecord::Base
  before_save :encrypt_password

  def self.encrypt(pass,salt)
    Digest::SHA2.hexdigest(salt+pass)
  end

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

Hajtsuk végre a migrációt, és ellenőrizzük, hogy valóban titkosítódik-e a jelszó. Láthatjuk közben, hogy a `users` tábla struktúrája módosult.

```
kovacsg@debian:~/gyakorlat/$ rake db:migrate
(in /home/kovacsg/gyakorlat)
==  AddSaltToUser: migrating
    ==========================================================
-- add_column(:users, :salt, :string)
   -> 0.2208s
-- add_column(:users, :type, :integer)
```

```
   -> 0.0100s
-- rename_column(:users, :password, :encrypted_password
   )
   -> 0.0311s
==  AddSaltToUser: migrated (0.2623s)
   ================================================
```

Először nézzük meg, milyen egyéb módunk van hash kulcs előállítására Railsben.

```
kovacsg@debian:~/gyakorlat/$ rails console
irb(main):001:0> Digest::SHA1.hexdigest('titok')
=> "46ff53e764c4acf97b54db2020573049d2e3dab3"
irb(main):002:0> Digest::SHA2.hexdigest('titok')
=> "5
   be2bcf5718118eaeab4fe7ae57543262082a8fce89420a5fc4799d99af2f161
   "
irb(main):003:0> SecureRandom.hex(8)
=> "576befbceb133a3e"
irb(main):004:0> SecureRandom.hex(16)
=> "a599d8adfc255eb81e6256ae2ccad050"
irb(main):005:0> SecureRandom.base64(8)
=> "FR27gRzi1V4="
```

Hozzunk létre egy új felhasználót a létrehozott weboldalon, és ellenőrizzük, hogy a jelszó titkosítása megfelelően működik-e. Nézzük meg a különböző `ActiveRecord` keresési függvényeket: az egy rekord rendezett vagy rendezetlen lekérdezését elvégző `first`, `take`, `last` függvényeket, a `find_by` és `where` szűrőfeltételeinak használatát!

```
irb(main):017:0> User.last
  User Load (0.6ms)  SELECT `users`.* FROM `users`
    ORDER BY `users`.`id` DESC LIMIT 1
=> #<User id: 3, username: "Lusta", encrypted_password:
   "9544
   bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
   first_name: "Lusta", family_name: "Diszno", email:
   "lusta@bme.hu", last_login: nil, created_at:
   "2014-04-01 10:43:41", updated_at: "2014-04-01
   10:43:41", salt: "1c58656e0f659d2d98a76a73784653 59",
   type: nil>
irb(main):020:0> User.find_by_first_name  'Lusta'
```

```
  User Load (1.6ms)  SELECT `users`.* FROM `users`
    WHERE `users`.`first_name` = 'Lusta' LIMIT 1
=> #<User id: 3, username: "Lusta", encrypted_password:
    "9544
   bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
    first_name: "Lusta", family_name: "Diszno", email:
   "lusta@bme.hu", last_login: nil, created_at:
   "2014-04-01 10:43:41", updated_at: "2014-04-01
   10:43:41", salt: "1c58656e0f659d2d98a76a73784653 59",
    type: nil>
irb(main):021:0> User.
   find_by_first_name_and_family_name 'Lusta', 'Diszno
   '
  User Load (1.4ms)  SELECT `users`.* FROM `users`
    WHERE `users`.`first_name` = 'Lusta' AND `users`.`
    family_name` = 'Diszno' LIMIT 1
=> #<User id: 3, username: "Lusta", encrypted_password:
    "9544
   bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
    first_name: "Lusta", family_name: "Diszno", email:
   "lusta@bme.hu", last_login: nil, created_at:
   "2014-04-01 10:43:41", updated_at: "2014-04-01
   10:43:41", salt: "1c58656e0f659d2d98a76a73784653 59",
    type: nil>
irb(main):023:0> User.find_by :username => 'Lusta'
  User Load (0.8ms)  SELECT `users`.* FROM `users`
    WHERE `users`.`username` = 'Lusta' LIMIT 1
=> #<User id: 3, username: "Lusta", encrypted_password:
    "9544
   bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
    first_name: "Lusta", family_name: "Diszno", email:
   "lusta@bme.hu", last_login: nil, created_at:
   "2014-04-01 10:43:41", updated_at: "2014-04-01
   10:43:41", salt: "1c58656e0f659d2d98a76a73784653 59",
    type: nil>
irb(main):024:0> User.find 3
  User Load (0.6ms)  SELECT `users`.* FROM `users`
    WHERE `users`.`id` = 3 LIMIT 1
=> #<User id: 3, username: "Lusta", encrypted_password:
    "9544
   bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
```

```
    first_name: "Lusta", family_name: "Diszno", email:
    "lusta@bme.hu", last_login: nil, created_at:
    "2014-04-01 10:43:41", updated_at: "2014-04-01
    10:43:41", salt: "1c58656e0f659d2d98a76a7378465359",
     type: nil>
irb(main):025:0> User.find_by username: 'Lusta'
  User Load (0.6ms) SELECT `users`.* FROM `users`
    WHERE `users`.`username` = 'Lusta' LIMIT 1
=> #<User id: 3, username: "Lusta", encrypted_password:
    "9544
    bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
     first_name: "Lusta", family_name: "Diszno", email:
    "lusta@bme.hu", last_login: nil, created_at:
    "2014-04-01 10:43:41", updated_at: "2014-04-01
    10:43:41", salt: "1c58656e0f659d2d98a76a7378465359",
     type: nil>
irb(main):026:0> User.all
  User Load (0.3ms) SELECT `users`.* FROM `users`
=> #<ActiveRecord::Relation [#<User id: 1, username: "
    valaki", encrypted_password: nil, first_name: "
    Valaki", family_name: "Valaki", email: "valaki@bme.
    hu", last_login: nil, created_at: "2014-04-01
    10:40:50", updated_at: "2014-04-01 10:40:50", salt:
    nil, type: nil>, #<User id: 2, username: "Valakimas
    ", encrypted_password: nil, first_name: nil,
    family_name: nil, email: nil, last_login: nil,
    created_at: "2014-04-01 10:41:49", updated_at:
    "2014-04-01 10:41:49", salt: nil, type: nil>, #<User
     id: 3, username: "Lusta", encrypted_password: "9544
    bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
     first_name: "Lusta", family_name: "Diszno", email:
    "lusta@bme.hu", last_login: nil, created_at:
    "2014-04-01 10:43:41", updated_at: "2014-04-01
    10:43:41", salt: "1c58656e0f659d2d98a76a7378465359",
     type: nil>]>
irb(main):027:0> User.all.order(family_name: :desc)
  User Load (0.9ms) SELECT `users`.* FROM `users`
    ORDER BY `users`.`family_name` DESC
=> #<ActiveRecord::Relation [#<User id: 1, username: "
    valaki", encrypted_password: nil, first_name: "
    Valaki", family_name: "Valaki", email: "valaki@bme.
```

```
    hu", last_login: nil, created_at: "2014-04-01
    10:40:50", updated_at: "2014-04-01 10:40:50", salt:
    nil, type: nil>, #<User id: 3, username: "Lusta",
    encrypted_password: "9544
    bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
     first_name: "Lusta", family_name: "Diszno", email:
    "lusta@bme.hu", last_login: nil, created_at:
    "2014-04-01 10:43:41", updated_at: "2014-04-01
    10:43:41", salt: "1c58656e0f659d2d98a76a7378465359",
     type: nil>, #<User id: 2, username: "Valakimas",
    encrypted_password: nil, first_name: nil,
    family_name: nil, email: nil, last_login: nil,
    created_at: "2014-04-01 10:41:49", updated_at:
    "2014-04-01 10:41:49", salt: nil, type: nil>]>
irb(main):028:0> User.where(username: 'valaki')
  User Load (0.8ms) SELECT `users`.* FROM `users`
    WHERE `users`.`username` = 'valaki'
=> #<ActiveRecord::Relation [#<User id: 1, username: "
    valaki", encrypted_password: nil, first_name: "
    Valaki", family_name: "Valaki", email: "valaki@bme.
    hu", last_login: nil, created_at: "2014-04-01
    10:40:50", updated_at: "2014-04-01 10:40:50", salt:
    nil, type: nil>]>
```

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

```
rails generate controller sessions create destroy
```

A `sessions` kontrollerhez nem tartozik nézet, a `loginform`, illetve a `logout` link eseményeit kezeli le. Ellenőrizzül, hogy a `layouts/_loginform.html.erb`-ben a form akciója a `/sessions/create`-re mutat-e, illetve a belépett felhasználó menüjében (`layouts/application.html.erb`) a `Logout` link a `/sessions/destroy`-ra mutat-e. A létrejött nézeteket töröljük, nincs szükség önálló nézetre belépéskor, illetve kilépéskor, megpróbálunk az aktuális oldalon maradni.

A következő lépés a felhasználó hitelesítésének megvalósítása, amit a `User` modellben teszünk meg egy osztálymetódussal. A hitelesítés két argumentummal rendelkezik egy felhasználónévvel és egy jelszóval, és a sikeresen hitelesített felhasználó objektumával vagy `nil`-lel tér vissza. Először megkeresi a rekordok között a felhasználó azonosítójának megfelelő rekordot a

`find_by_username`[2] metódussal, 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 < ActiveRecord::Base
  def self.authenticate(username,pass)
    user = find_by_username username
    user && user.authenticated?(pass) ? user : nil
  end

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

A kontrollerünk ezek után a következőképp néz ki. A hitelesítés imént megírt metódusának visszatérési értékét a `current_user` kontroller példányváltozóhoz rendeljük. A felhasználónevet és a jelszót a `params` hash-ből vesszük ki a `loginform`-ban 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 SessionController < ApplicationController
  def create
    current_user = User.authenticate(params[:username],
        params[:password])
    if current_user
      session[:user] = current_user.id
      current_user.last_login = Time.now
      current_user.save
      redirect_to :back
    else
      flash[:notice] = "Invalid user name or password"
```

[2]Rails 4-től verb!User.where(username:'user')!

```
      redirect_to :back
    end
  end

  def destroy
    reset_session
    flash[:notice] = 'Logged out successfully'
    redirect_to :back
  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 `_loginform.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` kontorller `create` akciója tartozik, a profil módosításához pedig az `update` akció tartozik. Takarítsuk ki az előző gyakorlaton bedrótozott értékeket a kontrollerből, és tegyük vissza a `set_user` metódusba az alapértelmezett viselkedést!

A regisztrációkor az elmentendő felhasználót még az elmentés előtt validáljuk, a felhasználónév attribútumnak nemüresnek (`:presence`) és egyedinek kell lennie (`:uniqueness`), valamint hossza 4 és 18 köze kell esnie, és ha ez nem teljesül, akkor visszajelzünk, hogy nem megfelelő felhasználónévről van szó. A jelszónak és annak ismétlésének meg kell egyeznie (`:confirmation`), ha az elmentett jelszó nem üres (`password_required?` metódussal vizsgálva). Ezeket az ellenőrzéseket a modell osztályban helper metódusokkal tesszük meg.

```
class User < ActiveRecord::Base
  validates :username,
      :presence => true,
      :uniqueness => true,
      :length => { :within => 4..18, :too_long => "
         Username is too long", :too_short => "Username
         is too short"}
  validates :password, :confirmation => true, :if => :
     password_required?
  def password_required?
    encrypted_password.blank? || !password.blank?
  end
end
```

Konzolon ellenőrizhetjük, hogy elmenthető-e létező vagy túl rövid felhasználónévvel egy rekord, illetve, hogy a jelszó és annak megerősítésének meg kell egyeznie! 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.

```
irb(main):001:0> User.find_by username: 'Lusta'
  User Load (0.3ms)  SELECT `users`.* FROM `users`
    WHERE `users`.`username` = 'Lusta' LIMIT 1
=> #<User id: 3, username: "Lusta", encrypted_password:
    "9544
    bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
    first_name: "Lusta", family_name: "Diszno", email:
    "lusta@bme.hu", last_login: "2014-04-01 11:07:16",
    created_at: "2014-04-01 10:43:41", updated_at:
    "2014-04-01 11:07:16", salt: "1
    c58656e0f659d2d98a76a7378465359", type: nil>
irb(main):002:0> w = User.new username:'Lusta'
=> #<User id: nil, username: "Lusta",
    encrypted_password: nil, first_name: nil,
    family_name: nil, email: nil, last_login: nil,
    created_at: nil, updated_at: nil, salt: nil, type:
    nil>
irb(main):003:0> w.save
   (0.3ms)  BEGIN
  User Exists (0.9ms)  SELECT 1 AS one FROM `users`
    WHERE `users`.`username` = BINARY 'Lusta' LIMIT 1
[deprecated] I18n.enforce_available_locales will
```

```
    default to true in the future. If you really want to
     skip validation of your locale you can set I18n.
    enforce_available_locales = false to avoid this
    message.
    (0.4ms)  ROLLBACK
=> false
irb(main):004:0> w.errors
=> #<ActiveModel::Errors:0x00000003c9fab0 @base=#<User
    id: nil, username: "Lusta", encrypted_password: nil,
     first_name: nil, family_name: nil, email: nil,
    last_login: nil, created_at: nil, updated_at: nil,
    salt: nil, type: nil>, @messages={:username=>["has
    already been taken"]}>
irb(main):005:0> w.errors.messages
=> {:username=>["has already been taken"]}
irb(main):006:0> w.username = 'Lus'
=> "Lus"
irb(main):007:0> w.save
    (0.3ms)  BEGIN
  User Exists (0.8ms)  SELECT 1 AS one FROM `users`
     WHERE `users`.`username` = BINARY 'Lus' LIMIT 1
    (0.4ms)  ROLLBACK
=> false
irb(main):008:0> w.errors.messages
=> {:username=>["Username is too short"]}
irb(main):009:0> x = User.new
=> #<User id: nil, username: nil, encrypted_password:
    nil, first_name: nil, family_name: nil, email: nil,
    last_login: nil, created_at: nil, updated_at: nil,
    salt: nil, type: nil>
irb(main):010:0> x.username = 'Lusta3'
=> "Lusta3"
irb(main):011:0> x.password = 'titok'
=> "titok"
irb(main):012:0> x.password_confirmation
=> nil
irb(main):013:0> x.password_confirmation = 'a'
=> "a"
irb(main):014:0> x.save
    (0.5ms)  BEGIN
  User Exists (0.6ms)  SELECT 1 AS one FROM `users`
```

```
    WHERE `users`.`username` = BINARY 'Lusta3' LIMIT 1
  (1.8ms)  ROLLBACK
=> false
```

Ezután módosítsuk az adatbázis sémánkat! Egy felhasználónak több feladata is lehet, e kapcsolat megvalósítása céljából már felvettünk idegen kulcsot a `task` táblába az egész típusú `user_id` attribútummal. Ez egy egy-több kapcsolatot definiál, amit a modell osztályokban a `has_many`, illetve a `belongs_to` metódusok által generált setterek és getterek segítségével elérhetővé tehetünk.

```
class User < ActiveRecord::Base
  has_many :tasks
end
```

A kapcsolat a másik irányban a `belongs_to` metódussal tesszük navigálhatóvá.

```
class Task < ActiveRecord::Base
  belongs_to :user
end
```

Hogy a feladatunk ne legyen egyszerű, tegyük kommentelhetővé a feladatokat! A komment lehessen egy feltöltött kép vagy egy szöveges megjegyzés. Hozzuk először létre a modellt, majd definiáljuk annak kapcsolatát a két, már meglévő modellünkkel! Ezt két idegen kulcs felvételével tesszük meg.

```
rails generate model Comment comment:text user:
    references task:references filename:string mime:
    string
rake db:migrate
```

Egy felhasználó sok megjegyzés tehet, e kapcsolat azonosítására használjuk a `comments_made` settert és gettert. Egy felhasználó egy feladatára sok megjegyzés születhet, ez a kapcsolat nem közvetlen, hanem a feladatokon mint kapcsolótáblán kereszül valósul meg!

```
class User < ActiveRecord::Base
  has_many :comments, :through => :tasks
  has_many :comments_made, :class_name => "Comment"
end
```

Hasonlóan egy feladatra sok komment születhet, így a feladatok modellben is vegyük fel az egy-több relációt, valamint egy feladatra több felhasználó

is tehet kommentet, ez ismét egy közvetett kapcsolat, ami a kommentek modelljén keresztül valósul meg.

```
class Task < ActiveRecord::Base
  belongs_to :user
  has_many :comments
  has_many :users, :through => :comments
end
```

A kommentek modellből nézve is navigálhatóvá tesszük az egy-több relációt az egy irányba a `belongs_to` metódussal.

```
class Comment < ActiveRecord::Base
  belongs_to :user
  belongs_to :task
end
```

Nézzük meg, hogy működnek-e a modell osztályok közötti kapcsolatok! Kérdezzük le egy felhasználó feladatait! Láthatjuk, hogy a getter a `has_many` után megadott azonosítóval létrejött. Hozzunk létre egy új feladatot, és rendeljük a feladatot az egyik felhasználónkhoz, majd mentsük el az adatbázisba, majd kérdezzük le a felhasználó feladatait! Következő lépésként hozzunk létre egy új megjegyzést, amit asszociáljunk ehhez a feladathoz, és egy másik felhasználóhoz! Az asszociáció történhet a `belongs_to` setterén keresztül, és a felhasználó és a feladat azonosítóját hozzárendelhetjük közvetlenül a komment példány `user_id`, illetve `task_id` mezőjéhez, a két megoldás ekvivalens. Ellenőrizzük a létrejött asszociációkat különös tekintettel a `comments_made` nevű átnevezett kapcsolatot!

```
rails console
irb(main):001:0> User.first.tasks
  User Load (0.3ms)  SELECT `users`.* FROM `users`
    ORDER BY `users`.`id` ASC LIMIT 1
  Task Load (0.4ms)  SELECT `tasks`.* FROM `tasks`
    WHERE `tasks`.`user_id` = 1
=> #<ActiveRecord::Associations::CollectionProxy []>
irb(main):002:0> l = User.find 3
  User Load (1.1ms)  SELECT `users`.* FROM `users`
    WHERE `users`.`id` = 3 LIMIT 1
=> #<User id: 3, username: "Lusta", encrypted_password:
    "9544
   bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
    first_name: "Lusta", family_name: "Diszno", email:
```

```
    "lusta@bme.hu", last_login: "2014-04-01 11:07:16",
    created_at: "2014-04-01 10:43:41", updated_at:
    "2014-04-01 11:07:16", salt: "1
    c58656e0f659d2d98a76a7378465359", type: nil>
irb(main):003:0> t = Task.new
=> #<Task id: nil, deadline: nil, title: nil, user_id:
    nil, state: nil, description: nil, created_at: nil,
    updated_at: nil>
irb(main):004:0> t.deadline = Time.now + 3.days
=> 2014-04-04 13:35:28 +0200
irb(main):005:0> t.user = l
=> #<User id: 3, username: "Lusta", encrypted_password:
     "9544
    bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
     first_name: "Lusta", family_name: "Diszno", email:
    "lusta@bme.hu", last_login: "2014-04-01 11:07:16",
    created_at: "2014-04-01 10:43:41", updated_at:
    "2014-04-01 11:07:16", salt: "1
    c58656e0f659d2d98a76a7378465359", type: nil>
irb(main):006:0> t.user
=> #<User id: 3, username: "Lusta", encrypted_password:
     "9544
    bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
     first_name: "Lusta", family_name: "Diszno", email:
    "lusta@bme.hu", last_login: "2014-04-01 11:07:16",
    created_at: "2014-04-01 10:43:41", updated_at:
    "2014-04-01 11:07:16", salt: "1
    c58656e0f659d2d98a76a7378465359", type: nil>
irb(main):007:0> t.state = 5
=> 5
irb(main):009:0> t.description = 'Rendetlenseg van'
=> "Rendetlenseg van"
irb(main):010:0> t
=> #<Task id: nil, deadline: "2014-04-04 11:35:28",
    title: nil, user_id: 3, state: 5, description: "
    Rendetlenseg van", created_at: nil, updated_at: nil>
irb(main):011:0> t.save
   (0.3ms)  BEGIN
  SQL (0.7ms)  INSERT INTO `tasks` (`created_at`, `
     deadline`, `description`, `state`, `updated_at`, `
     user_id`) VALUES ('2014-04-01 11:36:48', '
```

```
      2014-04-04 11:35:28', 'Rendetlenseg van', 5, '
      2014-04-01 11:36:48', 3)
   (94.0ms)  COMMIT
=> true
irb(main):012:0> l.tasks
  Task Load (0.6ms)  SELECT `tasks`.* FROM `tasks`
    WHERE `tasks`.`user_id` = 3
=> #<ActiveRecord::Associations::CollectionProxy [#<
   Task id: 1, deadline: "2014-04-04 11:35:28", title:
   nil, user_id: 3, state: 5, description: "
   Rendetlenseg van", created_at: "2014-04-01
   11:36:48", updated_at: "2014-04-01 11:36:48">]>
irb(main):002:0> l = User.find 1
  User Load (1.1ms)  SELECT `users`.* FROM `users`
    WHERE `users`.`id` = 1 LIMIT 1
=> #<User id: 1, username: "valaki", encrypted_password
   : nil, first_name: "Valaki", family_name: "Valaki",
   email: "valaki@bme.hu", last_login: nil, created_at:
    "2014-04-01 10:40:50", updated_at: "2014-04-01
   10:40:50", salt: nil, type: nil>
irb(main):004:0> c = Comment.new
=> #<Comment id: nil, comment: nil, user_id: nil,
   task_id: nil, filename: nil, mime: nil, created_at:
   nil, updated_at: nil>
irb(main):005:0> c.comment="legkozelebb alaposabban
   takarits"
=> "legkozelebb alaposabban takarits"
irb(main):006:0> c.task_id = t.id
=> 1
irb(main):007:0> c.user_id = l.id
=> 1
irb(main):008:0> c.save
   (0.3ms)  BEGIN
  SQL (0.6ms)  INSERT INTO `comments` (`comment`, `
     created_at`, `task_id`, `updated_at`, `user_id`)
     VALUES ('legkozelebb alaposabban takarits', '
     2014-04-01 11:43:08', 1, '2014-04-01 11:43:08', 1)
   (484.4ms)  COMMIT
=> true
irb(main):002:0> l.comments_made
  Comment Load (0.1ms)  SELECT `comments`.* FROM `
```

```
      comments` WHERE `comments`.`user_id` = 1
=> #<ActiveRecord::Associations::CollectionProxy [#<
    Comment id: 1, comment: "legkozelebb alaposabban
    takarits", user_id: 1, task_id: 1, filename: nil,
    mime: nil, created_at: "2014-04-01 11:43:08",
    updated_at: "2014-04-01 11:43:08">]>
irb(main):003:0> l.comments
  Comment Load (0.8ms)  SELECT `comments`.* FROM `
      comments` INNER JOIN `tasks` ON `comments`.`
      task_id` = `tasks`.`id` WHERE `tasks`.`user_id` =
      1
=> #<ActiveRecord::Associations::CollectionProxy []>
irb(main):001:0> t = Task.find 1
  Task Load (0.1ms)  SELECT `tasks`.* FROM `tasks`
     WHERE `tasks`.`id` = 1 LIMIT 1
=> #<Task id: 1, deadline: "2014-04-04 11:35:28", title
    : nil, user_id: 3, state: 5, description: "
    Rendetlenseg van", created_at: "2014-04-01
    11:36:48", updated_at: "2014-04-01 11:36:48">
irb(main):002:0> t.user
  User Load (0.4ms)  SELECT `users`.* FROM `users`
     WHERE `users`.`id` = 3 ORDER BY `users`.`id` ASC
      LIMIT 1
=> #<User id: 3, username: "Lusta", encrypted_password:
     "9544
    bdac71f655b58df3dbca02bb3c7a487af1dcc020e739178...",
     first_name: "Lusta", family_name: "Diszno", email:
    "lusta@bme.hu", last_login: "2014-04-01 11:07:16",
    created_at: "2014-04-01 10:43:41", updated_at:
    "2014-04-01 11:07:16", salt: "1
    c58656e0f659d2d98a76a7378465359", type: nil>
irb(main):003:0> t.users
  User Load (1.8ms)  SELECT `users`.* FROM `users`
     INNER JOIN `comments` ON `users`.`id` = `comments
      `.`user_id` WHERE `comments`.`task_id` = 1
=> #<ActiveRecord::Associations::CollectionProxy [#<
    User id: 1, username: "valaki", encrypted_password:
    nil, first_name: "Valaki", family_name: "Valaki",
    email: "valaki@bme.hu", last_login: nil, created_at:
     "2014-04-01 10:40:50", updated_at: "2014-04-01
    10:40:50", salt: nil, type: nil>]>
```

Az adatbázisunkat mindjárt fel is tölthetjük kezdeti adatokkal, hogy fejlesztés közben adatbázisból származó adatokkal tudjunk dolgozni. Ezt a `db/seeds.rb` fájlban elhelyezett Ruby metódushívásokkal tudjuk megtenni. Ide pontosan azon utasításoknak kell szerepelniük, amiket a konzolon kiadtunk. Az adatokatokat ezek megadása után a következő paranccsal tudjuk betölteni az adatbázisba:

```
rake db:seed
```