Rails MVC, modell, session
Gyakorlat

Kovács Gábor

2012. október 24.

Az előző gyakorlaton megkezdett példát folytatjuk a megoldások beadásának megvalósításával. Az előző alkalommal megvalósított nézetek mögé megtervezzük a modelleket és a kontrollereket. Az előző alkalommak két modellt hoztunk létre, a `User` és a `Subject` modellt, 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 |
| email      | varchar(255) | YES  |     | NULL    |                |
| password   | varchar(255) | YES  |     | NULL    |                |
| username   | varchar(255) | YES  |     | NULL    |                |
| created_at | datetime     | NO   |     | NULL    |                |
| updated_at | datetime     | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
mysql> describe subjects;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | int(11)      | NO   | PRI | NULL    | auto_increment |
| label      | varchar(255) | YES  |     | NULL    |                |
| name       | varchar(255) | YES  |     | NULL    |                |
| code       | varchar(255) | YES  |     | NULL    |                |
| sheet      | varchar(255) | YES  |     | NULL    |                |
| created_at | datetime     | NO   |     | NULL    |                |
| updated_at | datetime     | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
```

Először is vegyünk fel egy szerep attribútumot a `users` táblába. Ez legyen egész típusú és különböztesse meg az oktató, hallgató és adminisztrátor típusú felhasználókat. Majd kapcsoljuk össze e két modellt, készítsünk egy migrációt, ami felvesz egy a `users` tábla rekordjait hivatkozó idegen kulcs attribútumot a `subjects` táblába. A `User` modell osztályban felvesszük az `attr_accessible` metódus argumentumai közé az újonnan létrehozott attribútumot (`:role`).

```
rails generate migration AddRoleToUsers role:integer
rails generate migration AddLecturerIdToSubjects
  lecturer_id:integer
```

Az első migrációt végrehajtó Ruby osztály így néz ki:

```
class AddRoleToUsers < ActiveRecord::Migration
  def change
    add_column :users, :role, :integer
  end
end
```

Az utóbbi pedig így:

```
class AddLecturerIdToSubjects < ActiveRecord::Migration
  def up
    add_column :subjects, :lecturer_id, :integer
  end

  def down
    remove_column :subjects, :lecturer_id
  end
end
```

Hozzuk létre a harmadik modellünket a tanórákat, amelyeket a névütközést elkerülendő hívjunk `Clazz`-nak. A tanóra egy tantárgyhoz tartozik, ezért fel kell vennünk egy arra vonatkozó idegen kulcsot (`subject_id`), továbbá rendelkezik egy a szoba számával azonosítható hellyel (`room`) és időponttal (`date`), amiket egy a `string`, illetve `datetime` típusú attribútummal reprezentálunk.

```
rails generate model Clazz date:datetime room:string
  subject_id:integer
```

A hallgatók jelenlétét az egyes tanórákon a `users` és a `clazzs` táblák összekapcsolásával modellezhetjük. Mivel ez egy több-több reláció a két tábla között, fel kell vennünk egy kapcsolótáblát:

```
rails generate migration CreateClazzsUsersJoinTable
```

A tábla egy-egy idegen kulcsot tartalmaz a `users`, illetve a `clazzs` táblákra, és az `id` attribútum generálását letiltjtuk.

```
class CreateClazzsUsersJoinTable < ActiveRecord::
  Migration
```

```
  def up
    create_table :clazzs_users, :id => false do |t|
      t.integer :clazz_id
      t.integer :user_id
    end
  end

  def down
    drop_table :clazzs_users
  end
end
```

Végrehajtjuk a migrációt:

```
rake db:migrate
```

Hozzunk létre néhány objektumot, és tároljuk el azokat az adatbázisban. Egy felhasználót, egy tantárgyat és a tárgy néhány óráját. Az `ActiveRecord` leszármazottak, vagyis az összes modell osztály `create` osztálymetódusa megfelel egy konstruktorhívásnak és egy rákövetkező `save`-nek. A tanórákat egy ciklussal inicializáljuk, a dátumot a ciklusváltozóval módosítva. A Rails érdekes kiegészítése az egész literálok vonatkozásában az idő típusú használatuk (9. sor).

```
irb(main):001:0> u = User.create neptun:"oweyoa", email
    :"kovacsg@tmit.bme.hu", passwd:"titok", role:1
   (0.2ms)  BEGIN
  SQL (0.8ms)  INSERT INTO `users` (`created_at`, `
      email`, `neptun`, `passwd`, `role`, `updated_at`)
      VALUES ('2012-10-10 16:08:00', 'kovacsg@tmit.bme.
      hu', 'oweyoa', 'titok', 1, '2012-10-10 16:08:00')
   (497.4ms)  COMMIT
=> #<User id: 1, neptun: "oweyoa", passwd: "titok",
    email: "kovacsg@tmit.bme.hu", created_at:
    "2012-10-10 16:08:00", updated_at: "2012-10-10
    16:08:00", role: 1>
irb(main):002:0> s = Subject.new
=> #<Subject id: nil, name: nil, code: nil, sheet: nil,
     created_at: nil, updated_at: nil, lecturer_id: nil>
irb(main):003:0> s.name="RoR"
=> "RoR"
irb(main):004:0> s.code = "BMEVITMBV17"
=> "BMEVITMBV17"
```

```
irb(main):005:0> s.sheet = "ror.txt"
=> "ror.txt"
irb(main):006:0> s.lecturer_id = u.id
=> 1
irb(main):007:0> s.save
   (0.2ms)  BEGIN
  SQL (2.4ms)  INSERT INTO `subjects` (`code`, `
      created_at`, `lecturer_id`, `name`, `sheet`, `
      updated_at`) VALUES ('BMEVITMBV17', '2012-10-10␣
      16:09:33', 1, 'RoR', 'ror.txt', '2012-10-10␣
      16:09:33')
   (482.2ms)  COMMIT
=> true
irb(main):008:0> for i in 0..13
irb(main):009:1>   c = Clazz.new date:8.weeks.ago+i*1.
   week, room:"I.B.138"
irb(main):010:1>   c.subject_id = s.id
irb(main):011:1>   c.save
irb(main):012:1> end
   (0.1ms)  BEGIN
  SQL (0.5ms)  INSERT INTO `clazzs` (`created_at`, `
      date`, `room`, `subject_id`, `updated_at`) VALUES
      ('2012-10-10␣16:14:02', '2012-08-15␣16:14:02', 'I.
      B.138', 1, '2012-10-10␣16:14:02')
   (402.6ms)  COMMIT
...
=> 0..13
```

Az adatbázisban ezek után már léteznek a relációk az frissen létrehozott rekordjaink között. Ahhoz, hogy ez a modell osztályok közötti kapcsolatokban is megjelenjen, ki kell egészítenünk a modell oszályainkat.

A `User` modellünk felhasználókat modelleznek, egy felhasználónak, legyen az oktató vagy hallgató, sok tantárgya lehet, ezért a `has_many` metódussal összekapcsoljuk a két modellt, mivel a `Subject` modellben a felhasználókra vonatkozó attribútum neve `lecturer_id`, ezt külön jelenzünk kell. Egy felhasználónak sok tantárgya lehet, amelyeknek sok tanórái vannak, a felhasználó tanóráit a `Subject` modellen keresztül (`through`) közvetve elérhetővé tesszük a `clazzs` accessorokkal. A felhasználó tanórái tekintetében az imént definiált kapcsolótáblát használjuk fel. Mivel az előbb már elhasználtuk a tanórák osztályra vonatkozó azonosítónkat, új azonosítót kell kitalálnunk miközben megadjuk az azonosító által vonatkozott modell osztály nevét.

```
class User < ActiveRecord::Base
  has_many :subjects, :foreign_key => "lecturer_id"
  has_many :clazzs, :through => :subjects
  has_and_belongs_to_many :classes, :class_name => "
      Clazz"
end
```

A tantárgyak modelljét két accessorral egészítjük ki. Minden tantárgynak van oktatója (`belongs_to`), akire a `lecturer` attribútummal hivatkozunk, de mivel a típusa nem `Lecturer`, hanem `User`, az osztálynevet külön meg kell említenünk. Egy tantárgynak több tanórája van, ezt a `has_many` metódussal jelezzük.

```
class Subject < ActiveRecord::Base
  belongs_to :lecturer, :class_name => "User"
  has_many :clazzs
end
```

Egy tanóra egy tárgyhoz tartozik, a `belongs_to`-val navigálthatóvá tesszük a modellek között is ezt a relációt. A tanórán résztvevő hallgatókat az `attendees` accessorral érjük el, amivel a `User` modell osztály példányaira hivatkozunk.

```
class Clazz < ActiveRecord::Base
  belongs_to :subject
  has_and_belongs_to_many :attendees, :class_name=>"
      User"
end
```

Hozzunk létre egy új felhasználót (13. sor), egy hallgatót, aki később szorgalmasan látogatja a tantárgyunk óráit. A keresésre a `find` metódust használhatjuk, ami paraméterezhető az elsődleges kulccsal, kikereshetjük az első (`:first`) rekordot, és az összes (`:all`) rekord enumerációját. Meggyőződhetünk, hogy a felhasználók és a tárgyak modell objektumai között is használhatók a relációk (20. sor), és a modell osztályokon keresztül megvalósuló kapcsolt reláció is működik.

```
irb(main):013:0> u = User.create :neptun => "aaaaaa", :
    email => "a@mail.bme.hu", :passwd => "a", :role => 2
   (0.1ms)  BEGIN
  SQL (2.0ms)  INSERT INTO `users` (`created_at`, `
      email`, `neptun`, `passwd`, `role`, `updated_at`)
```

```
      VALUES ('2012-10-10 16:15:57', 'a@mail.bme.hu', '
      aaaaaa', 'a', 2, '2012-10-10 16:15:57')
    (383.8ms)  COMMIT
=> #<User id: 2, neptun: "aaaaaa", passwd: "a", email:
    "a@mail.bme.hu", created_at: "2012-10-10 16:15:57",
    updated_at: "2012-10-10 16:15:57", role: 2>
irb(main):015:0> u = User.find :first
  User Load (2.7ms)  SELECT `users`.* FROM `users`
      LIMIT 1
=> #<User id: 1, neptun: "oweyoa", passwd: "titok",
    email: "kovacsg@tmit.bme.hu", created_at:
    "2012-10-10 16:08:00", updated_at: "2012-10-10
    16:08:00", role: 1>
irb(main):016:0> u = User.find 1
  User Load (0.6ms)  SELECT `users`.* FROM `users`
      WHERE `users`.`id` = 1 LIMIT 1
=> #<User id: 1, neptun: "oweyoa", passwd: "titok",
    email: "kovacsg@tmit.bme.hu", created_at:
    "2012-10-10 16:08:00", updated_at: "2012-10-10
    16:08:00", role: 1>
irb(main):017:0> u = User.find 2
  User Load (0.8ms)  SELECT `users`.* FROM `users`
      WHERE `users`.`id` = 2 LIMIT 1
=> #<User id: 2, neptun: "aaaaaa", passwd: "a", email:
    "a@mail.bme.hu", created_at: "2012-10-10 16:15:57",
    updated_at: "2012-10-10 16:15:57", role: 2>
irb(main):018:0> u = User.find :all
  User Load (0.7ms)  SELECT `users`.* FROM `users`
=> [#<User id: 1, neptun: "oweyoa", passwd: "titok",
    email: "kovacsg@tmit.bme.hu", created_at:
    "2012-10-10 16:08:00", updated_at: "2012-10-10
    16:08:00", role: 1>, #<User id: 2, neptun: "aaaaaa",
     passwd: "a", email: "a@mail.bme.hu", created_at:
    "2012-10-10 16:15:57", updated_at: "2012-10-10
    16:15:57", role: 2>]
irb(main):019:0> u = User.find 1
  User Load (0.3ms)  SELECT `users`.* FROM `users`
      WHERE `users`.`id` = 1 LIMIT 1
=> #<User id: 1, neptun: "oweyoa", passwd: "titok",
    email: "kovacsg@tmit.bme.hu", created_at:
    "2012-10-10 16:08:00", updated_at: "2012-10-10
```

```
    16:08:00", role: 1>
irb(main):020:0> u.subjects
  Subject Load (0.6ms)  SELECT `subjects`.* FROM `
      subjects` WHERE `subjects`.`lecturer_id` = 1
=> [#<Subject id: 1, name: "RoR", code: "BMEVITMBV17",
    sheet: "ror.txt", created_at: "2012-10-10 16:09:33",
     updated_at: "2012-10-10 16:09:33", lecturer_id: 1>]
irb(main):021:0> u.clazzs
  Clazz Load (0.5ms)  SELECT `clazzs`.* FROM `clazzs`
     INNER JOIN `subjects` ON `clazzs`.`subject_id` = `
      subjects`.`id` WHERE `subjects`.`lecturer_id` = 1
=> [#<Clazz id: 1, date: "2012-08-15 16:14:02", room: "
    I.B.138", subject_id: 1, created_at: "2012-10-10
    16:14:02", updated_at: "2012-10-10 16:14:02">, #<
    Clazz id: 2, date: "2012-08-22 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 3, date: "2012-08-29 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 4, date: "2012-09-05 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 5, date: "2012-09-12 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 6, date: "2012-09-19 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 7, date: "2012-09-26 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 8, date: "2012-10-03 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 9, date: "2012-10-10 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 10, date: "2012-10-17 16:14:03", room: "I.
    B.138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
```

```
    Clazz id: 11, date: "2012-10-24 16:14:03", room: "I.
    B.138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 12, date: "2012-10-31 16:14:03", room: "I.
    B.138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 13, date: "2012-11-07 16:14:03", room: "I.
    B.138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 14, date: "2012-11-14 16:14:03", room: "I.
    B.138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">]
```

Hasonlóan, a tárgyak és a tanórák közötti reláció is működik. Látogassa a frissen létrehozott felhasználó rendszeresen a tárgy óráit (10-12. sor).

```
irb(main):005:0> s = Subject.find 1
  Subject Load (0.4ms)  SELECT `subjects`.* FROM `
      subjects` WHERE `subjects`.`id` = 1 LIMIT 1
=> #<Subject id: 1, name: "RoR", code: "BMEVITMBV17",
    sheet: "ror.txt", created_at: "2012-10-10 16:09:33",
     updated_at: "2012-10-10 16:09:33", lecturer_id: 1>
irb(main):006:0> s.clazzs
  Clazz Load (0.6ms)  SELECT `clazzs`.* FROM `clazzs`
     WHERE `clazzs`.`subject_id` = 1
=> [#<Clazz id: 1, date: "2012-08-15 16:14:02", room: "
    I.B.138", subject_id: 1, created_at: "2012-10-10
    16:14:02", updated_at: "2012-10-10 16:14:02">, #<
    Clazz id: 2, date: "2012-08-22 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 3, date: "2012-08-29 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 4, date: "2012-09-05 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 5, date: "2012-09-12 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 6, date: "2012-09-19 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
```

```
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 7, date: "2012-09-26 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 8, date: "2012-10-03 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 9, date: "2012-10-10 16:14:03", room: "I.B
    .138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 10, date: "2012-10-17 16:14:03", room: "I.
    B.138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 11, date: "2012-10-24 16:14:03", room: "I.
    B.138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 12, date: "2012-10-31 16:14:03", room: "I.
    B.138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 13, date: "2012-11-07 16:14:03", room: "I.
    B.138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">, #<
    Clazz id: 14, date: "2012-11-14 16:14:03", room: "I.
    B.138", subject_id: 1, created_at: "2012-10-10
    16:14:03", updated_at: "2012-10-10 16:14:03">]
irb(main):010:0> for c in s.clazzs do
irb(main):011:1* c.attendees << v
irb(main):012:1> end
```

Következő lépésként tegyük rendbe a felhasználói session kezelését, ami a `loginform` kontroller akcióinak megvalósítását, a jelszó titkosítását, és a titkosított jelszó 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.

```
rails generate migration AddSaltToUsers salt:string
```

A migrációban minden egyes felhasználói rekordhoz hozzáadunk egy egyéni a jelszó titkosításához használt kulcsot, a jelszó attribútumot pedig átnevezzük, így az titkosítatlanul nem kerülhet bele az adatbázisba.

```
class AddSaltToUsers < ActiveRecord::Migration
  def up
    add_column :users, :salt, :string
    rename_column :users, :password, :encrypted_password
  end
  def down
    rename_column :users, :encrypted_password, :password
    remove_column :users, :salt
  end
end
```

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

```
class User < ActiveRecord::Base
  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::SHA1.hexdigest(salt+pass)
  end

  def encrypt_password
    return if password.blank?
    if new_record?
      self.salt = Digest::SHA1.hexdigest(Time.now.to_s+
        email+"--"+username)
```

```
    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. A táblában lévő rekordokra a `salt` attribútum nem inicializált, ezért csak annak felvétele után titkosíthatjuk a jelszavunkat.

```
rake db:migrate
rails console
irb(main):006:0> u.salt = Digest::SHA1.hexdigest(Time.
   now.to_s+u.email+'--'+u.neptun)
=> "9a532bb7afb35973ad887a85a941eb03f8575222"
irb(main):007:0> u.save
   (0.1ms)  BEGIN
   (1.2ms)  UPDATE `users` SET `salt` = '9
      a532bb7afb35973ad887a85a941eb03f8575222', `
      encrypted_passwd` = '
      d33ed8a9e9f7ad359db58e3088ad59012a539565', `
      updated_at` = '2012-10-10 16:41:22' WHERE `users
      `.`id` = 1
   (2.5ms)  COMMIT
=> true
irb(main):008:0>
=> true
```

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
```

A `session` 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.

```
rails generate controller session
```

A következő lépés a felhasználó hitelesítésének megvalósítása, amit a `User` modellben teszünk meg egy osztálymetódussal. A hitelesítés két argumen-

tummal 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` 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,password)
    user = find_by_username username
    user && user.authenticated?(password) ? 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
      redirect_to :back
    else
      flash[:notice]='Invalid user name or password'
```

```
      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.

Regisztrárciókor a form paraméterei alapján létrehozunk egy új felhasználó objektumot, és megpróbáljuk elmenteni azt az adatbázisba, a sikeres volt, akkor a felhasználót automatikusan bejelentkeztetjük, beállítjuk a `session` értékét, és egy `flash` üzenet kíséretében átirányítjuk a felhasználót a kezdőoldalra. Sikertelen mentés esetén újból a regisztrációs oldalt jelenítjük meg rajta egy hibaüzenettel.

```
class UsersController < ApplicationController
  def create
    @user = User.new params[:user]
    if @user.save
      @current_user = @user
```

```
      session[:user] = @user.id
      flash[:notice] = 'Successful registration'
      redirect_to :controller=>'issues', :action=>'
        index'
    else
      flash[:notice] = 'User name already used'
      redirect_to :action=>'new'
    end
  end
end
```

Az aktuális felhasználó példányára a `new` és `create` kivételével minden akció esetén szükségünk van, így azt a `before_filter`-rel minden akció lefutása előtt inicializálni tudjuk. A filterhez a privát `find_user` metódust rendeljük, ami beállítja a `@user` példányváltozó értékét a `session` alapján. A profil szerkesztést ezután e példányváltozón a `update_attributes` metódussal tesszük meg. Sikeres módosítás esetén a kezdőoldalra navigáljuk a felhasználót egy üzenettel, míg sikertelenség esetén újból a profil oldalt jelenítjük meg egy hibaüzenet kíséretében.

```
class UsersController < ApplicationController
  before_filter :find_user, :except => [:new, :create]
  def update
    if @user.update_attributes params[:user]
      flash[:notice] = 'Update successful'
      redirect_to :controller=>'issues', :action=>'
        index'
    else
      flash[:notice] = 'Could not update user profile'
      redirect_to :action => 'edit', :id=>@user.id
    end
  end

  private
  def find_user
    @user = User.find(params[:id])
  end
end
```