Tesztelés Rails-ben
Gyakorlat

Kovács Gábor

2011. november 22.

A gyakorlat során az előző gyakorlatokon megkezdett feladat megoldását folytatjuk.

A tesztelés előkészítésére először teszt adatokat definiálunk, amelyekre a teszteseteinkben hivatkozni fogunk. Ezeket a Rails alkalmazásunk `test/fixtures` könyvtárában helyezzük el. Az egységtesztek és funkcionális tesztek számára ezeket az adatokat a minden egyes teszt eset elején hivatkozott `test/test_helper.rb` fájl fogja elérhetővé tenni, integrációs tesztek esetén pedig magunk töltjük be.

Definiáljuk az alábbi két `User` példányt az `users.yml` fájlban. Először definiáljuk a titkosításhoz használt `salt` attribútum értékét, amelyre minden egyes felhasználó példányában hivatkozni fogunk. A jelszó megadához a modell osztály `encrypt` metódusát hívjuk segítségül.

```
<% SALT = 'ezzeltitkositunk' %>
<% PASS = 'titok' %>
me:
  username: kovacsg
  email: kovacsg@tmit.bme.hu
  salt: <%= SALT %>
  encrypted_password: <%= User.encrypt PASS, SALT %>
```

A felhasználókhoz hasonló módon megadunk egy forrást is a `sources.yml` fájlban.

```
rorea:
  source: Kovacs Gabor
  subject: RoR eloadas
  released_at: 2011-11-22 12:25
```

A beküldött idézetek tesztadataként (`solutions.yml`) a következő rekordot használjuk. Az idegen kulcsoknál a megfelelő tesztadatfájl egy kulcsát használjuk, ami jelen esetben `rorea` a `sources.yml`-t tekintve, és `me` a `users.yml`-re nézve. Ennek hatására a `source_id` és a `user_id` attribútum automatikusan állítódik.

```
ror:
  quote: "A kovetkezo eloadason zh"
  evaluation: 5
  votes: 7
  user: me
  source: rorea
```

Az idézetekhez fűzött kommentek tekintetében (`comments.yml`) egyetlen tesztrekordot definiálunk, amely a `ror` kulccsal azonosított idézetre vonatkozik.

```
one:
   comment: Nagyon, nagyon vicces
   user: me
   quote: ror
```

Töltsük be a teszt adatbázisba ezeket az adatokat ügyelve arra, hogy a környezetként a teszt környezet legyen beállítva.

```
RAILS_ENV='test' rake db:test:prepare
RAILS_ENV='test' rake db:migrate
RAILS_ENV='test' rake db:fixtures:load
```

Először egy egységtesztet írunk. Az egységtesztek a modell osztályok metódusait és validációit hivatottak ellenőrizni. Az egyégteszteket a Rails projektünk `test/unit` könyvtárában találjuk. Minden egyes modell létrehozása után automatikusan létrejön hozzá egy ahhoz kapcsolódó teszt osztály itt.

Írjunk teszteseteket, amelyek a `User` modellünk

```
validates_presence_of :username
validates_uniquness_of :username
```

validációinak megtörténtét ellenőrzik. Először létrehozunk egy új felhasználót mindenféle inicializáció nélkül, majd megpróbáljuk eltárolni az adatbázisba. A feltételezésünk az, hogy a mentésnek nem szabad sikerülnie. A másik tesztesetben megpróbálunk kétszer azonos felhasználónévvel elmenteni egy-egy felhasználó objektumot.

```
require 'test_helper'

class UserTest < ActiveSupport::TestCase
  test "username is present" do
    u = User.new
    assert !u.save, "Houston, we have a problem"
  end

  test "username is unique" do
    u = User.new
    u.username = 'lalala'
    assert u.save
    v = User.new
    v.username = 'lalala'
    assert !v.save, "Duplicate user name saved"
  end
end
```

A teszt esetet futtatva meggyőződhetünk róla, hogy tényleg nem történik meg a mentés.

```
> rake test:units
Loaded suite /var/lib/gems/1.9.1/gems/rake-0.9.2/lib/
   rake/rake_test_loader
Started

UserTest:
     PASS username is present (0.20s)
     PASS username is unique (0.01s)

Finished in 0.217362 seconds.

2 tests, 3 assertions, 0 failures, 0 errors, 0 skips
```

A funkcionális tesztek a kontrollerek és a nézetek helyes működését ellenőrzik. Alapértelmezés szerint minden egyes a létrehozáskor megadott kontroller akcióra létrejön egy az akció sikeres megjelenítését ellenőrző teszteset.

Először egészítsük ki a `SessionControllerTest` tesztet, amely jelenleg nem tartalmaz teszteseteket. A bejelentkezés eseményt a `SessionController` `create` metódusát használja, a kijelentezés esemény pedig a `destroy` metódust. A bejelentkezés vizsgálatára két tesztesetet készítünk, az egyik a sikeres

esetet vizsgálja, a másik a sikertelent. A kijelentezés vizsgálatára egyetlen tesztesetet írunk. A tesztben a `post` metódust használjuk, amelynek első paramétere a teszelendő akció, a második a HTTP kérés paraméterei, a harmadok pedig a `session` paraméterek. A sikeres bejelentkezés tesztben a tesztadatokban szereplő felhasználónév, jelszó párost küldjük el, a sikertelenben hibás jelszót adunk meg. Sikeres esetben azt feltételezzük, hogy visszairányítódunk a belépés előtti nézetre, a `:user` session paraméter értéke nem `nil`, ráadásul megegyezik a tesztadat azonosítójával. Sikertelen esetben szintén átirányítást várunk, és azt, hogy a `:user` session paraméter `nil` értéket tartalmaz. A kilépés tesztben HTTP kérés paramétereket nem adunk meg, tehát a második paraméter `nil`, azonban harmadik paraméterként beállítjuk a `:user` session paramétert azt imitálva, hogy van bejelentkezett felhasználónk. Válaszként átirányítást várunk az aktuális oldalunkra, és azt, hogy a session paraméter kinullázódik.

```
class SessionControllerTest < ActionController::
    TestCase
  test "login" do
    post :create,
      { :username=>users(:me).username, :password => '
          titok'}
    assert_response :redirect
    assert_not_nil session[:user]
    assert_equal session[:user], users(:me).id
  end

  test "unsuccessful_login" do
    post :create,
      { :username=>users(:me).username, :password => '
          titok2'}
    assert_response :redirect
    assert_nil session[:user]
  end

  test "logout" do
    post :destroy, nil, {:user => users(:me).id}
    assert_response :redirect
    assert_nil session[:user]
  end
end
```

A tesztesetünk kész van azonban nem futhat le sikeresen ugyanis a kont-

rollerben szereplő `redirect_to :back` átirányításban nem definiált a `:back` értéke. Ezt a Rails a `HTTP_REFERER` HTTP kérés paraméterből veszi, így ezt minden egyes tesztesetben be kell állítanunk. Ezt hatékonyan a `setup` metódusban tehetjük meg, amely minden teszteset előtt lefut.

```
class SessionControllerTest < ActionController::
    TestCase
  setup do
    @request.env["HTTP_REFERER"]='/say/hello'
  end
```

A `SourcesControllerTest`-et scaffolddal hoztuk létre, amely automatikusan generálta a teszteseteket és tesztadatokat. A tesztadatokat módosítottuk, így az azokra való hivatkozást frissítenünk kell a `setup`-ban `:rorea`-ra. A tesztesetek közül hármat módosítunk, és kettő továbbit hozzáadunk. Az `index` akció ellenőrzésekor ne elégedjünk meg a HTTP státus kód helyességének vizsgálatával, nézzük meg, hogy a `sources` példányváltozó hordoz-e értéket a kontrollerben, ezt az `assigns` hash segítségével tehetjük meg.

```
class SourcesControllerTest < ActionController::
    TestCase
  include SourcesHelper

  setup do
    @source = sources(:rorea)
  end

  test "should␣get␣index" do
    get :index
    assert_response :success
    assert_not_nil assigns(:sources)
  end
end
```

Új forrás létrehozása csak bejelentkezett felhasználó számára lehetséges, így erre az akcióra írunk még egy tesztesetet. Ha nincs bejelentkezett felhasználó, akkor átirányítás HTTP státuszkódot várunk. Bejelentkezett felhasználó esetén sikeres státuszkódot várunk. A kérést azonban úgy kell megfogalmaznunk, hogy HTTP paramétereket nem adunk meg, tehát a második paraméter `nil`, azonban session paraméterként beállítjuk a `:user` értékét a tesztadatnak megfelelően.

```
class SourcesControllerTest < ActionController::
    TestCase
```

```
  test "should get new" do
    get :new
    assert_response :redirect
  end

  test "should get new when logged in" do
    get :new, nil, :user=>users(:me).id
    assert_response :success
  end

end
```

Forrás szerkesztése esetén a gondolatmenet megegyezik a fentivel azzal a különbséggel, hogy itt mindkét esetben van HTTP kérés paraméterünk, amit a `setup`-ban a tesztadatok alapján inicializált `source` példányváltozóból veszünk.

```
class SourcesControllerTest < ActionController::
   TestCase
  test "should get edit" do
    get :edit, id: @source.to_param
    assert_response :redirect
  end

  test "should get edit when logged in" do
    get :edit, {:id => @source.to_param}, {:user=>users
       (:me).id}
    assert_response :success
  end
end
```

A `UsersControllerTest` osztályunk tesztesetei jelenleg elbuknak, az `edit`, `show` és `forgotten` akciók teszteseteiben a `get` második paramétereként be kell állítanunk az `id` paramétert a tesztadatok alapján. Minden mást az egyszerűség kedvéért változatlanul hagyunk.

```
class UsersControllerTest < ActionController::TestCase
  test "should get new" do
    get :new
    assert_response :success
  end

  test "should get edit" do
```

```
    get :edit, :id => users(:me).id
    assert_response :success
  end

  test "should get show" do
    get :show, :id => users(:me).id
    assert_response :success
  end

  test "should get forgotten" do
    get :forgotten, :id => users(:me).id
    assert_response :success
  end
end
```

Idézetek tesztjeit a `QuotesControllerTest` osztályban definiáljuk. Új idézetet csak bejelenkezett felhasználó hozhat létre. Ezt a források kontrollere esetén bemutatott módon tesszük meg két tesztesettel. Sikertelen esetben átirányítást várunk. Sikeres esetben, amikor beállítjuk a `user` session paramétert a tesztadatok alapján, sikeres státuszkódot várunk, és azt, hogy az idézet kontroller példányváltozója rendelkezik értékkel.

A `comment` akció tesztesetét ki kell egészítenünk egy HTTP kérés paraméterrel, amelyet a tesztadatokból veszünk.

Az idézetek listázását végző `show` akció inicializálja a `quotes` példányváltozót a kontrollerben, tehát feltételezhetjük, hogy rendelkezik értékkel, és a kérés sikeres státuszkóddal tér vissza. A megjelenített nézeten egyetlen idézet jelenik meg, amelyre a következő `assert_select` illesztést végezzük. Azt feltételezzük, hogy a `div.quotebox b` CSS szelektor által kijelölt HTML elem értéke a második paraméterben megadott string, amely a tesztadatokra hivatkozik.

```
class QuotesControllerTest < ActionController::TestCase
  test "should get new" do
    get :new
    assert_response :redirect
  end

  test "should get new when logged in" do
    get :new, nil, :user=>users(:me).id
    assert_response :success
    assert_not_nil assigns[:quote]
  end
```

```
  test "should get comment" do
    get :comment, :id => quotes(:ror).id
    assert_response :success
  end

  test "should get show" do
    get :show
    assert_not_nil assigns[:quotes]
    assert_response :success
    assert_select "div.quotebox b",
      "#{sources(:rorea).source} said in the \n    "+
      "#{sources(:rorea).subject} in year \n    "+
      "#{sources(:rorea).released_at.strftime('%Y')}"
  end
end
```

A funkcionális teszteseteket futtatva láthatjuk, hogy sikeresen lefutnak.

```
rake test:functionals
(in /home/kovacsg/gyakorlat)
Loaded suite /var/lib/gems/1.9.1/gems/rake-0.9.2/lib/
   rake/rake_test_loader
Started

QuotesControllerTest:
     PASS should get comment (0.40s)
     PASS should get edit (0.01s)
     PASS should get new (0.00s)
     PASS should get new when logged in (0.01s)
     PASS should get show (0.12s)

SessionControllerTest:
     PASS login (0.02s)
     PASS logout (0.01s)
     PASS unsuccessful login (0.01s)

SourcesControllerTest:
     PASS should create source (0.08s)
     PASS should destroy source (0.01s)
     PASS should get edit (0.00s)
     PASS should get edit when logged in (0.03s)
```

```
      PASS should get index (0.01s)
      PASS should get new (0.00s)
      PASS should get new when logged in (0.02s)
      PASS should show source (0.01s)
      PASS should update source (0.01s)

UsersControllerTest:
      PASS should get edit (0.03s)
      PASS should get forgotten (0.03s)
      PASS should get new (0.01s)
      PASS should get show (0.02s)

Finished in 0.850195 seconds.

21 tests, 31 assertions, 0 failures, 0 errors, 0 skips
```

A tesztek harmadik típusa az integrációs teszt, amellyel egy böngészési folyamatot ellenőrzünk. Készítsünk egy tesztet az idézetek feltöltésének esetére!

```
rails generate integration_test quote_submission
```

E parancs kiadása után a `test/integration` könyvtárban létrejött egy `quote_submission_test.rb` nevű állomány, ahol az integrációs tesztünk metódusait helyezzük el.

A fájl elején töltsük be az összes teszt adatot, majd definiáljuk a böngészési folyamatot úgy, hogy az a bejelentkezéstől jusson el legalább a forráskiválasztásig. Ne feledkezzünk el a karakterkódolás beállításáról sem, ha ékezetes karaktert használunk!

Az integrációs tesztesetet a funkcionális teszteknél megismert tesztesetekből mint tesztlépésekből tevődik össze. Az első lépésben egy be nem jelentkezett felhasználó próbál új idézetet felvinni a `quotes` kontroller `new` akcióval. Feltételezzük, hogy átirányítás státuszkóddal rendelkező választ kapunk, és visszakerülünk a `/quotes/show` nézetre.

Második tesztlépésként bejelentkezünk a `post_via_redirect` metódussal bejelentkeztetjük a tesztesetünket. A célja a `/session/create` akció. A második paramétere a HTTP kérés paramétereit tartalmazó hash, a harmadik paraméter a `:back` átirányítást támogató HTTP fejrész paramétert állítja be. A feltételezéseink az, hogy átirányítódunk a `/quotes/show` nézetre, és a `:user` session paraméter rendelkezik értékkel, ami nem más, mint a tesztadatok között megadott felhasználó azonosító.

Harmadik tesztlépésként megismételjük az első kérést. Ezúttal sikeres

választ várunk, és azt, hogy a nézeten megjelenik egy link a *Create a new source* felirattal, ezt az `assert_select`-tel ellenőrizzük.

Negyedik tesztlépésként követjül a linket, amely az `add_source` akcióra mutat a `quotes` kontrollerben. Azt feltételezzük, hogy átirányítódunk a `/sources/new` nézetre, és az `:adding_source` session paramétere `true` értéket vesz fel.

```
require 'test_helper'

class QuoteSubmissionTest < ActionDispatch::
   IntegrationTest
  fixtures :all

  test "quote submission" do
    get url_for(:controller=>"quotes", :action=>"new")
    assert_response :redirect
    assert_redirected_to "/quotes/show"
    post_via_redirect '/session/create',
      {:username => users(:me).username,
       :password => 'titok'},
      { "HTTP_REFERER" => '/quotes/show'}
    assert_equal path, '/quotes/show'
    assert_not_nil session[:user]
    assert_equal session[:user], users(:me).id
    get '/quotes/new'
    assert_response :success
    assert_select "a", "Create a new source"
    get '/quotes/add_source'
    assert_redirected_to '/sources/new'
    assert session[:adding_source]
  end
end
```

Futtatva az integrációs tesztet láthatjuk, hogy az egy teszt eset hat teszt lépése során mind a tizenegy ellenőrzésen sikeresen átment.

```
rake test:integration
(in /home/kovacsg/gyakorlat)
Loaded suite /var/lib/gems/1.9.1/gems/rake-0.9.2/lib/
   rake/rake_test_loader
Started
```

```
QuoteSubmissionTest:
     PASS quote submission (0.55s)

Finished in 0.551852 seconds.

1 tests, 9 assertions, 0 failures, 0 errors, 0 skips
```