A Rails keretrendszer
Gyakorlat

Kovács Gábor

2010. október 1.

Amire szükségünk lesz a gyakorlat során: egy telepített Ruby környezet. Először ellenőrizzük, hogy elérhető-e a `ruby` értelmező, majd azt, hogy elérhető-e a `gem` Ruby csomagkezelő. A windowsos telepítő mindkettőt automatikusan elérhetővé teszi, Linuxon a `ruby` és a `rubygems` csomagokat kell telepítenünk [1].

```
kovacsg@vb:~$ ruby --version
ruby 1.8.7 (2010-08-16 patchlevel 302) [i486-linux]
kovacsg@vb:~$ gem --version
1.3.7
kovacsg@vb:~$ locate libsqlite3|less
```

A következő lépés a Rails keretrendszer telepítése, amit a `gem` paranccsal teszünk meg. Ez egyben feltelepíti a legfontosabb kiegészítőket is, illetve a hozzájuk kapcsolódó dokumentációt.

```
kovacsg@vb:~$ sudo bash
[sudo] password for kovacsg:
root@vb:/home/kovacsg# gem install rails
Successfully installed activesupport-3.0.0
Successfully installed builder-2.1.2
Successfully installed i18n-0.4.1
Successfully installed activemodel-3.0.0
Successfully installed rack-1.2.1
Successfully installed rack-test-0.5.6
Successfully installed rack-mount-0.6.13
Successfully installed tzinfo-0.3.23
```

[1] Az egyes disztribúciók esetén eltérhetnek e nevek, Debian csomagkezelőt használó disztribúciók esetén futnak ilyen név alatt

```
Successfully installed abstract-1.0.0
Successfully installed erubis-2.6.6
Successfully installed actionpack-3.0.0
Successfully installed arel-1.0.1
Successfully installed activerecord-3.0.0
Successfully installed activeresource-3.0.0
Successfully installed mime-types-1.16
Successfully installed polyglot-0.3.1
Successfully installed treetop-1.4.8
Successfully installed mail-2.2.6.1
Successfully installed actionmailer-3.0.0
Successfully installed rake-0.8.7
Successfully installed thor-0.14.2
Successfully installed railties-3.0.0
Successfully installed bundler-1.0.0
Successfully installed rails-3.0.0
24 gems installed
Installing ri documentation for activesupport-3.0.0...
Installing ri documentation for builder-2.1.2...
Installing ri documentation for i18n-0.4.1...
Installing ri documentation for activemodel-3.0.0...
Installing ri documentation for rack-1.2.1...
Installing ri documentation for rack-test-0.5.6...
Installing ri documentation for rack-mount-0.6.13...
Installing ri documentation for tzinfo-0.3.23...
Installing ri documentation for abstract-1.0.0...
Installing ri documentation for erubis-2.6.6...
Installing ri documentation for actionpack-3.0.0...
Installing ri documentation for arel-1.0.1...
Installing ri documentation for activerecord-3.0.0...
Installing ri documentation for activeresource-3.0.0...
Installing ri documentation for mime-types-1.16...
Installing ri documentation for polyglot-0.3.1...
Installing ri documentation for treetop-1.4.8...
Installing ri documentation for mail-2.2.6.1...
Installing ri documentation for actionmailer-3.0.0...
Installing ri documentation for rake-0.8.7...
Installing ri documentation for thor-0.14.2...
Installing ri documentation for railties-3.0.0...
Installing ri documentation for bundler-1.0.0...
Installing ri documentation for rails-3.0.0...
```

```
Installing RDoc documentation for activesupport
    -3.0.0...
Installing RDoc documentation for builder -2.1.2...
Installing RDoc documentation for i18n -0.4.1...
Installing RDoc documentation for activemodel -3.0.0...
Installing RDoc documentation for rack -1.2.1...
Installing RDoc documentation for rack-test -0.5.6...
Installing RDoc documentation for rack-mount -0.6.13...
Installing RDoc documentation for tzinfo -0.3.23...
Installing RDoc documentation for abstract -1.0.0...
Installing RDoc documentation for erubis -2.6.6...
Installing RDoc documentation for actionpack -3.0.0...
Installing RDoc documentation for arel -1.0.1...
Installing RDoc documentation for activerecord -3.0.0...
Installing RDoc documentation for activeresource
    -3.0.0...
Installing RDoc documentation for mime-types -1.16...
Installing RDoc documentation for polyglot -0.3.1...
Installing RDoc documentation for treetop -1.4.8...
Installing RDoc documentation for mail -2.2.6.1...
Installing RDoc documentation for actionmailer -3.0.0...
Installing RDoc documentation for rake -0.8.7...
Installing RDoc documentation for thor -0.14.2...
Installing RDoc documentation for railties -3.0.0...
Installing RDoc documentation for bundler -1.0.0...
Installing RDoc documentation for rails -3.0.0...
```

A következő lépésben az adatbáziskezelő Ruby illesztőjét telepítjük. Először az SQLite, majd a MySQL adapterét.

```
root@vb:/home/kovacsg# gem install sqlite3-ruby
Building native extensions.  This could take a while...
Successfully installed sqlite3-ruby-1.3.1
1 gem installed
Installing ri documentation for sqlite3-ruby-1.3.1...
Installing RDoc documentation for sqlite3-ruby-1.3.1...
root@vb:/home/kovacsg# gem install mysql2
Building native extensions.  This could take a while...
Successfully installed mysql2-0.2.4
1 gem installed
Installing ri documentation for mysql2-0.2.4...
Enclosing class/module 'mMysql2' for class Client not
```

```
    known
Enclosing class/module 'mMysql2' for class Result not
    known
Installing RDoc documentation for mysql2-0.2.4...
Enclosing class/module 'mMysql2' for class Client not
    known
Enclosing class/module 'mMysql2' for class Result not
    known
```

Ezután ellenőrizzük, hogy a `rails` parancs elérhető-e. Ha nem, akkor adjuk hozzá a `PATH` környezeti változóhoz, majd ismételjük meg az ellenőrzést.

A `-d` kapcsoló különös jelentőséggel bír számunkra, ezzel adhatjuk meg a használni kívánt adatbáziskezelő típusát, ami lehet `mysql`, `oracle`, `postgresql`, `sqlite3`, `frontbase` vagy `ibm_db`. A gyakorlatok keretében ezek közül a `sqlite`-ot mint alapértelmezett adatbáziskezelőt és a `mysql`-t fogjuk használni.

```
kovacsg@vb:~$ rails
bash: rails: command not found
kovacsg@vb:~$ export PATH=$PATH:/var/lib/gems/1.9.1/bin
    /
kovacsg@vb:~$ rails
Usage:
  rails new APP_PATH [options]

Options:
  -r, [--ruby=PATH]              # Path to the Ruby binary
      of your choice
                                 # Default: /usr/bin/ruby1
                                     .9.1
  -d, [--database=DATABASE]      # Preconfigure for
      selected database (options: mysql/oracle/
      postgresql/sqlite3/frontbase/ibm_db)
                                 # Default: sqlite3
  -b, [--builder=BUILDER]        # Path to an application
      builder (can be a filesystem path or URL)
  -m, [--template=TEMPLATE]      # Path to an application
      template (can be a filesystem path or URL)
       [--dev]                   # Setup the application
           with Gemfile pointing to your Rails checkout
       [--edge]                  # Setup the application
           with Gemfile pointing to Rails repository
```

```
      [--skip-gemfile]            # Don't create a Gemfile
  -O, [--skip-active-record]      # Skip Active Record
      files
  -T, [--skip-test-unit]          # Skip Test::Unit files
  -J, [--skip-prototype]          # Skip Prototype files
  -G, [--skip-git]                # Skip Git ignores and
      keeps

Runtime options:
  -f, [--force]    # Overwrite files that already exist
  -p, [--pretend]  # Run but do not make any changes
  -q, [--quiet]    # Supress status output
  -s, [--skip]     # Skip files that already exist

Rails options:
  -v, [--version]  # Show Rails version number and quit
  -h, [--help]     # Show this help message and quit

Description:
    The 'rails new' command creates a new Rails
       application with a default
    directory structure and configuration at the path
       you specify.

Example:
    rails new ~/Code/Ruby/weblog

    This generates a skeletal Rails installation in ~/
       Code/Ruby/weblog.
    See the README in the newly created application to
       get going.
```

A Rails keretrendszerben a `rails` parancs az az univerzális eszköz mellyel többek között új komponenseket hozhatunk létre és elindíthatjuk a beépített webszervert. Egy új Rails alkalmazást a `rails` parancsnak `new` opciót megadva hozhatunk létre, a második argumentum az alkalmazás neve. A paracsot kiadva a konzolon láthatjuk az automatikusan generált fájlokat. Az egyes könyvtárak értelmezéséről előadáson esett szó, e gyakorlat keretében megnézzük az egyes fájlok szerepét és tartalmát.

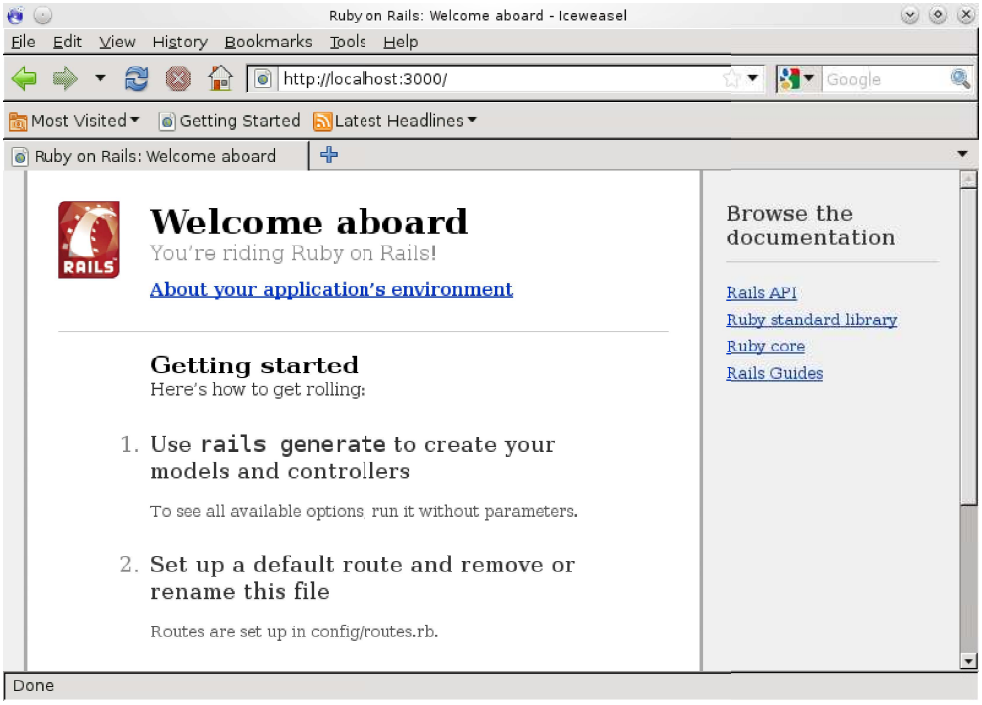
```
kovacsg@vb:~$ rails new feladat
      create
```

```
      create  README
      create  Rakefile
      create  config.ru
      create  .gitignore
      create  Gemfile
      create  app
      create  app/helpers/application_helper.rb
      create  app/views/layouts/application.html.erb
      create  app/controllers/application_controller.rb
      create  app/mailers
      create  app/models
      create  config
      create  config/routes.rb
      create  config/application.rb
      create  config/environment.rb
      create  config/environments
      create  config/environments/development.rb
      create  config/environments/test.rb
      create  config/environments/production.rb
      create  config/initializers
      create  config/initializers/session_store.rb
      create  config/initializers/secret_token.rb
      create  config/initializers/inflections.rb
      create  config/initializers/mime_types.rb
      create  config/initializers/backtrace_silencers.
        rb
      create  config/locales
      create  config/locales/en.yml
      create  config/boot.rb
      create  config/database.yml
      create  db
      create  db/seeds.rb
      create  doc
      create  doc/README_FOR_APP
      create  lib
      create  lib/tasks
      create  lib/tasks/.gitkeep
      create  log
      create  log/server.log
      create  log/production.log
      create  log/development.log
```

```
      create  log/test.log
      create  public
      create  public/favicon.ico
      create  public/robots.txt
      create  public/500.html
      create  public/404.html
      create  public/index.html
      create  public/422.html
      create  public/images
      create  public/images/rails.png
      create  public/stylesheets
      create  public/stylesheets/.gitkeep
      create  public/javascripts
      create  public/javascripts/dragdrop.js
      create  public/javascripts/effects.js
      create  public/javascripts/prototype.js
      create  public/javascripts/application.js
      create  public/javascripts/controls.js
      create  public/javascripts/rails.js
      create  script
      create  script/rails
      create  test
      create  test/performance/browsing_test.rb
      create  test/test_helper.rb
      create  test/fixtures
      create  test/functional
      create  test/unit
      create  test/integration
      create  tmp
      create  tmp/sessions
      create  tmp/sockets
      create  tmp/cache
      create  tmp/pids
      create  vendor/plugins
      create  vendor/plugins/.gitkeep
```

A beépített webszervert a `rails server` opciójával indíthatjuk el a Rail alkalmazásunk gyökérkönyvtárából. Ha a szervert háttéralkalmazásként kívánjuk futtatni, akkor a `-d` kapcsolót is hozzá kell fűznünk a parancssorhoz. Ezután a szerver elérhető a `http://localhost:3000` webcímen, lásd 1. ábra.

```
kovacsg@vb:~/feladat$ rails server
```



1. ábra. A beágyazott webszerver elérése

```
=> Booting WEBrick
=> Rails 3.0.0 application starting in development on
   http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
[2010-10-01 14:44:27] INFO  WEBrick 1.3.1
[2010-10-01 14:44:27] INFO  ruby 1.9.2 (2010-08-18) [
   i486-linux]
[2010-10-01 14:44:27] INFO  WEBrick::HTTPServer#start:
   pid=4850 port=3000
^C[2010-10-01 14:45:24] INFO  going to shutdown ...
```

```
[2010-10-01 14:45:24] INFO  WEBrick::HTTPServer#start
    done.
Exiting
kovacsg@vb:~/feladat$ rails server -d
=> Booting WEBrick
=> Rails 3.0.0 application starting in development on
    http://0.0.0.0:3000
```

Két fontos konfigurációs fájl érdemel most említést: a adatbázis hozzáférés konfigurációjára szolgáló `database.yml` és az URL leképezéseket definiáló `routes.rb`.

Mivel a Rails alkalmazás létrehozásakor nem adtunk meg, hogy melyik adatbáziskezelőt kívánjuk használni, ezért az alapértelmezett SQLite konfigurációs paraméterei jelennek meg a `database.yml`-ben mindhárom definiált környezethez, vagyis a fejlesztési (`development`), tesztelési (`test`) és éles (`production`). Az SQLite az adatait fájlokban tárolja, amelyek a Rails alkalmazás `db` könyvtárában találhatók, és a környezet nevével azonosítottak e fájl szerint.

```
# SQLite version 3.x
#   gem install sqlite3-ruby (not necessary on OS X
    Leopard)
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000

# Warning: The database defined as "test" will be
    erased and
# re-generated from your development database when you
    run "rake".
# Do not set this db to the same as development or
    production.
test:
  adapter: sqlite3
  database: db/test.sqlite3
  pool: 5
  timeout: 5000

production:
  adapter: sqlite3
```

```
  database: db/production.sqlite3
  pool: 5
  timeout: 5000
```

A másik leírónk azt adja meg, hogy milyen struktúrájú legyen az URL, amivel elérjük a Rails alkalmazásunk egyes funkcióit. Az alábbi kódrészlet a legláltalánosabb beállítást tartalmazza. A webszerver IP címe után a Controller osztály neve (`:controller`), majd a Controller osztály egy metódusa (`:action`), majd egy adatbázis azonosító (`:id`), és végül formázási útmutató következik. Az utolsó három megadása opcionális.

```
Feladat::Application.routes.draw do
  match ':controller(/:action(/:id(.:format)))'
end
```

Nézzük meg, hogy miként tudunk dinamikus tartalmat létrehozni Railsszel. A példák a [1] könyvből valók.

Hozzunk létre egy új controllert a `rails` parancs `generate` opciójával. A második argumentum (`controller`) azt mondja meg, hogy egy új controllert hozunk létre, a harmadik a controller nevét. A negyedik és minden további paraméter a controllerben definiál akciókat. E parancs négy Ruby forrásfájlt és egy könyvárat hoz létre az akcióknak megfelelő weboldalak, view-k számára. A controller nevének megfelelő controller osztályt (`say_controller.rb`), helper osztályt, illetve ezek egységtesztjéhez használható osztályokat.

```
kovacsg@vb:~/feladat$ rails generate controller say
      create  app/controllers/say_controller.rb
      invoke  erb
      create    app/views/say
      invoke  test_unit
      create    test/functional/say_controller_test.rb
      invoke  helper
      create    app/helpers/say_helper.rb
      invoke    test_unit
      create      test/unit/helpers/say_helper_test.rb
```

Bármilyen tartalom megjelenítéséhez a `routes.rb` alapján a view könytárban kell elhelyeznünk az akciónak megfelelő néven egy beágyazott Ruby kódot tartalmazó HTML fájlt (`.rhtml` vagy `.html.erb`).

Az alkalmazás keretét az `app/views/layouts/application.html.erb` fájl definiálja, amely a HTML dokumentum törzs helyén egy beágyazott `yield` parancsot tartalmaz, amely átadja a vezérlést az akció HTML-ének.

Ezalapján nézzük meg a szokásos Hello, world alkalmazást ezúttal Railsben. Az `app/views/say` könyvtárban létrehozunk egy `hello.html.erb` nevű view-t, amely a `say` controller `hello` akciójához kötődik. Az eredményt a `http://localhost:3000/say/hello` linken ellenőrizhetjük.

```
<h1>Hello, world!</h1>
```

Ez dinamikussá tehetjük az aktuális idő kiírásával.

```
<h1>Hello, world!</h1>
<%= Time.now %>
```

Mivel a View-ba nem illik logikát rakni, csak a megjelenítendő értéket, ezért áttesszük az idő lekérdezését a Controllerbe, annak is az akciónak megfelelő metódusába, a `hello`-ba

```
class SayController < ApplicationController
  def hello
    @time=Time.now
  end
end
```

a View-ban hivatkozhatunk a Controller példányváltozóira.

```
<h1>Hello, world!</h1>
<%= @time %>
```

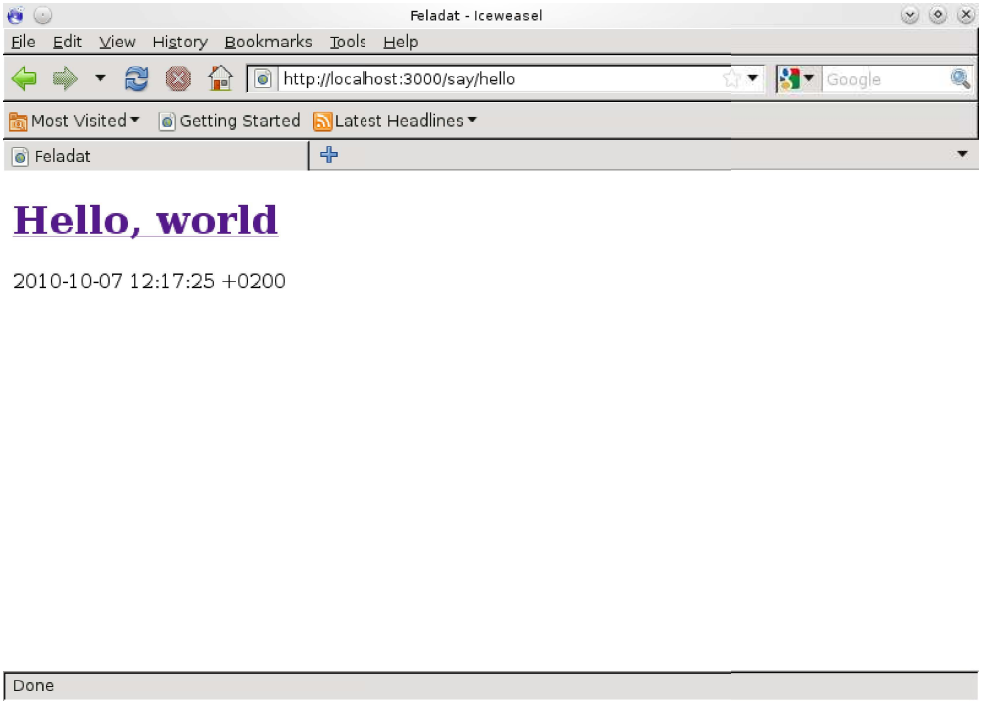
Az oldalak akciói átadhatják egymásnak a vezérlést, ehhez linkeket definálunk a `link_to` metódussal. Az alábbi példában a link neve `"Hello, world"` lesz, a kapcsolódó akció pedig `helloworld` névre hallgat. Ez utóbbihoz definiálnunk kell egy új metódust a Controller osztályban, és egy a View-t megvalósító HTML oldalt.

```
<h1><%= link_to "Hello, world", :action => "helloworld"
    %></h1>
<%= @time %>
```

A controllernek és akciónak megfelelő megjelenített oldalt a 2. ábra mutatja.

Az oldal forrását megnézve felismerjük benne a layout által nyújtott keretet és a View beágyazott kódját.

```
<!DOCTYPE html>
<html>
<head>
  <title>Feladat</title>
```



2. ábra. A hello akció megjelenítve

```
  <script src="/javascripts/prototype.js?1285936196" type="text/javascript">
      </script>
<script src="/javascripts/effects.js?1285936196" type="text/javascript"></
    script>
<script src="/javascripts/dragdrop.js?1285936196" type="text/javascript"></
    script>
<script src="/javascripts/controls.js?1285936196" type="text/javascript"></
    script>
<script src="/javascripts/rails.js?1285936196" type="text/javascript"></
    script>
<script src="/javascripts/application.js?1285936196" type="text/javascript">
    </script>
  <meta name="csrf-param" content="authenticity_token"/>
<meta name="csrf-token" content="/fH1YnsUuVUK/DVrneLqV4TYJx+bMfB2L27AlccAUWg
    ="/>
</head>
```

```
<body>

<h1><a href="/say/helloworld">Hello, world</a></h1>
2010-10-07 12:17:25 +0200


</body>
</html>
```

A Rails MVC filozófiájának harmadik eleme a modell, amelyet szintén a `rails` parancs `generate` opciójával hozhatunk létre. A harmadik argumentum a modell oszlály neve, amely a konvenció alapján egy egyes számban megadó és a szavakat `_` szimbólummal összefűző string. Ennek többes számú változatával jön létre az az adatbázisban egy tábla. A parancs kiadása négy fájlt hoz létre: egy adatbázis migrációs Ruby szkriptet, egy `ActiveRecord::Base` leszármazottat a modell osztályok közé, és két teszt szkriptet.

```
kovacsg@vb:~/feladat$ rails generate model submission
      invoke  active_record
      create    db/migrate/20101001131129
          _create_submissions.rb
      create    app/models/submission.rb
      invoke    test_unit
      create      test/unit/submission_test.rb
      create      test/fixtures/submissions.yml
kovacsg@vb:~/feladat$
```

Az adatbázis migrációs szkriptben az adatmodell változtatásait adjuk meg. A modell létrehozása után egy olyan táblát hozna létre az alapértelmezett szkript, amelyben egy azonosító és két időpecsét oszlop szerepelne. A feladatunk az, hogy ezt kiegészítsük az általunk tárolni kívánt adatokkal. A tábla neve `submissions`, ehhez hozzáadunk két egész szám típusú és három string típusú attribútumot.

```
class CreateSubmissions < ActiveRecord::Migration
  def self.up
    create_table :submissions do |t|
      t.integer :sorszam
      t.string :fajlnev
      t.string :idopont
      t.string :komment
      t.integer :statusz
      t.timestamps
    end
```

```
  end

  def self.down
    drop_table :submissions
  end
end
```

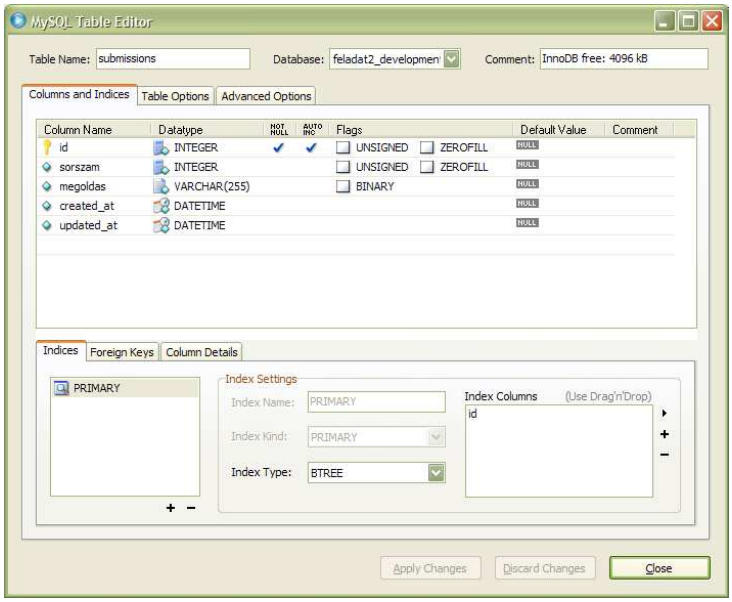
Ha a Rails alkalmazás létrehozásak megadtuk volna a mysql adatbázis opciót, akkor először létre kell hoznunk a megfelelő táblát, majd utána elvégezhetjük a tábla struktúrájának módosítását. Ezekben a `rake` parancs nyújt segítséget. A `rake db:create` létrehozza a táblákat, a `rake db:migrate` módotítja a táblák struktúráját.

```
kovacsg@vb:~/feladat$ rake db:create
(in /home/kovacsg/feladat)
kovacsg@vb:~/feladat$ rails generate model submission
      invoke  active_record
      create    db/migrate/20101001132515
          _create_submissions.rb
      create    app/models/submission.rb
      invoke    test_unit
      create      test/unit/submission_test.rb
      create      test/fixtures/submissions.yml
kovacsg@vb:~/feladat$ rake db:migrate
(in /home/kovacsg/feladat)
==  CreateSubmissions: migrating
    ==============================================
-- create_table(:submissions)
   -> 0.0083s
==  CreateSubmissions: migrated (0.0085s)
    =============================================
```

A gyakorlaton a mysql illesztővel létrehozott Rails alkalmazás a migráció után a 3. ábrán látható táblát hozta létre.

Az adatokon végzett manipulációt a `rails console` paranccsal előhívható interaktív Ruby konzolon (`irb`) illusztráljuk. A `Submission` osztály konstruktorának argumentumai között a tábla attribútumaira a migrációs szkriptben megadott szimbólumokkal hivatkozhatunk, és rendelhetünk hozzá értéket. A nem inicializált attribútumok `null` értéket vesznek fel.

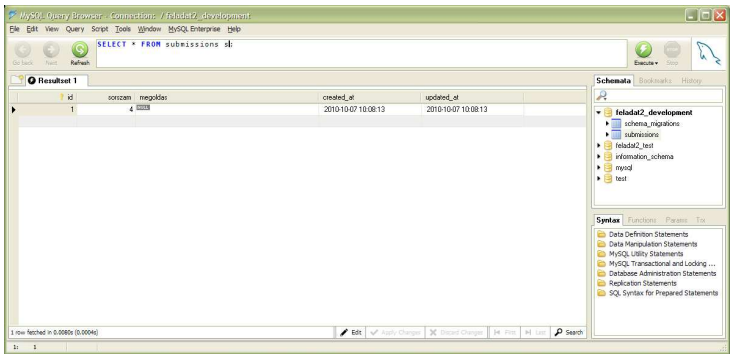
```
kovacsg@vb:~/feladat/db$ rails console
Loading development environment (Rails 3.0.0)
irb(main):003:0> s = Submission.new(:sorszam => "4")
```



3. ábra. A migráció által létrehozott tábla

```
=> #<Submission id: nil, sorszam: 4, fajlnev: nil,
    idopont: nil, komment: nil, statusz: nil, created_at
    : nil, updated_at: nil>
irb(main):004:0> s.save
=> true
irb(main):005:0>
```

A `save` metódussal elmentettük a létrehozott példányt az adatbázisba, amelyet a 4. ábra erősít meg.



4. ábra. Az elmetett rekord

Hivatkozások

[1] David Heinemeier Hansson et al. Sam Ruby, Dave Thomas. *Agile Web Development with Rails*, volume Third Edition. The Pragmatic Bookshelf, 2009 Mar.