Open Source Frameworks for Rapid Application Development

Marek Krętowski
Krzysztof Bandurski, Tomasz Łukaszuk, Tomasz Rybak

Software Departament
Faculty of Computer Science
Bialystok University of Technology

m.kretowski@pb.edu.pl
k.bandurski@pb.edu.pl, t.lukaszuk@pb.edu.pl, t.rybak@pb.edu.pl

Lecture topic

Configurations and conventions in Ruby on Rails

Configurations and conventions in Ruby on Rails: Table of content

Introduction

Introduction I

Ruby on Rails
Rails
RoR

- open source web application framework for the Ruby programming language
- intended to be used with an Agile development methodology
- extracted by David Heinemeier Hansson (Basecamp, 37signals)
- first released as open source in July 2004 (shared commit rights from February 2005)

Introduction II

- in August 2006 Apple announced that it would ship Ruby on Rails with Mac OS X v10.5 "Leopard", which was released in October 2007
- uses the Model-View-Controller (MVC) architecture pattern
- includes scaffolding that can automatically construct some of the models and views needed for a basic website
- includes WEBrick - a simple ruby web server
- includes Rake - a build system
- extensive use of the JavaScript libraries Prototype and Script.aculo.us for Ajax
- has support for web services
- by default offers both HTML and XML as output formats (since version 2.0)
- stable release 3.2.2 / March 1, 2012

Installation

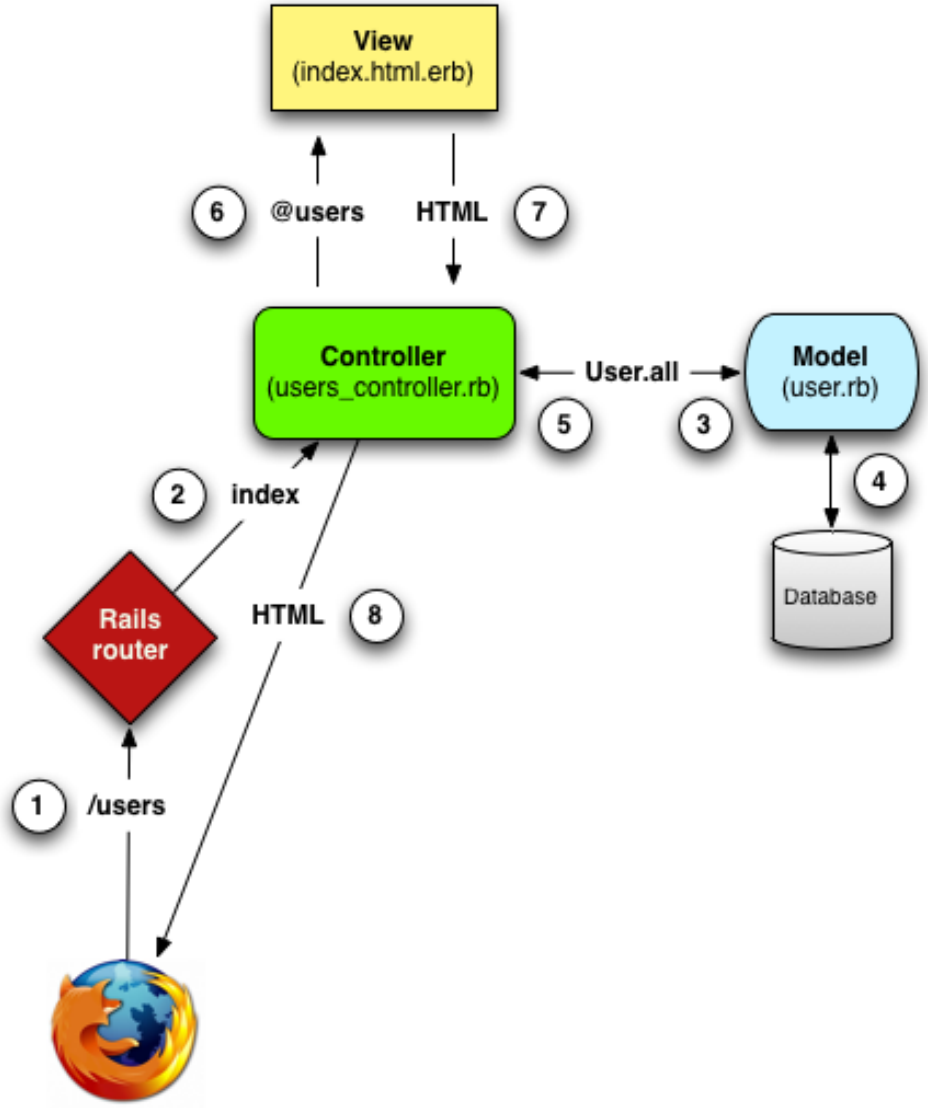
- install Ruby (e.g. version 1.9.2)
  ruby -v
  apt-get install ruby
- install RubyGems
  download RubyGems (http://rubyforge.org/frs/?group_id=126)
  ruby setup.rb
  gem update --system
- install Rails
  gem install rails [-v 3.2.2]

Sample application

```
1  $ rails new sample_app
2  $ cd sample_app
3  $ ruby script/generate scaffold Post name:string title:string content:text
4  $ rake db:migrate
5  $ rails server
6  # => http://localhost:3000/posts
7
8  $ rails generate controller home index
9  $ mcedit app/views/home/index.html.erb   # your own home page
10 $ rm public/index.html
11 $ mcedit config/routes.rb   # root :to => "home#index"
```

Framework structure

The MVC Architecture



Components of Rails

- Action Pack
  Action Controller - manages the controllers
  Action View - manages the views
  Action Dispatch - handles routing
- Action Mailer - framework for building e-mail services
- Active Model - interface between the Action Pack gem services and Object Relationship Mapping gems
- Active Record - the base for the models
- Active Resource - framework for web services
- Active Support - collection of utility classes and standard Ruby library extensions
- Railties - core Rails code

REST

- Representational State Transfer
- RESTful architecture
- Using resource identifiers such as URLs to represent resources
- Transferring representations of the state of that resource between system components
- e.g. DELETE /photos/17

Structure of application

Content of main Rails application folder I

File/Folder	Purpose
Gemfile	This file allows you to specify what gem dependencies are needed for your Rails application.
README	This is a brief instruction manual for your application.
Rakefile	This file contains batch jobs that can be run from the terminal.
app/	Contains the controllers, models, and views for your application. You'll focus on this folder during Rails application development.
config/	Configure your application's runtime rules, routes, database, ...
config.ru	Rack configuration for Rack based servers used to start the application.
db/	Shows your current database schema and the database migrations.
doc/	In-depth documentation for your application.
lib/	Extended modules for your application.
log/	Application log files.

Content of main Rails application folder II

File/Folder	Purpose
public/	The only folder seen to the world as-is. This is where your images, javascript, stylesheets (CSS), and other static files go.
script/	Scripts provided by Rails to do recurring tasks.
test/	Unit tests, fixtures, and other test apparatus.
tmp/	Temporary files
vendor/	A place for third-party code.

Model-View-Controller folders

Folder	Purpose
db/migrate	migrations defining i.a. database structure
app/models	files with classes corresponding to tables in the database
app/views	html based files definig the application user interface
app/controllers	files with classes responsible for communication between models and views

MVC - Model

> rails generate model Post name:string title:string content:text

db/migrate/20100324124235_create_posts.rb

```
1  class CreatePosts < ActiveRecord::Migration
2     def self.up
3        create_table :posts do |t|
4           t.string :name
5           t.string :title
6           t.text :content
7
8           t.timestamps
9        end
10    end
11
12    def self.down
13       drop_table :posts
14    end
15 end
```

app/models/post.rb

```
1  class Post < ActiveRecord::Base
2     validates_presence_of :name, :title
3     validates_length_of :title, :minimum => 5
4     has_many :comments
5  end
```

binding models - has_one, has_many, belongs_to, ...

MVC - Controller

> rails generate controller posts index show edit update

app/controllers/posts_controller.rb

```
1  class PostsController < ApplicationController
2     def index
3        @posts = Post.find(:all)
4        respond_to do |format|
5           format.html # index.html.erb
6           format.xml  { render :xml => @posts }
7        end
8     end
9
10    def show
11       # ...
12    end
13    def edit
14       # ...
15    end
16    def update
17       # ...
18    end
19    ...
20 end
```

MVC - View

app/views/posts/index.html.erb

```
1  <h1>Listing posts</h1>
2  <table>
3    <tr>
4      <th>Name</th>
5      <th>Title</th>
6      <th>Content</th>
7    </tr>
8  <% for post in @posts %>
9    <tr>
10     <td><%=h post.name %></td>
11     <td><%=h post.title %></td>
12     <td><%=h post.content %></td>
13     <td><%= link_to 'Show', post %></td>
14     <td><%= link_to 'Edit', edit_post_path(post) %></td>
15     <td><%= link_to 'Destroy', post, :confirm => 'Are you sure?', :method => :dele
16   </tr>
17 <% end %>
18 </table>
19 <br />
20 <%= link_to 'New post', new_post_path %>
```

Application settings

Configuring a Database

- file `config/database.yml`
- sections for three different environments: development, test, production
- a default database configuration using SQLite3
- Rails also supports MySQL and PostgreSQL ”out of the box”, and has plugins for many database systems
- creating the database: `rake db:create`

```
1  development:
2    adapter: sqlite3
3    database: db/development.sqlite3
4    pool: 5
5    timeout: 5000
```

Locations for Initialization Code

- application.rb
  RAILS_ROOT/config/application.rb
- Environment-specific Configuration Files
  RAILS_ROOT/config/environments/production.rb
  RAILS_ROOT/config/environments/development.rb
  RAILS_ROOT/config/environments/test.rb
- Initializers (load_application_initializers)
  RAILS_ROOT/config/initializers/
- After-Initializers

Configuring Rails Components

- configuration files: config/application.rb, config/environments/production.rb|development.rb|test.rb
- `config` object
- `config.param_name = value` - pass settings to Rails itself
- `config.component_name.param_name = value` - pass settings to individual Rails components
- `http://guides.rubyonrails.org/configuring.html` - about config parameters

```
1  config.filter_parameters += [:password]
2
3  config.active_record.timestamped_migrations = false
```

Using Initializers

- an initializer is any file of ruby code stored under `/config/initializers`
- loads after the framework, gems and plugins in your application
- used to hold configuration settings that should be made after all of the frameworks and plugins are loaded

Using an After-Initializer

- run after any initializers are loaded
- after_initialize block in any of the Rails configuration files

```
1  config.after_initialize do
2     SomeClass.init
3  end
```

Rails Environment Settings

- environment variables are recognized by various parts of Rails
- ENV["RAILS_ENV"] - production, development, test
- ENV["RAILS_RELATIVE_URL_ROOT"]
- ENV["RAILS_ASSET_ID"]
- ENV["RAILS_CACHE_ID"]
- ENV["RAILS_APP_VERSION"]

Rails Command Line Tools

Commands

- rails console
- rails server
- rake
- rails generate
  [controller, integration_test, mailer, migration, model, observer, performance_test, plugin, resource, scaffold, session_migration]
- rails dbconsole
- rails new app_name
- rails plugin
- rails runner
- rails destroy
- rake about

Rake

- Rake is ruby make
- uses a Rakefile and .rake files to build up a list of tasks
- rake --tasks
- db: Database
  doc: Documentation
  gems: Ruby gems
  notes: Code note enumeration
  rails: Rails-specific tasks
  test: Rails tests
  time: Timezones
  tmp: Temporary files

References

- http://en.wikipedia.org/wiki/Ruby_on_Rails
- Ruby on Rails guides - http://guides.rubyonrails.org/
- Przewodniki po Ruby on Rails - http://apohllo.pl/guides/index.html
- Rake - http://ruby-on-rails.pl/szkola/2007/09/02/tutorial-ruby-on-rails-rake/