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

URLs mapping in Ruby on Rails

URLs mapping in Ruby on Rails: Table of content

Introduction

Introduction

- HTTP (Hypertext Transfer Protocol)
  a request-response standard of client-server computing, client submits HTTP requests, server performs the requested actions and returns the reply
- HTTP request methods
  HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT, PATH
- standard URLs
  http://host.com/resource.php?param=5
- URLs mapping
  connecting URLs to code
  generating URLs from code

The dual purpose of routing

- connecting URLs to code

```
1  GET /patients/17
```

- generating URLs from code

```
1  @patient = Patient.find(17)
2
3  <%= link_to "Patient Record", patient_path(@patient) %>
4
5  # http://example.com/patients/17
```

Basis of Rails routing

routes.rb file I

- two components to routing in Rails

  the routing engine and the file config/routes.rb

- format of routes.rb

  one big block sent to AppName::Application.routes.draw,

  includes comments and the lines of code defining a route in your application

```
1  AppName::Application.routes.draw do
2     # The priority is based upon order of creation:
3     # first created -> highest priority.
4
5     # Sample of regular route:
6     #   match 'products/:id' => 'catalog#view'
7     ...
8     # Sample resource route (maps HTTP verbs to controller actions automatical
9     #   resources :products
10    ...
11 end
```

routes.rb file II

- types of routes.rb content
  RESTful (Resource) Routes
  Named Routes

  Regular Routes
- routes.rb processing
  file is processed from top to bottom
  the request will be dispatched to the first matching route

  if there is no matching route, then Rails returns HTTP status 404

Types of Rails route

- Resource Routes

```
1  resources :products
2  resources :products do
3      resources :comments, :sales
4      resource :seller
5  end
```

- Regular Routes

```
1  match 'products/:id' => 'catalog#view'
```

- Named Routes

```
1  match 'products/:id/purchase' => 'catalog#purchase', :as => :purchase
```

- Root of site routed with "root"

```
1  root :to => "welcome#index"
```

Resource Routing

RESTful introduction

- the current standard for routing in Rails
- the foundation of RESTful routing is generally considered to be Roy Fielding's doctoral thesis, "Architectural Styles and the Design of Network-based Software Architectures"
- REST, an acronym for Representational State Transfer
- two main principles of REST
  - Using resource identifiers (URLs) to represent resources
  - Transferring representations of the state of that resource between system components

```
1   DELETE /photos/17
```

RESTful - http verbs and actions

HTTP verbs - controller actions - CRUD operations in a database

```
1   resources :photos
```

http verb	URL	controller	action	used for
GET	/photos	Photos	index	display a list of all photos
GET	/photos/new	Photos	new	return an HTML form for creating a new photo
POST	/photos	Photos	create	create a new photo
GET	/photos/1	Photos	show	display a specific photo
GET	/photos/1/edit	Photos	edit	return an HTML form for editing a photo
PUT	/photos/1	Photos	update	update a specific photo
DELETE	/photos/1	Photos	destroy	delete a specific photo

URLs and paths

helpers	actions
`photos_url`, `photos_path`	index, create
`new_photo_url`, `new_photo_path`	new
`edit_photo_url`, `edit_photo_path`	edit
`photo_url`, `photo_path`	show, update, destroy

- `_url` helper generates a string containing the entire URL
- `_path` helper generates a string containing the relative path from the root of the application

```
1  <%= link_to "List of photos", photos_url %>
2
3  photos_url      # => "http://www.example.com/photos"
4  photos_path     # => "/photos"
```

Customizing Resourceful Routes I

- :controller (specifying a controller to use)

```
1  resources :photos, :controller => "images"
```

- :constraints (specifying constraints)

```
1  resources :photos, :constraints => {:id => /[A-Z][A-Z][0-9]+/}
2  # photos/RR27
3
4  constraints(:id => /[A-Z][A-Z][0-9]+/) do
5      resources :photos
6      resources :accounts
7  end
```

- :as (overriding the named helpers)

```
1  resources :photos, :as => "images"
2  # /images/1/edit
```

Customizing Resourceful Routes II

- :path_names (overriding the new and edit segments)

```
1  map.resources :photos, :path_names => {:new=>'make', :edit=>'change'}
2  # /photos/make
3  # /photos/1/change
```

- :as (prefixing the named route helpers)

```
1  scope "admin" do
2       resources :photos, :as => "admin_photos"
3  end
4  resources :photos
5  #provide route helpers such as admin_photos_path, new_admin_photo_path etc.
```

- :only, :except (restricting the routes created)

```
1  resources :photos, :only => [:index, :show]
2  resources :photos, :except => :destroy
```

Customizing Resourceful Routes III

- :path (translated paths)

```
1  scope(:path_names => { :new => "neu", :edit => "bearbeiten" }) do
2       resources :categories, :path => "kategorien"
3  end
4  #/kategorien/neu
5  #/kategorien/:id/bearbeiten
```

Defining Multiple Resources at the Same Time

```
1  resources :photos, :books, :videos
2
3  #This works exactly the same as
4
5  resources :photos
6  resources :books
7  resources :videos
```

Singular Resources

Singular resource - a resource that clients always look up without referencing an ID

```
1   resource :geocoder
```

Verb	Path	action	used for
GET	/geocoder/new	new	return an HTML form for creating the geocoder
POST	/geocoder	create	create the new geocoder
GET	/geocoder	show	display the one and only geocoder resource
GET	/geocoder/edit	edit	return an HTML form for editing the geocoder
PUT	/geocoder	update	update the one and only geocoder resource
DELETE	/geocoder	destroy	delete the geocoder resource

Controller Namespaces and Routing

organize groups of controllers under a namespace, e.g. administrative controllers under an `Admin::` namespace, place these controllers under the `app/controllers/admin` directory

```
1  namespace "admin" do
2     resources :posts, :comments
3  end
```

Verb	Path	action	helper
GET	/admin/posts	index	admin_posts_path
GET	/admin/posts/new	new	new_admin_posts_path
POST	/admin/posts	create	admin_posts_path
GET	/admin/posts/1	show	admin_post_path(id)
GET	/admin/posts/1/edit	edit	edit_admin_post_path(id)
PUT	/admin/posts/1	update	admin_post_path(id)
DELETE	/admin/posts/1	destroy	admin_post_path(id)

Nested Resources

resources that are logically children of other resources

Listing 1: models

```
1  class Magazine < ActiveRecord::Base
2    has_many :ads
3  end
4
5  class Ad < ActiveRecord::Base
6    belongs_to :magazine
7  end
```

Listing 2: routes.rb

```
1  resources :magazines do
2    resources :ads
3  end
4  # /magazines/1/ads
5  # /magazines/1/ads/new
```

Creating Paths and URLs From Objects

In addition to using the routing helpers, Rails can also create paths and URLs from an array of parameters.

Listing 3: routes.rb

```
1  resources :magazines do
2    resources :ads
3  end
```

Listing 4: view file

```
1  <%= link_to "Ad details", magazine_ad_path(@magazine, @ad) %>
2  <%= link_to "Ad details", url_for(@magazine, @ad) %>
3  <%= link_to "Ad details", [@magazine, @ad] %>
4  <%= link_to "Magazine details", @magazine %>
```

Adding More RESTful Actions

- seven routes that RESTful routing creates by default
- you may add additional routes that apply to the collection or individual members of the collection

```
1  resources :photos do
2    member do
3      get 'preview'
4    end
5  end
6  # recognize /photos/1/preview
7  # create the preview_photo_url and preview_photo_path helpers
8
9  resources :photos do
10   collection do
11     get 'search'
12   end
13 end
14 # recognize /photos/search
15 # create the search_photos_url and search_photos_path route helpers
```

Other routing methods

Regular routes

- regular routing is a way to map URLs to controllers and actions
- you must set up each route within your application separately
- you can mix the two styles (RESTful and regular routing) within a single application

```
1  match ':controller/:action/:id'
2  # /photos/show/1
3  match ':controller/:action/:id/:user_id'
4  # /photos/show/1/2
5  match ':controller/:action/:id/with_user/:user_id'
6  # photos/show/1/with_user/2
```

- hash table `params`
- querystring

```
1  # /photos/show/1
2  params[:id] is set to 1
3  # /photos/show/1?user_id=2
4  params[:id] is set to 1 and params[:user_id] is set to 2
```

Regular routes - Defining Defaults

```
1  match 'photos/:id' => 'photos#show'
2  # incoming path of /photos/12 to the show action of PhotosController
3
4  match 'photos/:id' => 'photos#show', :defaults => { :format => 'jpg' }
5  # match photos/12 to the show action of PhotosController
6  # and set params[:format] to "jpg"
```

Naming Routes

You can specify a name for any route.

```
1  match 'exit' => 'sessions#destroy', :as => :logout
2  # create logout_path and logout_url
3  # calling logout_path will return /exit
```

Redirection

You can redirect any path to another path.

```
1  match "/stories" => redirect("/posts")
2  match "/stories/:name" => redirect("/posts/%{name}")
```

The empty route

- empty route - route requests that come in to the root of the web site
- root command

```
1   root :to => 'pages#main'
```

Testing routes

Seeing Existing Routes with rake

```
1  rake routes
```

```
1             users GET  /users          {:controller=>"users",:action=>"index"}
2  formatted_users GET  /users.:format {:controller=>"users",:action=>"index"}
3                  POST /users          {:controller=>"users",:action=>"create"}
4                  POST /users.:format {:controller=>"users",:action=>"create"}
```

References

- Ruby on Rails guides - http://guides.rubyonrails.org/
- Przewodniki po Ruby on Rails - http://apohllo.pl/guides/index.html