REST Support in Rails

Sang Shin
www.javapassion.com
“Learning is fun!”

Topics

- What is REST?
- REST support in Rails
- Format of the response via *response_to*

What is REST?

REST

- REpresentational State Transfer
- Name coined by Roy Fielding in his Ph.D thesis*
- Architectural Style of the Web

* http://ics.uci.edu/~fielding/pubs/dissertation/top.htm

Characteristics of REST

- RESTful services are stateless
  - Each request from client to server must contain all the information necessary to understand the request
- RESTful services have a uniform interface
  - GET, POST, PUT, and DELETE.
- REST-based architectures are built from resources (pieces of information) that are uniquely identified by URIs
  - In a RESTful purchasing system, each purchase order has a unique URI

Characteristics of REST

- In REST system, resources are manipulated through the exchange of representations of the resources
  - For example, a purchase order resource is represented by an XML document.
  - Within a RESTful purchasing system, a purchase order might be updated by usering an XML document containing the changed purchase order to its URI
- REST-based architectures communicate primarily through the transfer of representations of resources
  - State is maintained within a resource representation

REST Support in Rails

Rails Support in Rails

- Rails 1.2 adds REST support through routes facility called *map.resources*
- *map.resources* are defined in *routes.rb*

```
ActionController::Routing::Routes.draw do |map|
  map.resources :users
...
end
```

- The *map.resources* adds seven new routes and four new route helpers
- Seven actions are defined in the controller
- Scaffolding on a model creates codes for RESTful operations

Actions Generated

Method	URL path	Action	Helper
GET	*/users*	index	users_url
POST	*/users*	create	users_url
GET	*/users/new*	new	new_user_url
GET	*/users/1*	show	user_url(:id=>1)
PUT	*/users/1*	update	user_url(:id=>1)
GET	*/users/1/edit*	edit	edit_user_url(:id=>1)
DELETE	*/users/1*	destroy	user_url(:id=>1)

Actions Generated

- *index* - return a list of resources
- *show* - return the resource identified by the params[:id]
- new - construct a new resource, not saved into the table
- *edit* - return the resource identified by the params[:id] in a form suitable for editing
- *create* - create a new resource, saved into the table
- *update* - update the resource identified by the params[:id] with the data associated with the request
- *destroy* - destroy the resource identified by the params[:id]

index action

- *index* action return a list of resources

```
# GET /users
# GET /users.xml
def index
  @users = User.find(:all)

  respond_to do |format|
    format.html # index.html.erb
    format.xml  { render :xml => @users }
  end
end
```

show action

- *show* returns the resource identified by the params[:id] in *show.html.erb*

```
# GET /users/1
# GET /users/1.xml
def show
  @user = User.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.xml  { render :xml => @user }
  end
end
```

new action

- new - construct a new resource, not saved into the table

```
# GET /users/new
# GET /users/new.xml
def new
  @user = User.new

  respond_to do |format|
    format.html # new.html.erb
    format.xml  { render :xml => @user }
  end
end
```

edit action

- *edit* - return the resource identified by the params[:id] in a form suitable for editing

```
# GET /users/1/edit
def edit
  @user = Post.find(params[:id])
end
```

create action

- *create* - create a new resource, then save it into the table

```
# POST /users
# POST /users.xml
def create
  @user = User.new(params[:user])

  respond_to do |format|
    if @user.save
      flash[:notice] = 'User was successfully created.'
      format.html { redirect_to(@user) }
      format.xml  { render :xml => @user, :status => :created, :location
  => @user }
    else
      format.html { render :action => "new" }
      format.xml  { render :xml => @user.errors, :status =>
  :unprocessable_entity }
    end
  end
end
```

update action

- *update* - update the resource identified by the *params[:id]* with the data associated with the request

```
# PUT /users/1
# PUT /users/1.xml
def update
  @user = User.find(params[:id])

  respond_to do |format|
    if @user.update_attributes(params[:user])
      flash[:notice] = 'User was successfully updated.'
      format.html { redirect_to(@user) }
      format.xml  { head :ok }
    else
      format.html { render :action => "edit" }
      format.xml  { render :xml => @user.errors, :status =>
  :unprocessable_entity }
    end
  end
end
```

destroy action

- *destroy* - destroy the resource identified by the params[:id]

```
# DELETE /users/1
# DELETE /users/1.xml
def destroy
  @user = User.find(params[:id])
  @user.destroy

  respond_to do |format|
    format.html { redirect_to(users_url) }
    format.xml  { head :ok }
  end
end
```

Format of the response via respond_to

respond_to Block

- From Rails 1.2, within a *respond_to* block, you can selects format of the response depending on what is being requested by the user
  - HTML (default) - typically for humans
  - XML
  - JSON
  - YAML

How Does a User Request a Format?

- The format request is specified in the *routes.rb*

```
ActionController::Routing::Routes.draw do |map|
...
# Install the default routes as the lowest priority.
  map.connect ':controller/:action/:id'
  map.connect ':controller/:action/:id.:format'
end
```

- The user sends a URL with a format

```
http://localhost:8081/users/1.xml
http://localhost:8081/users/1.json
http://localhost:8081/users/1.ymal
```

How To Handle a Format Request?

- Specify the formats to support via *format.<format-to-support> { ..}* inside *respond_to* block

```
class WeblogController < ActionController::Base
  def index
    @users = Post.find :all
    respond_to do |format|
      format.html
      format.xml { render :xml => @users.to_xml }
      format.rss { render :action => "feed.rxml" }
    end
  end
end
```

How Does a User Request a Format?

- Then Rails framework selects the respond format according to the format request of the URL

```
GET /weblog          # returns HTML from browser
   Accept header
GET /weblog.xml      # returns the XML
GET /weblog.rss      # returns the RSS
```

Adding Custom Actions

Thank you!

Sang Shin
http://www.javapassion.com
“Learning is fun!”