Rails Routing Roundup

David A. Black

Ruby Power and Light, LLC

http://www.rubypal.com

Roadmap

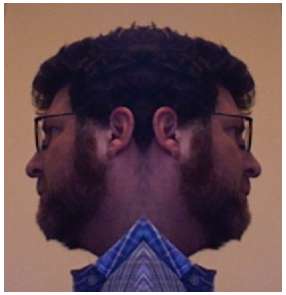
- Basics of routing system
  - Recognition
  - Generation
- Defining routes (routing rules) in `config/routes.rb`
  - Simple routes (routing rules)
  - Named routes
  - RESTful routes
- Some thoughts on when to use what and why

Janus, the God of Routing

- Janus sees incoming URLs, and routes them
- Janus accepts routing specifications and creates a URL that matches them

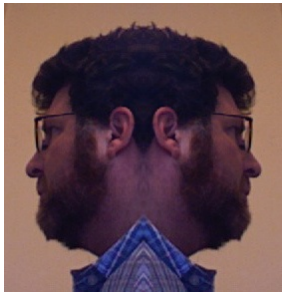
URL *recognition*

`http://www.janusgod.com/events/show/14`



- Probably what "routing" means to most people
- Starts with a URL (or path)
- Figures out what controller to instantiate and what action to execute
- Populates the `params` hash with...other stuff

URL *generation*



```
:controller => "events",
:action => "show",
:id => 14
```

- Recognition, only backwards and in high heels
- More inferential
- Start with what's supposed to happen, and...
- ...figure out what URL is required to make it happen

Routes themselves are *rules*

- Routes live in `config/routes.rb`
- Each route (rule) enables *both* recognition *and* generation
- Rules include:
  - simple routes
  - named routes
  - "resource" (REST-enabling) routes

Simple routes

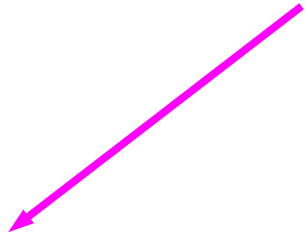
- A call to `map.connect`
- `map` is the "magic pen"
- `map.connect` wants:
  - a pattern-string
  - zero or more segment specifiers

Simple route examples

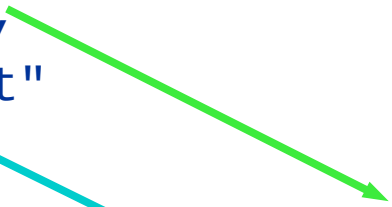
```
map.connect 'help',
            :controller => "main",
            :action     => "assist"

map.connect 'help/:action',
            :controller => "assistance"

map.connect 'help/:topic',
            :controller => "main",
            :action     => "assist"
```

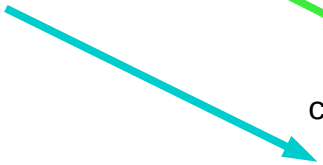
Simple route 1: recognition

```
http://www.janusgod.com/help
```



```
map.connect 'help',
  :controller => "main",
  :action     => "assist"
```





```
class MainController < ApplicationController
  def assist
  end
end
```


Simple route 1: generation

```
<h1>Welcome!</h1>
<p>You can get
<a href="/help">help!</a></p>
```

```
map.connect 'help',
  :controller => "main",
  :action     => "assist"
```

```
<h1>Welcome!</h1>
<p>You can get
<%= link_to "help!",
      :controller => "main",
      :action => "assist" %></p>
```

Simple route 2: recognition

```
http://www.janusgod.com/help/membership
```

```
map.connect 'help/:action',
  :controller => "assistance"
```

```
class AssistanceController < ApplicationController
  def membership
  end
end
```

Simple route 2: generation

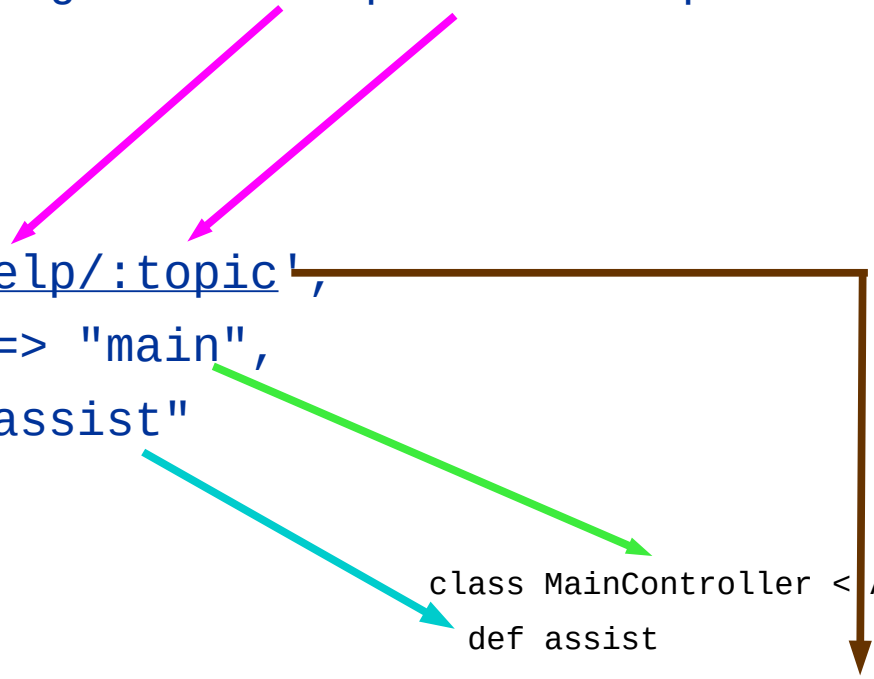
```
<h1>Welcome!</h1>
<p>You can get
<a href="/help/membership">membership help!</a></p>
```

```
map.connect 'help/:action',
  :controller => "assistance"
```

```
<p>You can get
<%= link_to "membership help!",
      :controller => "assistance",
      :action => "membership" %></p>
```

Simple route 3: recognition

```
http://www.janusgod.com/help/membership
```



```
map.connect 'help/:topic',
  :controller => "main",
  :action => "assist"
```

```
class MainController < ApplicationController
  def assist
    @topic = params[:topic]  # "membership"
  end
end
```

Simple route 3: generation

```
<h1>Welcome!</h1>
<p>You can get
<a href="/help/membership">help!</a></p>
```

```
map.connect 'help/:topic',
  :controller => "main",
  :action => "assist"
```

```
<p>You can get
<%= link_to "membership help!",
    :controller => "main",
    :action => "assist",
    :topic => "membership" %></p>
```

Recognition lessons from simple routes

- You always need a `:controller` and an `:action`
- They can be in the pattern string, or given as hash parameters
- Other `:things` in the pattern string
  - get matched positionally to segments of the URL
  - get stashed in the `params` hash
- Static substrings in the pattern string
  - get matched positionally to segments of the URL

Generation lessons from simple routes

- The parameters you provide have to map to the parameters of a particular route
- The matching can involve:
  - `:things` inside the pattern string
  - params provided in the argument hash
- If you don't provide params, they will be guessed, where possible, from the current response environment

Named routes

- A generation thing
- Short-circuit the process of finding a route whose path to generate
- The heart of the REST-friendly routing support in Rails

Named route example

```
<p>You can get <%= link_to "help!", assist_path(:action => "membership") %></p>
```

+

```
map.assist 'assistance/:action', :controller => "assistance"
```

=

```
<p>You can get <a href="/assistance/membership">help!</a></p>
```

RESTful routes

- REST (Representational State Transfer) is a nest of constraints imposed on inter-component communication in a system
- A component is, e.g., a Web browser or server
- Constraints include:
  - client-server model
  - statelessness
  - facilitation of caching
  - uniform interface between components
- HTTP is REST-friendly (though not all Web practices are [e.g., cookies])
- REST supports Web services via HTTP's inherent capabilities

Rails and REST

- A new layer of facilities mostly involving named routes
- Price of entry is low: `map.resources :items`
- Payoff is big:
  - four named routes (and their `_path` equivalents)
    - `item_url`
    - `items_url`
    - `new_item_url`
    - `edit_item_url`
  - pre-coded to trigger seven pre-defined actions
  - *seven?!*

named_route * request_method = action

- "RESTful" routes branch on HTTP request method
- You have to specify the request method, unless it's implied
  - `<% form_tag ... %>` implies POST
  - `<%= link_to ... %>` implies GET
  - `<%= link_to ..., :method => "put" %>` forces PUT
  - *etc.*
- The actions executed are predetermined:
  - `index, show, new, create, edit, update, destroy`

RESTful route examples

```
map.resources :items
```

```
<p><%= link_to "See item details", item_path(@item) %></p>
```

```
<p><a href="/items/14">See item details</a></p>
```

```
GET :controller => "items", :action => "show"
```

RESTful route examples

```
map.resources :items
```

```
<p><%= link_to "Edit item", edit_item_path(@item) %></p>
```

```
<p><a href="/items/14;edit">Edit item</a></p>
```

```
GET :controller => "items", :action => "edit"
```

RESTful route examples

```
map.resources :items
```

```
<% form_for "item", :url => items_path do %>
```

```
<form action="/items" method="post">
```

```
POST :controller => "items", :action => "create"
```

When to use `map.resources`

- When you want to converge on CRUD-like action names
- If you need or want to play in the REST ballpark
  - stable URLs
  - retrieval and manipulation of the same resource with the same URL
  - harnessing what's already in HTTP instead of rolling your own
- If you're curious about what all the fuss has been for the past year!

Other considerations

- `map.resources` is not REST
  - It's a way to facilitate REST-friendly development
- A "resource" in REST is not a database table, not an ActiveRecord class, not a controller
  - Resources are high-level and conceptual
  - Your job is to provide *representations* of them
  - REST *per se* doesn't care how you do it
- HTTP is *already* doing REST-like things (that's the point)
  - GET /help doesn't need seven actions

Conclusion

- REST in Rails helps you:
  - focus on CRUD
  - stop wasting time making up names for actions
  - switch over quickly to named routes
- Therefore:
  - Learn it
  - Judge where and when you need it
  - If it's tying knots rather than untying them, re-examine
  - RESTful Rails is mostly about named routes
    - So named routes must be good!
    - Don't neglect your named route skills

Further, ummm, resources

- Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000 (`http://www.ics.uci.edu/~fielding/pubs/dissertation`)
- `actionpack/lib/action_controller/{routing,resources}.rb`
- Rails blogs, in particular Jamis Buck's blog: `http://weblog.jamisbuck.org`
- Forthcoming from Pearson: *Rails Routing Roundup*, a "Short Cut" PDF book by David A. Black
- Ruby/Rails users groups, books, IRC channels, mailing lists, conferences... WE'VE GOT IT GOOD SO ENJOY IT!

Rails Routing Roundup

David A. Black

Ruby Power and Light, LLC

http://www.rubypal.com