

Resource Routes (in config/routes.rb)

Simple

```
map.resources :users, :sessions
```

Nested

```
map.resources :teams do |teams|
  teams.resources :players
end
```

Customized

```
map.resources :articles,
  :collection => { :sort => :put },
  :member => { :deactivate => :delete },
  :new => { :preview => :post },
  :controller => 'articles',
  :singular => 'article',
  :path_prefix => '/book/:book_id',
  :name_prefix => 'book_'
```

Including the default route (`/:controller/:action/:id`) will allow any verb to access any action. This is usually not what you want, so you should probably delete it from `routes.rb`

REST Screencast \$9 at <http://peepcode.com>

85 minutes of RESTful goodness

PeepCode

Standard Methods	Verb	Path	Action	Formats in the URL	Path
<code>plural_path</code>	GET	<code>/teams</code>	index	<code>formatted_plural_path(:xml)</code>	<code>/teams.xml</code>
<code>singular_path(id)</code>	GET	<code>/teams/1</code>	show	<code>formatted_singular_path(id, :rss)</code>	<code>/teams/1.rss</code>
<code>new_singular_path</code>	GET	<code>/teams/new</code>	new	<code>formatted_players_path(@team, :atom)</code>	<code>/teams/1/players.atom</code>
<code>plural_path</code>	POST	<code>/teams</code>	create	<code>formatted_player_path(@team, @player, :js)</code>	<code>/teams/1/players/5.js</code>
<code>edit_singular_path(id)</code>	GET	<code>/teams/1/edit</code>	edit	<code>formatted_player_path(:team_id => 1, :id => 5, :format => :js)</code>	<code>/teams/1/players/5.js</code>
<code>singular_path(id)</code>	PUT	<code>/teams/1</code>	update		
<code>singular_path(id)</code>	DELETE	<code>/teams/1</code>	destroy		

Each method also has a counterpart ending in `_url` that includes the protocol, domain, and port. There is also a `hash_for_` version of each method that returns a hash instead of a string.

Any URL-generating method can take a hash of options instead of bare arguments. This is the only way to generate a URL with extra querystring params.

```
button_to "Destroy", team_path(@team), :confirm => "Are you sure?", :method => :delete
link_to "Destroy", team_path(@team), :confirm => "Are you sure?", :method => :delete
link_to_remote "Destroy", :url => team_path(@team), :confirm => "Are you sure?", :method => :delete
form_for :team, @team, :url => team_path(@team), :html => { :method => :put } do |f| ...
```

Nested Resources	Path	Useful Plugins
<code>players_path(@team)</code>	<code>/teams/:team_id/players</code> <code>/teams/1/players</code>	<ul style="list-style-type: none"> Beast Forum (an app built with RESTful design) RESTful Authentication Plugin Simply Helpful Plugin
<code>player_path(@team, @player)</code>	<code>/teams/:team_id/players/:id</code> <code>/teams/1/players/5</code>	

Nested resources must be defined in `routes.rb`. See above for an example.

Custom Methods	Path	Action	Map Options
<code>sort_tags_path</code>	<code>/tags;sort</code>	sort	<code>:collection => { :sort => :put }</code>
<code>deactivate_tag_path(id)</code>	<code>/tag/1;deactivate</code>	deactivate	<code>:member => { :deactivate => :delete }</code>
<code>preview_new_tag_path</code>	<code>/tags/new;preview</code>	preview	<code>:new => { :preview => :post }</code>
<code>tags_path(book_id)</code>	<code>/book/:book_id/tags</code>	–	<code>:path_prefix => "/book/:book_id"</code>
<code>tag_path(book_id, id)</code>	<code>/book/:book_id/tags/:id</code>	–	(You get this for free with nested resources)
<code>book_tags_path(book_id)</code>	Usually used in a nested block or with a <code>path_prefix</code>	–	<code>:name_prefix => "book_"</code>
<code>book_tag_path(book_id, id)</code>			(Should be used with a <code>path_prefix</code> or in a nested resource declaration)
<code>book_new_tag_path(book_id)</code>			
<code>book_deactivate_tag_path(book_id, id)</code>			

Repeated resource names in `routes.rb` will overwrite previous declarations. Use `:name_prefix` to preserve dynamic method names for multiple declarations of the same resource.

```
respond_to { |wants| wants.all | .text | .html | .js | .ics | .xml | .rss | .atom | .yaml }
```

Add New MIME types (in config/environment.rb)

```
Mime::Type.register "image/jpg", :jpg
Mime::Type.register "application/vnd.visa+xml", :visa
```

Types listed here can be used in a `respond_to` block and as a forced format extension at the end of urls.

Scaffold Resource Generator

```
./script/generate scaffold_resource
Episode
title:string
description:text
program_id:integer
```

Use a singular word for the model/resource name. The other arguments will be used to pre-populate the database migration and fields in view templates.