Nested Resources

July 2012
by Anton

Nested resources

```
resources :pages do
  resources :posts
end
```

Routes and helpers

```
    page_posts GET    /pages/:page_id/posts(.:format)          posts#index
               POST   /pages/:page_id/posts(.:format)          posts#create
 new_page_post GET    /pages/:page_id/posts/new(.:format)      posts#new
edit_page_post GET    /pages/:page_id/posts/:id/edit(.:format) posts#edit
     page_post GET    /pages/:page_id/posts/:id(.:format)      posts#show
               PUT    /pages/:page_id/posts/:id(.:format)      posts#update
               DELETE /pages/:page_id/posts/:id(.:format)      posts#destroy
```

routes.rb

```
resources :pages, :only => :index

resources :pages do
  resources :posts
end
```

- Same controller
- Same action
- Same view

posts_controller.rb

```
class PostsController < ApplicationController
  before_filter :load_page

  def index
    if @page
      @posts = @page.posts.all
    else
      @posts = Post.all
    end
  end

  def show
    ...
  end

  def load_page
    @page = Page.find(params[:page_id]) if params[:page_id]
  end
  ...
```

/posts/index.slim

```
h1 Listing posts
table
  tr
    th Title
    th Content
    th Post type
    - unless @page
      th Page
    th
    th
    th
  - @posts.each do |post|
    tr
      td= post.title
      td= post.content
      td= post.post_type
      - unless @page
        td= post.page.name
      td= link_to 'Show', [post.page, post]
      td= link_to 'Edit', edit_page_post_path(post.page, post)
      td= link_to 'Destroy', [post.page, post], confirm: 'Are you sure?', method: :delete
```

Problems

- 'IF' statements everywhere
- Dirty views

Solution

- Split index view into
- /posts/index.slim
- /posts/page_index.slim

- Same controller
- Same action
- Different views

/posts/page_index.slim

```
h1 Listing posts
table
  tr
    th Title
    th Content
    th Post type
    th
    th
    th
  - @posts.each do |post|
    tr
      td= post.title
      td= post.content
      td= post.post_type
      td= link_to 'Show', [post.page, post]
      td= link_to 'Edit', edit_page_post_path(post.page, post)
      td= link_to 'Destroy', [post.page, post], confirm: 'Are you sure?', method: :delete
```

posts_controller.rb

```
class PostsController < ApplicationController
  before_filter :load_page

  def index
    if @page
      @posts = @page.posts.all
      render 'page_index'
    else
      @posts = Post.all
    end
  end

  def show
    ...
  end

  def new
    ...
  end

  ...
```

Problems

- Introducing new view naming (`page_index`)
- @page specific views can't be reused

Solution

- Since 'show', 'new', 'page_index', 'edit' specific for page, why not to group them under
- /pages/posts
- Note: renaming 'page_index' back to 'index'

```ruby
class PostsController < ApplicationController
  before_filter :load_page

  def index
    if @page
      @posts = @page.posts.all
      render 'pages/posts/index'
    else
      @posts = Post.all
    end
  end

  def show
    @post = @page.posts.find(params[:id])
    render 'pages/posts/show'
  end

  def new
    @post = @page.posts.new
    render 'pages/posts/new'
  end

  def edit
    @post = @page.posts.find(params[:id])
    render 'pages/posts/edit'
  end
```

Problem

- Specifying views manually
- "you have to type it with your hands" - Tim Oxley
- Views and Controller are in different scopes

Solution

- Split controller posts_controller.rb into
- posts_controller.rb
- /pages/posts_controller.rb

- Two controllers
- Same action

- Two views

routes.rb

```
resources :pages, :only => :index

resources :pages, :module => "pages" do
  resources :posts
end
```

/posts_controller.rb

```
class PostsController < ApplicationController

  def index
    @posts = Post.all
  end

end
```

/pages/posts_controller.rb

```ruby
class Pages::PostsController < ApplicationController
  before_filter :load_page

  def index
    @posts = @page.posts.all
  end

  def show
    @post = @page.posts.find(params[:id])
  end

  def new
    @post = @page.posts.new
  end

  ...
end
```

Wins

- Don't need to specify views
  e.g. `pages/posts/show'
- No 'IF's

Problem

- You will probably will have other resources nested under /pages/ and all of them require @page variable in controller

```
resources :pages, :only => :index

resources :pages do
  resources :posts
  resources :images
end
```

Solution

- Inherit application controller for page nested resources

/pages/application_controller.rb

```
class Pages::ApplicationController < ApplicationController
  before_filter :load_page

  private

  def load_page
    @page = Page.find(params[:page_id])
  end

end
```

/pages/posts_controller.rb

```ruby
class Pages::PostsController < Pages::ApplicationController

  def index
    @posts = @page.posts.all
  end

  def show
    @post = @page.posts.find(params[:id])
  end

  def new
    @post = @page.posts.new
  end

  ...
end
```

Problem

- A lot of repetition for a standard controller actions

Solution

- CanCan
  https://github.com/ryanb/cancan/

/pages/posts_controller.rb

```ruby
class Pages::PostsController < Pages::ApplicationController

  load_and_authorize_resource :through => :page

  def index
  end

  def show
  end

  def new
  end

  def edit
  end

  ...
end
```

Summary

- Controllers:
  /posts_controller.rb
  /pages/posts_controller.rb
  /pages/application_controller.rb

- Views
  /posts/index
  /pages/posts/index
  /pages/posts/show
  /pages/posts/new
  /pages/posts/edit
  /pages/posts/_form