Hunter Lang

Web Applications, Unit 10, Report For Topic Exploration

Due 3/26/14

Rails Routing

- Using the `resources` directive on a controller will automatically route the routes: index, new, create, show, edit, update, and destroy to the given controller
    - This expects these actions to be defined in the controller, and will error if they are not
- Routes are prioritized in the order that they are written in the routes file. This is to say that if you had a simple routes.rb like so:

```
resources :pets

get 'pets/new' to: 'pets#create'
```

  Then the route to access the (pets#create) would never get used due to it being matched earlier from the resources directive, and if the route specified by the resources directive did not exist (pets#new), then an error would occur.

- You can group `resources` directives on a single line, much like multiple assignment.
- The second argument of the route directive corresponds to the match function. If you pass this a symbol, then it will map directly to an action on the current controller. If you pass it a string, then it must be of the format "controller#action". Using this, you can cross route different controller actions to other controllers.
- You can do some namespacing on the controller format to create more informative routes, or even just to organize them.

```
namespace :user do
  resources :profile
end
```

  This would create the same routes as a standard resources directive, but with the 'user/' prefix.

  Note: This requires you to scope the controller ruby file by putting `User::` before the controller name.

- While you can do nested routes, they are strongly discouraged past one level deep, as the application will start to match very strange patterns
- The 'only' clause in a resources directive allows you to specify certain resources to create, and also allows you to do shallow nesting on resources so that not all nested objects have access to all the resources of the default directive

```
resources :books do

  resources :memorable_passages, only: [:show]

end

resources :memorable_passages, only: [:index, :new,
      :create, :edit, :update, :destroy]
```

This would cause the show action of the memorable_passages to only be valid inside of a book's action, and nowhere else. This is how shallow nesting works. You only include the parts of the route that you want to nest, to avoid nesting the whole controller.

I used the URL: http://guides.rubyonrails.org/routing.html for most of the report, and partly the rails docs on routing.