Ruby on Rails 3

Week 6

Scaffolding
materials from the book, Agile Web Development with Rails
by Sam Ruby

How-To Setup ActiveScaffold with Rails 3.0 by Volker Hochstein
What is Scaffolding?

- Scaffolding is a way to quickly put an Active Record class online by providing a series of standardized actions for listing, showing, creating, updating, and destroying objects of the class.
- Useful for quick prototyping.
- These standardized actions come with both controller logic and default templates that through introspection already know which fields to display and which input types to use.
A Little Controversial

• DHH’s blog in fifteen minutes was built on scaffolding
• Over time, however, it became less clear whether being used for production code, or intended to educate new Rails developers on best practices
• Eventually, scaffolding was supposed to be educational, illustrating the best practices around RESTful controllers and other Rails conventions
Scaffolding via Generator

• Create an application
  $ rails new myStore -d mysql

• Create a scaffold for the "Product" model
  $ cd myStore
  $ rails generate scaffold Product
      title:string description:text
      image_url:string price:decimal

• The generator creates a bunch of files; the important one is the migration, namely db/migrate/20101206102532_create_products.rb
Refine the "Product" model

• Though, we have already told Rails about the basic data types for the "Product" model, we can refine the table

```ruby
class CreateProducts < ActiveRecord::Migration
  def self.up
    create_table :products do |t|
      t.string :title
      t.text :description
      t.string :image_url
      t.decimal :price, :precision => 8, :scale => 2
      t.timestamps
    end
  end
end
```

• Price now can have eight digits of significance, and two digits after the decimal point, e.g. 123456.78
Apply the Migration

• To get Rails to apply the migration to our development database, we use
  
  $ rake db:migrate

• The products table is added to the database, defined by the development section of the database.yml file
• Let’s try our new application by starting the server
  
  
  $ rails server

• We use the controller name in lowercase (i.e., products) to access the list of products
Modify some Form Field

• Let’s change the number of lines in the description field of the product table in file `myStore/app/views/products/_form.html.erb`

```html
<div class="field">
  <%= f.label :description %><br />
  <%= f.text_area :description, :rows => 6 %>
</div>
```
Seed Data for Testing

• Rails has the ability to import seed data
• Try this by download the file from
• To populate the products table with test data, simply run
  $ rake db:seed
Refine a View Layout

• After we import the data, we need a couple of files for a better product display

• Copy all of the jpeg images into public/images folder in the application

• Copy the depot.css file into public/stylesheets folder in the application
Rails keeps the file that is used to create a standard page environment for the entire application.

This file is `application.html.erb`, a Rails view layout, residing in the `views/layouts` directory.
Inside Application Layout

• Here is an example:
  ```html
  <!DOCTYPE html>
  <html>
    <head>
      <title>Depot</title>
      <%= stylesheet_link_tag :all %>
      <%= javascript_include_tag :defaults %>
      <%= csrf_meta_tag %>
    </head>
    <body>
      <%= yield %>
    </body>
  </html>
  ```
Load Stylesheets

- `stylesheet_link_tag` creates HTML `<link>` tag, loading stylesheets from `public/stylesheets`
- Specific stylesheet name can also be used here
- `<%= stylesheet_link_tag :all %>` is used to load all stylesheets inside `public/stylesheets`
- Brief info about Unobtrusive Javascript (UJS) can be read from [http://www.themodestrubyist.com/2010/02/24/rails-3-ujs-and-csrfr-meta-tags/](http://www.themodestrubyist.com/2010/02/24/rails-3-ujs-and-csrfr-meta-tags/)
Listing products

<table>
<thead>
<tr>
<th>Product</th>
<th>Image</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product 1</td>
<td><img src="image1.jpg" alt="Image" /></td>
<td>Truncated Description 1 (80 characters)</td>
</tr>
<tr>
<td>Product 2</td>
<td><img src="image2.jpg" alt="Image" /></td>
<td>Truncated Description 2 (80 characters)</td>
</tr>
</tbody>
</table>
<td class="list_actions">
  <%= link_to 'Show', product %><br/>
  <%= link_to 'Edit', edit_product_path(product) %><br/>
  <%= link_to 'Destroy', product, :confirm => 'Are you sure?', :method => :delete %>
</td>
</tr>

<% end %>
</table>
</div>

<br />

<%= link_to 'New product', new_product_path %>
Explanations

• The rows in the listing have alternating background colors; the Rails helper method called `cycle` does this by setting the CSS class of each row to either `list_line_even` or `list_line_odd`

• The `truncate` helper displays just the first eighty characters of the description

• Before `truncate` was called, we called `strip_tags` to remove HTML tags from the description
Rollback Database

• you can experiment with rolling back the migration, so your schema will be transported back in time, and the products table will be disappeared

  $ rake db:rollback
Play More with Active Scaffold

• Install ActiveScaffold to the application

$ rails plugin install
https://github.com/vhochstein/active_scaffold.git
$ rails g active_scaffold_setup [prototype| jquery]

• If you are having troubles generating the setup, i.e.,

c:/Ruby192/lib/ruby/1.9.1/net/http.rb:677:in `connect': SSL_connect returned=1 errno=0 state=SSLv3 read server certificate B: certificate verify failed (OpenSSL::SSL::SSLError)

you should add the line,

OpenSSL::SSL::VERIFY_PEER = OpenSSL::SSL::VERIFY_NONE

into the config/application.rb at the location before the line, require 'rails/all'
Play More with Active Scaffold (2)

• Create 2 Active Scaffolds
  
  $ rails g active_scaffold Team name:string position:integer  
  $ rails g active_scaffold Player name:string injured:boolean salary:decimal date_of_birth:date team:references

• Data Migration
  
  $ rake db:migrate

• Create one-to-many relation between the players and the team by editing app/model/team.rb with adding "has_many :players" into the Team class

• Then, test your Scaffolds
Although most Rails models are backed by a database, models can also be ordinary Ruby classes, or Ruby classes that implement a set of interfaces as provided by the Active Model module.

Controller layer. The Controller layer is responsible for handling incoming HTTP requests and providing a suitable response. All versions of rails. 376 versions since July 25, 2009: 6.0.3.2 - June 17, 2020 (6.5 KB). 6.0.3.1 - May 18, 2020 (6.5 KB). 6.0.3 - May 06, 2020 (6.5 KB). 6.0.3.rc1 - May 01, 2020 (6.5 KB). 6.0.2.2 - March 19, 2020 (6.5 KB). 6.0.2.1 - December 18, 2019 (6.5 KB).

RubyGems.org is made possible through a partnership with the greater Ruby community. Fastly provides bandwidth and CDN support, Ruby Central covers infrastructure costs, and Ruby Together funds ongoing development and ops work. Learn more about our sponsors and how they work together. We need your help to fund the developer time that keeps RubyGems.org running smoothly for everyone. Join Ruby Together today.