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

```erb
<%-- 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
  
  http://media.pragprog.com/titles/rails4/code/depot_b/db/seeds.rb

• 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
  • http://media.pragprog.com/titles/rails4/code/depot_b/public/stylesheets/depot.css

• 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/stylesheet`
- Specific stylesheet name can also be used here
- `<%= stylesheet_link_tag :all %>` is used to load all stylesheets inside `public/stylesheet`
- Brief info about Unobtrusive Javascript (UJS) can be read from [http://www.themodestrubyist.com/2010/02/24/rails-3-ujs-and-csrf-meta-tags/](http://www.themodestrubyist.com/2010/02/24/rails-3-ujs-and-csrf-meta-tags/)
<div id="product_list">
<h1>Listing products</h1>
<table>
  <tr class="<%= cycle('list_line_odd', 'list_line_even') %>">
    <td><%= image_tag(product.image_url, :class => 'list_image') %></td>
    <td class="list_description">
      <dl>
        <dt><%= product.title %></dt>
        <dd><%= truncate(strip_tags(product.description), :length => 80) %></dd>
      </dl>
    </td>
  </tr>
  :<br>
</table>
</div>
<table>
  <tr>
    <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>
</table>

<%= 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
• 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
Ruby on Rails makes it much easier and more fun. It includes everything you need to build fantastic applications, and you can learn it with the support of our large, friendly community. Latest version Rails 6.1.4 released June 24, 2021. Released June 24, 2021. You’ve probably already used many of the applications that were built with Ruby on Rails: Basecamp, HEY, GitHub, Shopify, Airbnb, Twitch, SoundCloud, Hulu, Zendesk, Square, Cookpad. Those are just some of the big names, but there are literally hundreds of thousands of applications built with the framework since its release in 2004. Ruby on Rails is open source software, so not only is it free to use, you can also help make it better. More than 5,000 people already have contributed code to Rails. Rails 3.0 requires Ruby 1.8.7 or higher. Support for all of the previous Ruby versions has been dropped officially and you should upgrade as early as possible. Rails 3.0 is also compatible with Ruby 1.9.2. Note that Ruby 1.8.7 p248 and p249 have marshaling bugs that crash Rails 3.0. Ruby Enterprise Edition have these fixed since release 1.8.7-2010.02 though. On the 1.9 front, Ruby 1.9.1 is not usable because it outright segfaults on Rails 3.0, so if you want to use Rails 3 with 1.9.x jump on 1.9.2 for smooth sailing. 1.2 Rails Application object. As part of the groundwork for supporting running multiple Rails applications in the same process, Rails 3 introduces the concept of an Application object. Rails is a web-application framework that includes everything needed to create database-backed web applications according to the Model-View-Controller (MVC) pattern. 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. Ruby on Rails is an open-source web development framework written in Ruby. Ruby on Rails follows the principle of convention over configuration, freeing you from having to re-invent things to stay productive. Use this tag only for Rails 3-specific questions, and also tag those questions [ruby-on-rails]. Learn more}