# HAML and Rake Tasks

Devpoint Labs - Jake Sorce / Dave Jungst

# HAML Intro

Haml (HTML abstraction markup language) is based on one primary principle: markup should be beautiful. It's not just beauty for beauty's sake either; Haml accelerates and simplifies template creation.

# Installing and Using HAML

- gem "haml-rails", "~> 0.9"
- bundle install
- any new view files will be generated with the extension of .haml from now on.
- Any old .erb files will need to be renamed from .erb to .haml and converted
- Pay close attention to INDENTATION!

# ERB to HAML - Basic

```erb
<html>
  <body>
    <h1 class='header'>Users</h1>
    <hr />
    <div id='users'>
      <ul id='users_list'>
        <% @users.each do |user| %>
          <li class='user'><%= user.name %></li>
        <% end %>
      </ul>
    </div>
  </body>
</html>
```

```haml
%html
  %body
    %h1.header Users
    %hr
    #users
      %ul#users_list
        - @users.each do |user|
          %li.user
            = user.name
```

# ERB to HAML - More Attributes Example

```erb
<html>
  <body>
    <h2 class='authors_header center-align blue-text lighten-2'>Authors</h2>
    <% @authors.each do |author| %>
      <div class="author card" data-author-id="#{<%= author.id %>}">
        <%= author.info %>
      </div>
      <span> Author Last Posted At:</span>
      <span class='last_posted_at'><%= author.posts.last.updated_at %></span>
    <% end %>
  </body>
</html>
```

**Examples:**

.author.card{data: {author_id: author.id}}

%a#link.active{href: '/users'} Users

<a id='link' class='active' href='/users'>Users</a>

= link_to 'Users', users_path, class: 'active'

%li.user_item
  = user.name

```haml
%html
  %body
    %h2.authors_header.center-align.blue-text.lighten-2 Authors
    - @authors.each do |author|
      .author.card{"data-author-id" => "#{author.id}"}
        = author.info
      %span Author Last Posted At:
      %span.last_posted_at
        = author.posts.last.updated_at
```

# Rails - Rake Task Intro

Rake utility allows you to create a job/task which uses rails environment. So say, you want to count the votes a user has given to an article and save it somewhere. You write a rake task, in which you can use Rails models and other helpers and get it done without going away from Rails.

# Rake Task Commands

- rails g task migrate users projects
  - creates a new rake task in lib/tasks/ called migrate.rake with placeholder tasks called users and projects
- rails d task migrate
  - removes the migrate.rake file and all code inside of it
- bundle exec rake -T
  - shows all rake tasks for the application

# What Makes a Rake Task a Good Task?

- It has a meaningful and simple description.
- It uses namespace to group similar or related tasks.
- Its file structure follows the namespaces structure.
- It's isolated on a class so we can re use it and test it with ease.
- It displays details about its progress without being too verbose.
- Use a separate log file for each task

# Meaningful and Simple Description

```ruby
task migrate_projects: :environment do
end
```

Bad

Writing a description it's useful because it give us some details without reading the code. Also it's useful when you want to inspect the list of available rake tasks using rake -T. Now we only know that this task imports topics, nothing else.

```ruby
desc 'Migrate projects from legacy database to new database'
task migrate_projects: :environment do
end
```

Good

Adding a good description for the previous task like "Migrate projects from legacy database to new database" give us more details about what the task do or should do.

# Namespacing

Bad

```
# lib/tasks/migrate_projects.rake
desc 'Migrate projects from legacy database to new database'
task migrate_topics: :environment do
  ...
end
# lib/tasks/migrate_users.rake
desc 'Migrate users from legacy database to new database'
task migrate_users: :environment do
  ...
end
# lib/tasks/migrate_profiles.rake
desc 'Migrate profiles from legacy database to new database'
task migrate_questions: :environment do
  ...
end
```

```
# lib/tasks/migrate/topics.rake
namespace :migrate do
  desc 'Migrate projects from legacy database to new database'
  task projects: :environment do
    ...
  end
end
# lib/tasks/migrate/users.rake
namespace :migrate do
  desc 'Migrate users from legacy database to new database'
  task users: :environment do
    ...
  end
end
# lib/tasks/migrate/questions.rake
namespace :migrate do
  desc 'Migrate profiles from legacy database to new database'
  task profiles: :environment do
    ...
  end
end
```

Good

# File Structure Follows Namespace Structure

- BAD
  - # lib/tasks/migrate_projects.rake
  - # lib/tasks/migrate_users.rake
  - # lib/tasks/migrate_profiles.rake

- GOOD
  - # lib/tasks/migrate/projects.rake
  - # lib/tasks/migrate/users.rake
  - # lib/tasks/migrate/profiles.rake

# Isolated on a Class

Bad

```ruby
# lib/tasks/users/recalculate_badges.rake
namespace :users do
  desc 'Recalculates Badges for All Users'
  task recalculate_badges: :environment do
    User.find_each do |user|

      # Grants teacher badge
      if user.answers.with_votes_count_greater_than(5).count >= 1
        user.grant_badge('teacher')
      end

      ...

      # Grants favorite question badge
      user.questions.find_each do |question|
        if question.followers_count >= 25
          user.grant_badge('favorite question') && break
        end
      end

      # Grants stellar question badge
      user.questions.find_each do |question|
        if question.followers_count >= 100
          user.grant_badge('stellar question') && break
        end
      end

    end
  end
end
```

Good

```ruby
# lib/tasks/users/recalculate_badges.rake
namespace :users do
  desc 'Recalculates Badges for All Users'
  task recalculate_badges: :environment do
    User.find_each do |user|

      RecalculateBadges.new(user).all

    end
  end
end
```

# Isolated on a Class - Cont

app/services/recalculate_badges.rb

```ruby
# app/services/recalculate_badges.rb
class RecalculateBadges

  attr_reader :user, :questions, :answers

  def initialize(user)
    @user      = user
    @questions = user.questions
    @answers   = user.answers
  end

  def all
    teacher
    favorite_question
    stellar_question
  end

  def teacher
    ...
    grant_badge('teacher')
  end

  def favorite_question
    question_followers_count_badge(25, 'favorite question')
  end

  def stellar_question
    question_followers_count_badge(100, 'stellar question')
  end

  private

    def grant_badge(badge_name)
      return unless badge_name
      user.grant_badge(badge_name)
    end

    def question_followers_count_badge(followers_count, badge_name)
      ...
      grant(badge_name)
    end

end
```

# Non-Verbose / Clean Progress Output

Bad

```ruby
# lib/tasks/users/recalculate_badges.rake
namespace :users do
  desc 'Recalculates Badges for All Users'
  task recalculate_badges: :environment do
    User.find_each do |user|

      puts "#{user.first_name} #{user.last_name} - #{user.email}"
      RecalculateBadges.new(user).all

    end
  end
end
```

Good

```ruby
# lib/tasks/users/recalculate_badges.rake
namespace :users do
  desc 'Recalculates Badges for All Users'
  task recalculate_badges: :environment do
    users_count = User.count

    User.find_each.with_index do |user, index|
      recaulculate_badges = RecalculateBadges.new(user)

      if recalculate_badges.all
        puts "#{index}/#{users_count} - #{user.email}".green
      else
        puts "#{index}/#{users_count} - #{user.email} - #{recalculate_badges.errors}".red
      end

    end
  end
end
```

# Use a Log File For Each Rake Tasks

```ruby
# lib/tasks/users/recalculate_badges.rake
namespace :users do
  desc 'Recalculates Badges for All Users'
  task recalculate_badges: :environment do
    log = ActiveSupport::Logger.new('log/users_recalculate_badges.log')
    start_time = Time.now
    users_count = User.count

    log.info "Task started at #{start_time}"

    User.find_each.with_index do |user, index|
      recaulculate_badges = RecalculateBadges.new(user)

      if recalculate_badges.all
        log.info "#{index}/#{users_count} - #{user.email}"
      else
        log.info "#{index}/#{users_count} - #{user.email} - #{recalculate_badges.errors}"
      end

    end

    end_time = Time.now
    duration = (start_time - end_time) / 1.minute
    log.info "Task finished at #{end_time} and last #{duration} minutes."
    log.close
  end
end
```

# Use a Log File For Each Rake Tasks - Cont

Having a log file is a MUST when writing rake tasks. This helps you to keep track for every task triggered, consult it every time you want to and share it with anyone easily.

# Resources

- HAML
  - HAML Offical Page - http://haml.info
  - haml-rails gem github - https://github.com/indirect/haml-rails

- Rake
  - Rails Guides - http://guides.rubyonrails.org/v3.2.9/command_line.html#rake
  - Tutorial - http://edelpero.svbtle.com/everything-you-always-wanted-to-know-about-writing-good-rake-tasks-but-were-afraid-to-ask

# Afternoon Project

- **Basic Objectives:**
  - Create Car Dealership Rails Application
  - Starter models you may have more than this:
    - Dealership
    - Salesman
    - Customer
    - Car
  - Figure out all models and associations
  - Write all CRUD actions
  - Write all views in HAML
  - Use Faker and Populator Gems to  fill out your database for testing
  - Style it nice with bootstrap or materalize
- **Bonus Objectives**
  - Use at least 1 react component
  - Have 100% test coverage on models and controllers with RSpec