Topics in Rails:
Frontin', p.1

OR: "You're Working Too Hard"

Ben Vandgrift
ben@vandgrift.com / @bvandgrift
http://ben.vandgrift.com

http://bit.ly/bv-rails-front

Survey Topics

- Rendering
- Form Helpers and Gems (Formtastic, Active Admin)
- HAML, SCSS, and CoffeeScript
- Asset Pipeline
- Caching

The Broad Strokes

- browser makes request
- route to controller
- controller begins processing
  - render :view
- controller finishes processing
- response returned to browser

Basic Rendering

```
- render :nothing => true
- render @something(s)
- render :action_name # in this controller
- render 'path/to/template_name' # in view path
- render :inline => "<%= 'wut' %>" # booooo
- render :json => @object
- render :xml => @object
- render :js => "alert('WUT');" # booooo
```

Options? Plenty.

- `:layout => 'layout_filename' | false`
- `:status => 500 | :forbidden`
- `:xml => @photo, :location => photo_url(@photo)`
- `:action => 'edit'`
- `'books/edit' | :template => 'books/edit(.suffix)*'`
- `:layout => 'layout_filename' | false`
- `:file => '/path/to/some/template.haml',`
  `:content-type => 'application/donuts'`
- mix and match, sort of.

But Really...

- render does a lot of the work for you
- render @object # finds the partial for you
- render @objects # makes good decisions
- remember @object.to_s and @object.to_param
- stop working so hard
  - write good partials, and life is easier
  - let rails do the work

Honorable Mentions

- `head :bad_request # etc`
  - most HTTP headers represented
  - `head :301/2, :location => '/somewhere'`
- `redirect_to :action => :explode`
  - options identical to link_to and url_for
  - can include `:status => :whatever`
  - `:back # does exactly what you expect`

Layouts

- a layout is a template with one or more yield that lives in a special house. THAT'S ALL!
- layouts are inherited unless respecified
- layout "name"/false
  - :except/:only => [:action]

- layout.haml

```
!!!html
%head
  = yield :header
%body
  #content
    = yield
  #sidebar
    - if content_for? :sidebar
      = yield :sidebar
    - else
      = render 'sidebar'
```

- view.haml

```
- content_for :header do
  %title= @kitteh
  = stylesheet_link_tag "extra_stuff"
- end

%h1= @kitteh # calls .to_s, make life easy
= render @kitteh.toys
```

Brief Q/A?

Forms

- ~~form_tag~~ #in case of emergency, otherwise:

```
form_for @object, {:url => {}, :html =>
{}} do |f|
  f.field_type :attribute_name
end
```

- :remote => true # yes, please (another show?)

```
f.fields_for :associated_model do |am|
```

- extra credit: CSRF and authenticity tokens
- but that's all working too hard.

gem 'formtastic'

- ```
  semantic_form_for @kitten do |f|
    f.inputs :breed, :color, :is_bitchy?
    f.actions :submit, :cancel
  end #DONE
  ```
- all those :symbols? unnecessary!
- options a-plenty!
- (!) not so awesome with namespace'd and nested forms #meh
- still working too hard
  ```

gem 'active_admin'

- does much of the heavy lifting for you
- very configurable
- uses formtastic notation
- looks vurry vurry niiice
- http://activeadmin.info/

gem 'haml' # > erb

```
<div id="profile">
  <div class="left column">
    <div id="date"><%= print_date %></div>
    <div id="address"><%= current_user.address %></div>
  </div>
  <div class="right column">
    <div id="email"><%= current_user.email %></div>
    <div id="bio"><%= current_user.bio %></div>
  </div>
</div> <!-- ugly erb -->
```

```
#profile
  .left.column
    #date= print_date
    #address= current_user.address
  .right.column
    #email= current_user.email
    #bio= current_user.bio
```

haml syntax

```
- indentation => block structure
- %tag{:attribute => “value”}
- .div-class-name
- #div-id
```

haml interpolation

```
-  = some_method_with_output(args)
-  - some_method_no_output(args)
-  filters!
   -  :javascript
        alert('yuck!');
   -  :erb
        <%= "don't make me come over there." %>
   -  :coffeescript (w/gem 'coffee-filter')
        alert 'nice segue, eh?'
```

coffeescript

- .js:

```
square = function(x) {
  return x * x;
};
if (typeof elvis !== “undefined” &&
elvis !== null) {
  alert(“I knew it!”);
}
```

- .coffee:

```
square = (x) -> x * x
alert “I knew it!” if elvis
```

coffeescript, ctd.

- fewer {}, no ; or var, function() simplified
- lexical scope handled by block
- better truthiness
- rb-like returns
- easier to read and define loops
- inline decisions and existential op
  - alert “whut” if errors?
- gem ‘coffee-rails’, included by default.

SASS / .scss

- what happened to .sass?

- ```
  @import “colors” // list of color variables
  .sidebar {
    a {
      color: $link;
      text-decoration: none;
      &:hover {
        color: $accent;
      }
    }
  ```

- nesting, variables, functions, imports, mixins and more.

- visit the Charlotte UX group!
  ```

Brief Q/A?

Where does all this live?

In the ~~asshat~~ asset pipeline, of course.

Views and Assets

- app/views/* #=> views
- app/views/layouts/* #=> layouts
- assets/* #=> js, css, images
- vendor/assets/*
- lib/assets/*
- public/* #=> just files

Asset Pipeline

- if it ain't broke? wut?
- .scss/.js/.coffee easier to find, maintain, and debug
- isolates vendor/lib stuff
- minifies/packages everything for faster page loads
- fingerprints for cachiness

Asset Pipeline, ctd

- Manifests: application.(js|css)
- .js:
  - //= require thing
    //= require_tree ../templates
- .css:
  - *= require_self
    *= require path/to/thing

Asset Pipeline, ctd

- use the helpers!
  - javascript_include_tag
  - stylesheet_include_tag
  - ..and magic friends
- update your web server config in production, btw.

Search Paths?

- In config/application.rb or config/environments/whatever.rb
  - config.assets.paths << rrj(..)
- In controllers/initializers:
  - prepend_view_path(path)
  - NOT a great implementation, ensure this only happens once

MOAR extra credit

```
- config.asset_host #cdns!
- config.asset_path #weird deploys
- config/locales/en.yml
  - %h1=t :awesome_title
```

Brief Q/A?

Caching in General

- Caching prevents unnecessary execution and network traffic.
  - Browser Caches (Etags?)
  - Network Caching (CDN?)
  - Query Caching (Another Show?)
- Steve Souders:
  - http://www.stevesouders.com/blog/
  - *High Performance Web Sites*

- Make Fewer HTTP Requests
- Use a Content Delivery Network
- Add an Expires Header
- Gzip Components
- Put Stylesheets at the Top
- Put Scripts at the Bottom
- Avoid CSS Expressions
- Make JavaScript and CSS External
- Reduce DNS Lookups
- Minify JavaScript
- Avoid Redirects
- Remove Duplicate Scripts
- Configure ETags
- Make AJAX Cacheable
- ... more details at Souders' blog

Caching in Rails

- Page Caching
- Action Caching
- Fragment/Partial Caching
- Sweepers
- don’t forget to turn it on!
  - `config.action_controller.perform_caching = true`

Page Caching

- captures the whole page as an .html file in public/
- adjust your web server accordingly
- In the controller:
  - `caches_page :action_name`
  - `expire_page :action => :action_name`

Action Caching

- Use when filters are important!
- config.cache_store = :store_type, {}
- :memory_store, :file_store, :mem_cache_store, :ehcache (jRuby)
- caches_action / expire_action
- options same as page caching. mostly.

Fragment Caching

- same as Action cache, but happens granularly in the view

```
- cache(:action => :something,
        :action_suffix => “kitteh_toys”) do
    %h1= @kitteh
    = render @kitteh.toys
    ...
- end
```

```
 expire_fragment(options + :controller => :something)
```

or use a string id:

```
- cache(“stuff!”) do
```

Sweepers

- like Observer
- @object gets passed to sweeper
- have callbacks to @object's lifecycle events
- better than managing it yourself

One Last Q/A?

Thanks!

ben@vandgrift.com / @bvandgrift
http://ben.vandgrift.com