

Building and Working with Static Sites in Ruby on Rails

Ben Scofield
ben@viget.com
Senior Developer, Viget Labs

May 18, 2007

© Copyright 2007 Viget Labs, LLC – www.viget.com



Me

- 8 years building web apps (Perl, VBScript, C#, Java, PHP)
- Ruby and Rails for 2 years
- Tech Lead for Seth Godin's Squidoo
- ben@viget.com

Viget Labs

- Full-service web design and development consultancy
- Three types of clients
 - Flash-based entertainment sites
 - Web 2.0 startups
 - Mid-sized, **mostly static** sites

“Static” sites

- Fully static: CMS
- Small scale: About Us, Contact Info
- Large scale: ACVA (funside.com)

VIRGINIA ALEXANDRIA
The Fun Side of the Potomac

Meetings, Groups & Events | Press & Film | Co

Home About Alexandria Things to Do Calendar Accommodations Din

Upcoming Events
74th Annual Historic Garden Tour

Hotel Special
Girls Getaway Shopping Weekend

Shopping > Art Galleries

- Antiques
- Art Galleries
- Boutiques
- Children's
- Crafts
- Florists
- Gifts & Specialty
- Gourmet
- Health & Beauty
- Home & Garden
- Jewelry & Accessories
- Menswear
- Pet Boutiques
- Souvenir & Museum Shops

Shopping
Chic boutiques, national names & vintage finds

Art Galleries

Handicap Accessible Pets Allowed

The Art League, Inc.
105 N. Union St.
Alexandria, VA 22314 [map it](#)
Phone: 703/663-1760
Fax: 703/663-5786
[www.theartleague.org](#)

Artcraft Collection
132 King St.
Alexandria, VA 22314 [map it](#)
Phone: 703/299-6616
Fax: 703/299-6619
[www.artcraftonline.com](#)

About Alexandria > History

- History
- Timeline
- Colonial
- African-American
- Civil War
- Our Favorite Sons
- Official Visitors' Guide
- Maps and Transportation
- Neighborhoods
- City Government
- Business Support
- Regional Tourism Organizations
- Virtual Tour

About Alexandria
Minutes from Washington, DC and Mount Vernon on the scenic GW Parkway

History

Located within eyesight of Washington, DC, Alexandria is a Colonial seaport, once George Washington's hometown. The city's roster of historic sites underlines how historically influential it has been. Thomas Jefferson entertained at Gadsby's Tavern; Robert E. Lee worshipped at Christ Church. The Old Town district includes more than 4,000 historic buildings, outstanding examples of early American architecture that now house small businesses and gracious homes, all waiting to give you a modern welcome.

[Timeline](#)

[Colonial](#)

Hybrid sites

- Distinct static and dynamic content (About Us)
- Intermingled static and dynamic content

THE PROBLEM

- Building a static site should be easy

And in PHP, it is

- Drop a file and see it run, even with embedded PHP code
- Use configuration files to route requests, set page-specific data, handle redirects, set page metadata...

But in Rails?

- Actions for every view? sigh
- Actionless views? pfft

- and what about page-specific data?

So why bother with Rails?

- Less interesting: morale
- More interesting: productivity
 - task switching and specialization
- Most interesting: hybrid sites
 - current and future

Notes

- Error handling - ignore its absence
- Historical tour, Edge at the time

Solution 1: Rails on PHP

- Routing + Convention + Action + Caching = basic PHP?

Routing

- The beauty of the catch-all route:

```
map.connect '*path', :controller => 'pages', :action => 'show'~
```

- Declare all other routes explicitly!
or namespace the catch-all:

```
map.connect 'static/*path', :controller => 'pages', :action => 'show'~
```

Convention

- All templates under a single app/views subfolder
- Move as much to the layout as possible

Action

```
def show-  
  page_path = params[:path].join('/')-  
  if File.exists?("#{RAILS_ROOT}/app/views/static/#{page_path}.rhtml")-  
    render :template => "static/#{page_path}"-  
  end-  
end-
```

Caching

- Short and sweet: `cache_page :show`
- Rails caches under the request URI, not the controller/action

Problems with Solution #1

- Manage page data in the views? pfft
- Manually expire the cache? pfft
- No support for redirects (site revisions)? pfft

Solution 2: Configuratron 2000

- Use configuration files to overcome the problems of Solution 1

navigation.yml

- Central management for pages, page data, and relationships

```
--- !omap~
- about:~
  title: About Alexandria~
  page_variables:~
    sidebar: about~
    callout: ramsay,guide~
- about/history:~
  parent_url: about~
  title: History~
  page_variables:~
    section: history~
    callout: ramsay,architecture~
- about/maps:~
  parent_url: about~
```

Memory's cheap, right?

- Load it all into memory

```
PAGES = YAML::load(ERB.new(IO.read("#{RAILS_ROOT}/config/navigation.yml")).result)
```

- and pull it out when you need it

```
def show
  page_path = params[:path].join('/')
  unless PAGES.select{|page| page[0] == page_path}.empty?
    render :template => "static/#{page_path}"
  else
    # ... check for redirect
  end
end
```

Caching

- Expire the entire cache when navigation.yml is updated

redirects.yml

- Use the redirect entry to find the new page

```
welcome.asp: /about~  
visitor_pta.asp: /about/maps
```

```
redirect_hash = YAML::load(IO.read("#{RAILS_ROOT}/config/redirects.yml"))~  
if redirect_hash.has_key?(page_path)~  
  headers['Status'] = '301 Moved Permanently'~  
  return redirect_to redirect_hash[page_path]~  
end~
```

Problems with Solution #2

- Keeping it all in memory? pfft
- Upload new config files to change content? pfft
- All-or-nothing caching? pfft
- Not very Rails Wayish, is it?

Solution 3: CRUDification

- So what is Rails best at?
 - CRUD!
- Can we make static sites CRUDdier without creating a full-blown CMS?

Move it all to the DB...

```
create_table :pages do |t|
  t.column :parent_id, :integer
  t.column :url, :string, :null => false, :limit => 255
  t.column :title, :string, :limit => 255
  t.column :created_at, :datetime
  t.column :updated_at, :datetime
end

add_index :pages, :parent_id
add_index :pages, :url, :unique => true

create_table :variables do |t|
  t.column :page_id, :integer
  t.column :name, :string, :limit => 64
  t.column :value, :text
end

add_index :variables, :page_id

create_table :redirects do |t|
  t.column :old_url, :string, :limit => 255
  t.column :page_id, :integer
end

add_index :redirects, :old_url, :unique => true
add_index :redirects, :page_id
```

Pages and Variables

```
class Page < ActiveRecord::Base
  has_many :redirects, :dependent => :delete_all
  has_many :variables, :dependent => :delete_all

  validates_presence_of :url

  acts_as_tree

  def after_save
    if @new_variables
      self.variables.clear
      @new_variables.each do |v|
        v.page_id = self.id
        v.save
      end
    end
  end
end

#...
```

```
class Variable < ActiveRecord::Base
  belongs_to :page

  validates_presence_of :page_id, :name, :value
end
```

Redirects

```
class Redirect < ActiveRecord::Base
  belongs_to :page
  validates_presence_of :old_url, :page_id
end
```

Sitemaps for nothing

[sitemap](#) | [pages](#) | [redirects](#)

Sitemap

- [About Us](#)
 - [Contact Us](#)
 - [FAQ](#)
- [Help](#)

Cascading Variables for Free

```
def inherited_variables(reload = false)-
  @inherited_variables = nil if reload-
  parent_variables = (self.parent ? self.parent.all_variables(reload) : [])-
  @inherited_variables ||= parent_variables.reject { |var| self.variables.map(&:name).include?(var.name) }-
end-

-
def all_variables(reload = false)-
  @all_variables = nil if reload-
  if self.root == self-
    @all_variables ||= self.variables(reload)-
  else-
    @all_variables ||= self.inherited_variables(reload) + self.variables(reload)-
  end-
end-
```

Web UI

Redirects

- about.php => About Us (about) [destroy](#)

Create a new redirect

Old URL

New Destination

Save

[sitemap](#) | [pages](#) | [redirects](#)

New Page

Parent Page

URL

Title

Cache sweeping made easy

```
class PageSweeper < ActionController::Caching::Sweeper
  observe Page

  def after_save(page)
    expire_cache_for page
  end

  def after_destroy
    expire_cache_for page
  end

  private
  def expire_cache_for(page)
    PagesController.expire_page page.url
  end
end
```

Problems with Solution #3

- Search
- Portability

Solution 4: Feed the DB

- Store page content in the DB
- Upload files or enter content through the UI
- Search with ... whatever



Reduce, Reuse, ~~Recycle~~

- Package the framework for distribution



Question 1: ~~To Engine or To Plugin?~~

- ~~Engines~~
 - ~~Cons: deprecated, people yell at you~~
- ~~Plugins~~
 - ~~Pros: everybody loves plugins!~~

Question 2: To Generate or To App?

- Generators
 - Pros: customized code
 - Cons: hard to update
- Plugin Applications
 - Pros: easy to update, fast start
 - Cons: possible conflicts

Plugin Applications

- Miniature Rails applications - controllers, models, and views
- Grant complete, self-contained functionality to another application

Using the framework

- Install the plugin-app
- Update config files
 - environment.rb and routes.rb
 - (optional) .htaccess and environment.rb
- Run the migrations
- Enjoy your newly more-static site!

Key Points

- The catch-all route is the best kind of magic
- Play to Rails' strengths
- Handle both large- and small-scale static sites

Get the (alpha) framework at
<http://www.viget.com/railsconf/>

Questions?
Ask `em now or email me at
ben@viget.com

No questions, eh?

- Why even bother with this? Why not a CMS?
- Use content_for in the views to move more to the layout
- Plugems vs. engines vs. plugins?
- More about plugin-apps?