Why You Should Never Use an ORM
You crazy man...
Any intelligent fool can make things bigger, more complex, and more violent.

Albert Einstein
My Path Of Violence
HTTParty
Give Yourself Constraints

Recently, I had a hernia and surgery to fix it. This knocked me out of the game and onto the couch for a couple weeks. During my recovery, I had a lot of time to think. I also had a lot of time to miss what I do every day.

This was the longest period in several years for me without creating. Once I felt good enough to get back at it, even if only for a few hours, I decided to focus all this pent up energy on something new.

What I wanted to do, was to think through a problem different than I ever have. I have been creating applications pretty much the same way for quite some time. Sure, MongoDB changed my methods a bit, but I knew I had not used it to its full potential, as I typically start all new MongoDB projects with MongoMapper.

What to Build

First, I thought about what to build. I have a plethora of "someday" ideas that have never made it out of that stage. One of those ideas was to build a simple analytics program.

Sure, there are a crap ton of analytics apps out there, including Google Analytics, but not one thus far has hit the sweet spot I am looking for.

The Old Constraint

Back in the day, I would entertain every whim I had. This is great for learning a lot of new things, but I never really focused and finished anything. What I had was a project directory full of half (or less) finished ideas.

When I actually forced myself to work on only a project or two I chose (MongoMapper and Harmony), I noticed that I actually finished things and had something to show for myself.

Important: The constraint of what I could work on made me more productive than working on whatever I was inspired to work on.
Three Points
Think About Your

1) Interface
Think About Your

1) Interface
2) Data
Think About Your

1) Interface
2) Data
3) Code
Think About
Your Interface
ORMs too often lead to interface laziness.

John Nunemaker
site.memberships.create({
  :user_id => user.id,
  :owner   => true,
})
site.add_owner(user)
membership.update_attributes({
  :state => 1,
})
membership.update_attributes({
  :state => 'maximized',
})
user.maximize(site)
content.to_json({
    'a crap ton': 'of options'
})
ContentPresenter.new(site, date, {
    :page => params['page'],
}).to_json
class ContentPresenter
  include BasePresenter

  def initialize(site, date, options={})
    @site, @date = site, date
    @options = options
  end

  def page # ...
  def per_page # ...
  def total # ...
  def path # ...
  def prev_path # ...
  def next_path # ...
  def next_page_path # ...
  def prev_page_path # ...
  def paginated # ...
  def content # ...
  def as_json(options=nil)
    {
      'date' => @date.to_s,
      'prev_path' => prev_path,
      'next_path' => next_path,
      'content' => content,
      'page' => page,
      'per_page' => per_page,
      'total' => total,
      'prev_page_path' => prev_page_path,
      'next_page_path' => next_page_path,
    }
  end
end
class ContentPresenter
  include BasePresenter

  def initialize(site, date, options = {})
    @site, @date = site, date
    @options = options
  end

  def page
    # ...
  end

  def per_page
    # ...
  end

  def total
    # ...
  end

  def path
    # ...
  end

  def prev_path
    # ...
  end

  def next_path
    # ...
  end

  def next_page_path
    # ...
  end

  def prev_page_path
    # ...
  end

  def paginated
    # ...
  end

  def content
    # ...
  end

  def as_json(options = nil)
    {
      'date' => @date.to_s,
      'prev_path' => prev_path,
      'next_path' => next_path,
      'content' => content,
      'page' => page,
      'per_page' => per_page,
      'total' => total,
      'prev_page_path' => prev_page_path,
      'next_page_path' => next_page_path,
    }
  end
end
Content.paginate(
    {
        :conditions => {
            :site_id => site.id,
            :date => date,
        },
        :page => params['page'],
        :per_page => 15,
    }
)
site.content_for_date(date, {
    :page => params['page'],
})
Think About

Your Interface
Think About

Your Data
ORMs sometimes hide creative ways you can store and retrieve your data.

John Nunemaker
A Story of Failure and Triumph
{
    '_id' => 'site_id:2011-03-28',
    '/' => {'v' => 200, 't' => 'Home'},
    '/about/' => {'v' => 50, 't' => 'About'},
    '/foo/' => {'v' => 23, 't' => 'Foo!'},
}
You say heresy! I say use case...
<table>
<thead>
<tr>
<th>Page</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventMachine and Passenger /blog/archives/2011/05/04/event...</td>
<td>57</td>
</tr>
<tr>
<td>Home /</td>
<td>53</td>
</tr>
<tr>
<td>Class and Instance Variables In Ruby /blog/archives/2006/11/1...</td>
<td>47</td>
</tr>
<tr>
<td>Include vs Extend in Ruby /blog/archives/2009/05/15/include-vs...</td>
<td>19</td>
</tr>
<tr>
<td>Class and Instance Methods in Ruby /blog/archives/2009/05/11...</td>
<td>19</td>
</tr>
<tr>
<td>Parsing XML with Ruby /blog/archives/2008/08/11/parsing-xml...</td>
<td>13</td>
</tr>
<tr>
<td>Why I think Mongo is to Databases what Rails was to Framew...</td>
<td>13</td>
</tr>
<tr>
<td>I Have No Talent /blog/archives/2010/01/12/i-have-no-talent/</td>
<td>13</td>
</tr>
<tr>
<td>Bedazzle Your Bash Prompt with Git Info /blog/archives/2009/0...</td>
<td>12</td>
</tr>
<tr>
<td>It's an HTTParty and Everyone Is Invited! /blog/archives/2008/0...</td>
<td>10</td>
</tr>
<tr>
<td>jQuery on Rails: Why Bother? /blog/archives/2008/11/20/jquery....</td>
<td>9</td>
</tr>
<tr>
<td>Hi My Name is John... /blog/archives/2011/03/21/hi-my-name-is-...</td>
<td>8</td>
</tr>
<tr>
<td>You're An Idiot For Not Using Heroku /blog/archives/2009/11/0...</td>
<td>7</td>
</tr>
<tr>
<td>MongoMapper, The Rad Mongo Wrapper /blog/archives/2009/01/28/mongomapper...</td>
<td>7</td>
</tr>
<tr>
<td>SSH Tunneling in Ruby /blog/archives/2011/04/19/ssh-tunneling...</td>
<td>7</td>
</tr>
</tbody>
</table>

Viewing 1 to 15 of 74
Content.get("#{site.id}:2011-03-28")
Allowing us to turn off recording of content.

jnunemaker (author)
March 22, 2011
write data

ensure_index(site_id, date, path)
read data

ensure_index(site_id, date, views)
{
    '_id' => BSON::ObjectId.new,
    'sid' => site_id,
    'p' => '/about/',
    'd' => '2011-03-28',
    'v' => 50,
}

NewContent.for_site_and_date(site, date)
db['c.2011.5']
db['c.2011.5']
    read index
    write index
$db['c.2011.4'] \Rightarrow db['c.2011.5']$

read index

write index
NewContent is now Content.

jnunemaker (author)
March 28, 2011
/about/
/this/is/my/really/long/and/descriptive/path/and/of/course/we/need/to/have/a/query/string?gibberish=true&yunolikeshorturls=false#DontForgetToHashTagIt
Upsert based on hash instead of url and path for content.

junnemaker (author)
April 16, 2011
Every bit

Counts
Integers
Site.create(:state => 'enabled')
class Site
  States = {
    'enabled' => 1,
    'disabled' => 2,
  }
end

Site.create({
  :state => Site::States['enabled'],
})
class Site
    def enabled?
        state == States['enabled']
    end

    def disabled?
        !enabled?
    end
end
Compression
require 'rsmaz'

compressed = RSmaz.compress(str)
decompressed = RSmaz.decompress(compressed)
require 'zlib'

compressed = Zlib::Deflate.deflate(str)
decompressed = Zlib::Inflate.inflate(compressed)
require 'msgpack'

ids = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
compressed = MessagePack.pack(ids)
decompressed = MessagePack.unpack(compressed)
class Stylesheet
  class RSmazCompressor
    def self.compress(str)
      RSmaz.compress(str)
    end

    def self.decompress(str)
      RSmaz.decompress(str)
    end
  end
end

class ZlibCompressor
  def self.compress(str)
    Zlib::Deflate.deflate(str)
  end

  def self.decompress(str)
    Zlib::Deflate.inflate(str)
  end
end

Compressors = {
  1 => RSmazCompressor,
  2 => ZlibCompressor,
}

key = read_attribute(:compressor_id)
value = read_attribute(:contents)
compressor.decompress(value)
Compressors = {
    1 => RSmazCompressor,
    2 => ZlibCompressor,
}

key :compressor_id, Integer
key :contents, String

validates_inclusion_of :compressor_id,
    :within => Compressors.keys

def contents
    value = read_attribute(:contents)
    compressor.decompress(value)
end

def compressor
    Compressors[compressor_id]
end
end
Partial Updates
site.atomic_update_attributes(attrs)
Unused Fields
Where

Counts
Memory/Disk/Network
class SiteMode
  include Scam

  attr_accessor :name

  def password_required?
    id == 2
  end

  def item_cache?
    id == 1
  end
end
SiteMode.create({
  :id => 1,
  :name => 'live'
})

SiteMode.all

# find by id or string id
pp SiteMode.find(2)
pp SiteMode.find('2')
class Plan
  include Toy::Store

  store(:memory, {})

  attribute(:title, String)
  attribute(:price, Integer)
end
class Asset
  plugin Joint
  attachment :file

  def page_cache(version=\nnil)
    page_cache_original
    page_cache_version(version)
  end
end
end
current_user.sites.map do |site|
  Site.get(site['id'])
end
ids = current_user.sites.map do |sitel|
  site['id']
end

Site.all(:_id.in => ids)
A Little Ruby Can Go A Long Way
Move
Move
BigintMove
Move
BigintMove
MakeYouWannaMove
Move
BigintMove
MakeYouWannaMove
DaMove
Move
BigintMove
MakeYouWannaMove
DaMove
SmoothMove
Move
BigintMove
MakeYouWannaMove
DaMove
SmoothMove
NightMove
Move
BignumMove
MakeYouWannaMove
DaMove
SmoothMove
NightMove
DanceMove
Partition.create({
  :association => :moves,
  :model       => Move,
  :first_id    => 1,
  :writer      => false,
})
Partition.for_writes.model.create(...)
Partition.since(since_id, last_move_id).map do |p|
  send(p.association).since(since_id)
end
Think About

Your Data
Think About

Your Code
No code is faster than no code.
$ bx ruby performance/reads.rb

Collection#find_one
0.270231008529663

Site.first
0.69925594329834
<table>
<thead>
<tr>
<th>Path</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>/track</td>
<td>no</td>
</tr>
<tr>
<td>/content</td>
<td>no</td>
</tr>
<tr>
<td>/referrers</td>
<td>no</td>
</tr>
<tr>
<td>/sites</td>
<td>yes</td>
</tr>
<tr>
<td>/users</td>
<td>yes</td>
</tr>
</tbody>
</table>
class Hit
  def site
    @site ||= begin
      query = {_id => site_id}
      options = {fields => ['tz']}
      collection.find_one(query, options)
    end
  end
end
Don’t be afraid to write code
Don’t be afraid to read code
Don’t be afraid to fail
Site.find(id)

Site.create({
  :title => 'RailsTips',
})

site.update_attributes(attrs)

site.to_json
Think About Your Code
It takes a touch of genius—and a lot of courage—to move in the opposite direction.

Albert Einstein
Thank you!

john@orderedlist.com
@jnunemarker