MongoDB & Harmony

Real World Usage of MongoDB
Hello!

steve@orderedlist.com
@orderedlist
We make things simple. You like simple, right?

Our Latest Article
April 18
Great Lakes Ruby Bash
Last weekend in Lansing, MI, Steve and I both spoke at the Great Lakes Ruby Bash. Steve spoke on designing code and I on my lack of talent. We had a great time and love attending regional conferences like the Bash.

3 Comments

Other Recent Articles
- Lessons Learned
  - Bootstrapping Harmony
  - 13 Comments
- Stop Being an Idiot
  - 29 Comments
- Stop Blogging
  - 64 Comments

Recommended Resources
- Getting Started with HTML5
- Fun with Functions in JavaScript
- Rails & MVC
- Single Line CSS

Harmony
Harmony is a powerful web-based platform for creating and managing your websites.

Check out more of our Resources
Find out more about Harmony

Steve Smith
Steve architects awesome user interfaces and web designs.

John Nunemaker
John builds functional and accessible software for the web.
Welcome to Harmony

Enjoy Building Websites Again

http://get.harmonyapp.com
Things to Cover

1. What is Harmony?
2. Multiple Item Types
3. Completely Custom Data
4. Images and Files
5. Activity Streams
6. Wrap Up
What is Harmony?

Understanding Storage Needs
Harmony is for Building Websites
Harmony is designed to

Be Flexible
Harmony is designed to

Store Web Content
Demo Time
Things to Cover

1. What is Harmony?
2. Multiple Item Types
3. Completely Custom Data
4. Images and Files
5. Activity Streams
6. Wrap Up
Multiple Item Types

Flexibility in Collections
Harmony has Pages
Harmony has

Blogs
Harmony has

Blog Posts
Harmony has

Blog Labels and Archives
Harmony has Many More Types...
All These Things

Have a URL
We Use

Single Collection Inheritance
class Item
  include MongoMapper::Document

  key :_type, String, :index => true
  key :title, String, :required => true
  key :path, String, :required => true, :index => true
  key :template_name, String
  key :published_at, Time
  key :parent_id, ObjectId, :index => true
  key :parent_ids, Array, :index => true
  key :site_id, ObjectId, :required => true, :index => true
  key :nav_item, Boolean, :default => true
  key :asset_ids, Array, :index => true

timestamps!
userstamps!

  # more code here

end
class Page < Item

    key :position, Integer, :default => 1

    # more code here

end
class Blog < Item

  key :position,       Integer, :default => 1
  key :labels_as,      String,  :default => 'tags'
  key :custom_permalink_structure, String

  # more code here

end
class BlogPost < Item

  key :accepting_comments, Boolean, :default => true
  key :labels, Set, :index => true
  key :year_id, ObjectId
  key :month_id, ObjectId
  key :day_id, ObjectId

  # more code here

end
Separation of Code into

Meaningful Keys
One Collection Means

Simple Querying
Things to Cover

1. What is Harmony?
2. Multiple Item Types
3. Completely Custom Data
4. Images and Files
5. Activity Streams
6. Wrap Up
Custom Data

Unknown Possibilities
Every Item in Harmony gets

A Title and a Path
All Other Data is

Defined by the User
templates
templates

Template 1

Template 2

Template 3
This data is not related, it Belongs Together
templates

- **Template 1**
  - Field 1
  - Field 2
  - Field 3

- **Template 2**
  - Field 4
  - Field 5
  - Field 6

- **Template 3**
  - Field 7
  - Field 8
  - Field 9

Use Embedded Documents
items
items

Item 1

Item 2

Item 3
class Template
  include MongoMapper::Document

  key :filename, String,   :index => true
  key :theme_id, ObjectId, :index => true
  key :contents, String
  timestamps!
  userstamps!

  many :fields

  # more code here

end
class Field
  include MongoMapper::EmbeddedDocument
  
  key :name, String, :required => true
  key :key, String, :required => true
  key :field_type_id, Integer, :required => true
  key :help_text, String
  key :settings, Hash
  key :required, Boolean
  
  embedded_in :template

  # more code here
end
class Item
  include Mongomapper::Document

  key :_type, String, :index => true
  key :title, String, :required => true
  key :path, String, :required => true, :index => true
  key :template_name, String
  key :published_at, Time
  key :parent_id, ObjectId, :index => true
  key :parent_ids, Array, :index => true
  key :site_id, ObjectId, :required => true, :index => true
  key :nav_item, Boolean, :default => true
  key :asset_ids, Array, :index => true
  timestamps!
  userstamps!

  many :data

  # more code here

end
class Datum
  include MongoMapper::EmbeddedDocument

  key :key, String, :required => true, :length => 2..100
  key :file_upload, Boolean, :default => false
  key :value

  embedded_in :item

  # more code here

end
Embedding allows **Maximum Flexibility**
Embedding allows

Keeping Data Together
Things to Cover

1. What is Harmony?
2. Multiple Item Types
3. Completely Custom Data
4. Images and Files
5. Activity Streams
6. Wrap Up
Images and Files

Binary Data Storage
Harmony contains several Types of Files
Harmony contains

Text Files

Stylesheets, JavaScripts, Includes, and Templates
Harmony contains

Binary Files

Images, File Uploads
All these are

Stored in Mongo
class Stylesheet
  include MongoMapper::Document
  include PageCacheable

  key :filename, String, :index => true
  key :contents, String
  key :processor, String, :default => 'plain'
  key :theme_id, ObjectId, :index => true
  timestamps!
  userstamps!

  # more code here

end
class StylesheetsController < ApplicationController
  caches_page :show

  def show
    render_not_found and return if params[:filename].blank?
    filename = File.basename(params[:filename].first, '.css')

    if stylesheet = Stylesheet.first(:filename => filename,
                                    :theme_id => params[:theme_id])
      if stale?(:etag => stylesheet,
                :last_modified => stylesheet.updated_at.utc,
                :public => true)
        render :text => stylesheet.processed_contents, :content_type => 'text/css'
      end
    else
      render_not_found
    end
  end
end
class Asset
  include MongoMapper::Document

plugin Joint

  def self.versions
    @versions ||= {
      :feature => [:resize, {:width => 640}],
      :thumb => [:crop_resize, {:dimensions => [145, 75]}],
      :square => [:crop_resize, {:dimensions => [75, 75]}],
      :profile => [:crop_resize, {:dimensions => [100, 100]}],
      :profile_small => [:crop_resize, {:dimensions => [50, 50]}],
    }
  end

key :title_tag, String
key :description, String # long description
timestamps!
userstamps!

attachment :file

  # more code here

end
asset = Asset.new
asset.file = params[:uploaded_file]

asset.file.id
asset.file.size
asset.file.type
asset.file.name
class FlyImages
  OriginalRegex = /^\assets\((.*)\)/(.*)$/  # /assets/:id/name.ext
  VersionRegex = /^\assets\((.*)\)/(.*)\((.*)\)$/  # /assets/:id/:version/name.ext

  def initialize(app)
    @app = app
  end

  def call(env)
    case Rack::Request.new(env).path_info
    when VersionRegex
      id, version = $1, $2
      serve_asset(id, version)
    when OriginalRegex
      id = $1
      serve_asset(id)
    else
      @app.call(env)
    end
  end

  def serve_asset(id, version=nil)
    if asset = Asset.find(id)
      asset.page_cache(version)
      [200, {'Content-Type' => asset.content_type}, [File.read(asset.page_cache_path(version))]]
    else
      [404, {'Content-Type' => 'text/plain'}, ['File not found.']]
    end
  end
end
Things to Cover

1. What is Harmony?
2. Multiple Item Types
3. Completely Custom Data
4. Images and Files
5. Activity Streams
6. Wrap Up
Activity Streams

Quick Information
Recent Activity on My Sites

YESTERDAY

- **Ordered List**
  - Lessons Learned Bootstrapping Harmony received a comment
    - Time: 2:04 pm

- **RailsTips**
  - I Have No Talent received a comment
    - Time: 3:41 am

WEDNESDAY

- **Ordered List**
  - Stop Being an Idiot blog post updated by *Steve Smith*
    - Time: 7:52 pm

- **Ordered List**
  - Stop Being an Idiot received a comment
    - Time: 7:48 pm

- **Ordered List**
  - A New Look, a New CMS blog post updated by *Steve Smith*
    - Time: 11:16 am

- **Harmony Website Management**
  - Kickin’ Sass received a comment
    - Time: 10:27 am

- **Ordered List**
  - A New Look, a New CMS received 3 comments
    - Time: 9:34 am

- **RailsTips**
  - MongoMapper 0.7: Plugins received a comment
    - Time: 8:14 am

TUESDAY

- **RailsTips**
  - Canable: The Flesh Eating Permission System received a comment
    - Time: 8:46 pm

- **New City Catholic Church**
  - Prayer Requests updated 5 times by *Andrew Pautler*
    - Time: 8:18 pm

- **Ordered**
  - Great Lakes Ruby Bash received a comment
    - Time: 7:18 pm
activities

users

items

assets

themes
class Activity
  include MongoMapper::Document

  key :user,         Hash
  key :source,       Hash
  key :source_type,  String
  key :action,       String
  key :count,        Integer, :default => 0

  timestamps!

end
class Activity
  include MongoMapper::Document

  key :user, Hash
  key :source, Hash
  key :source_type, String
  key :action, String
  key :count, Integer, :default => 0

  timestamps!

  def source=(value)
    if value.is_a?(Hash)
      super
    else
      self.source_type = value.class.name
      super(value.to_mongo)
    end
  end

  def user=(value)
    if value.is_a?(User)
      super :id => value.id, :name => value.full_name
    else
      super
    end
  end
end
class Activity

# code on previous slide

def self.article_created(article)
    create(:source => article, :user => article.creator, :action => 'create')
end

def self.article_updated(article)
    first_or_new(:action => 'update', :source_id => article.id,
                :created_at.gt => Time.zone.now.beginning_of_day.utc).tap do |a|
        a.count += 1
        a.source = article
        a.user = article.updater
        a.save
    end
end

def self.article_published(article)
    create(:source => article, :user => article.updater, :action => 'publish')
end

def self.article_unpublished(article)
    create(:source => article, :user => article.updater, :action => 'unpublish')
end
end
Things to Cover

1. What is Harmony?
2. Multiple Item Types
3. Completely Custom Data
4. Images and Files
5. Activity Streams
6. Wrap Up
Wrapping Up

Recapping the Main Points
Take advantage of Dynamic Keys in Collections
Use Embedding to
Simplify Relationships
Storing Binary Data is
Simple & Useful
Embed Whole Objects for

Archiving and Query Reduction
Thank you!

steve@orderedlist.com

@orderedlist