

Don't Repeat Yourself

Repeat Others



**Why?**

# Why?

I am obsessed with improving

# Why?

I am obsessed with improving

I have learned a lot of late

# Why?

**I am obsessed with improving**

**I have learned a lot of late**

**I love sharing what I learn**

**Create**

**Steal**

**Think**

**Create**

**Steal**

**Think**



**What we have to learn to  
do, we learn by doing.**

**Aristotle**

**Don't Reinvent the Wheel.**

~~Don't~~ **Reinvent the Wheel.**



**Reinventing the wheel is as important to a developer's education and skill as weightlifting is to a body builder.**

**Jason Sage**

97 Things Every Programmer Should Know





**The lessons learned and deeper appreciation for ActiveRecord and DataMapper alone was enough to make it worth it.**

**John Nunemaker**

RailsTips.org Comment

# What Lessons?

I am glad you asked.

# Dynamic language

is dynamic

**autoload**

```
# person.rb
class Person; end

# some_other_ruby_file.rb
autoload Person, 'path/to/person'

# as soon as Person class is used,
# ruby requires the file
person = Person.new

# if it is not used, it is not required
```

```
module MongoMapper
```

```
  autoload :Document,      'mongo_mapper/document'  
  autoload :EmbeddedDocument, 'mongo_mapper/embedded_document'  
  autoload :Plugins,      'mongo_mapper/plugins'  
  autoload :Version,      'mongo_mapper/version'
```

```
module Plugins
```

```
  autoload :Associations, 'mongo_mapper/plugins/associations'  
  autoload :Callbacks,   'mongo_mapper/plugins/callbacks'  
  autoload :Clone,       'mongo_mapper/plugins/clone'  
  autoload :Descendants,   'mongo_mapper/plugins/descendants'  
  autoload :Dirty,       'mongo_mapper/plugins/dirty'
```

```
end
```

```
end
```

**method missing**

# dynamic finders

`find_by_first_name`

```
def method_missing(method, *args, &block)
  finder = DynamicFinder.new(method)

  if finder.found?
    dynamic_find(finder, args)
  else
    super
  end
end
```

# dirty keys

name\_changed?

```
def method_missing(method, *args, &block)
  if method.to_s =~ /(_changed\?!_change!_will_change!_was)$/
    method_suffix = $1
    key = method.to_s.gsub(method_suffix, '')

    if key_names.include?(key)
      case method_suffix
      when '_changed?'
        key_changed?(key)
      when '_change'
        key_change(key)
      when '_will_change!'
        key_will_change!(key)
      when '_was'
        key_was(key)
      end
    else
      super
    end
  else
    super
  end
end
```

**class\_eval, module\_eval, etc.**

```
class Person  
  include Mongomapper::Document  
end
```

```
module MongoMapper
  module Document
    def self.included(model)
      model.class_eval do
        extend Plugins
        plugin Plugins::Associations
        plugin Plugins::Equality
        plugin Plugins::Inspect
        # etc...
      end
    end
  end
end
end
```

```
class Activity
  include Mongomapper::Document

  key :source,      Hash
  key :source_type, String
  key :action,      String
  timestamps!
end
```

```
class Article
  include Mongomapper::Document

  key :title, String
end
```

```
article = Article.create(:title => 'Yo Dawg')
activity = Activity.create({
  :source          => article.to_mongo,
  :source_type     => 'Article',
  :action          => 'create'
})
```

```
class Activity
  include Mongomapper::Document

  key :source,      Hash
  key :source_type, String
  key :action,     String
  timestamps!

  def source=(value)
    self.source_type = value.class.name
    super value.to_mongo
  end
end
```

```
article = Article.create(:title => 'Yo Dawg')
activity = Activity.create({
  :source => article,
  :action => 'create'
})
```

```
class Activity
  module MongoMapperKeys
    def source
      read_key :source
    end

    def source=(value)
      write_key :source, value
    end

    def source?
      read_key(:source).present?
    end
  end

  include MongoMapperKeys
end
```

```
def create_accessors_for(key)
  accessors_module.module_eval <<-end_eval
    def #{key.name}
      read_key(:#{key.name})
    end

    def #{key.name}_before_typecast
      read_key_before_typecast(:#{key.name})
    end

    def #{key.name}=(value)
      write_key(:#{key.name}, value)
    end

    def #{key.name}?
      read_key(:#{key.name}).present?
    end
  end_eval

  include accessors_module
end
```

```
def accessors_module
  if key_accessors_module_defined?
    const_get 'MongoMapperKeys'
  else
    const_set 'MongoMapperKeys', Module.new
  end
end
```

# Dynamic language

is dynamic

# Objects

can do more than #new and #save

**Equality**

```
class Person
  attr_accessor :name

  def initialize(name)
    @name = name
  end
end
```

```
puts Person.new('John') == Person.new('John')
# false
```

```
puts Person.new('John').eql?(Person.new('John'))
# false
```

```
class Person
  attr_accessor :name

  def initialize(name)
    @name = name
  end

  def eql?(other)
    self.class.eql?(other.class) &&
      name == other.name
  end
end

puts Person.new('John') == Person.new('John')
# false

puts Person.new('John').eql?(Person.new('John'))
# true
```

```
class Person
  attr_accessor :name

  def initialize(name)
    @name = name
  end

  def eql?(other)
    self.class.eql?(other.class) &&
      name == other.name
  end
  alias :== :eql?
end

puts Person.new('John') == Person.new('John')
# true

puts Person.new('John').eql?(Person.new('John'))
# true
```

**Clone/dup**

```
class OptionsHash
  attr_reader :source

  def initialize(source)
    @source = source
  end

  def [](key)
    @source[key]
  end

  def []=(key, value)
    @source[key] = value
  end
end
```

```
hash1 = OptionsHash.new({:foo => 'bar'})  
hash2 = hash1.clone
```

```
puts hash1[:foo]  
# 'bar'
```

```
hash2[:foo] = 'surprise'
```

```
puts hash1[:foo]  
# 'surprise'
```

```
puts hash1.source.equal?(hash2.source)  
# true
```

```
class OptionsHash  
  def initialize_copy(other)  
    super  
    @source = @source.clone  
  end  
end
```

```
class OptionsHash
  def initialize_copy(other)
    super
    @source = @source.clone
  end
end
```

```
hash1 = OptionsHash.new({:foo => 'bar'})
hash2 = hash1.clone
```

```
puts hash1[:foo]
# 'bar'
```

```
hash2[:foo] = 'surprise'
```

```
puts hash1[:foo]
# 'bar'
```

```
puts hash1.source.equal?(hash2.source)
# false
```

# Hooks

```
class Item
  def self.inherited(subclass)
    puts self.inspect
    puts subclass.inspect
  end
end
```

```
class Page < Item
end
```

```
# Item
# Page
```

```
module MongoMapper
  module Plugins
    module Sci
      module ClassMethods
        def inherited(subclass)
          key :_type, String unless key?(:_type)
          unless subclass.embeddable?
            subclass.set_collection_name(collection_name)
          end
          super
        end
      end
    end
  end
end
end
end
```

```
module Heyooooooooo  
  def self.included(base)  
    puts "Heyooooooooo!"  
  end  
end
```

```
class User  
  include Heyooooooooo  
end
```

```
# Heyooooooooo!
```

```
module MongoMapper
  module Document
    def self.included(model)
      model.class_eval do
        extend Plugins
        plugin Plugins::Associations
        plugin Plugins::Equality
        plugin Plugins::Inspect
        # etc...
      end
    end
  end
end
end
```

# Exercises for the Listener

Validations, Callbacks, Comparable, Enumerable

# Objects

can do more than #new and #save

# Patterns

are not just for the enterprise

# Proxy

A class that is an interface to another class

```
class Proxy
  def initialize(source)
    @source = source
  end

  private
  def method_missing(method, *args, &block)
    @source.send(method, *args, &block)
  end
end
```

```
class Proxy
  def initialize(source)
    @source = source
  end

  private
  def method_missing(method, *args, &block)
    @source.send(method, *args, &block)
  end
end
```

```
results = [1, 2, 3, 4]
proxy = Proxy.new(results)
```

```
puts proxy.size # 4
puts results.size # 4
```

```
puts proxy[2] # 3
puts results[2] # 3
```

# Decorator

New behavior to object dynamically

```
query = Plucky::Query.new(collection)
docs = query.paginate(:per_page => 1)
```

```
puts docs.class           # Array
puts docs.total_entries   # 2
puts docs.total_pages     # 2
puts docs.current_page    # 1
```

```
puts [].total_pages # NoMethodError!
```

```
module Pagination
  def total_entries
    50
  end
```

```
  def total_pages
    5
  end
end
```

```
result = [1, 2, 3, 4]
result.extend(Pagination)
```

```
puts result.total_entries # 50
puts result.total_pages  # 5
```

```
module Plucky
  class Query
    def paginate(opts={})
      # some stuff
      paginator      = Pagination::Paginator.new(total, page, limit)
      query[:limit]  = paginator.limit
      query[:skip]   = paginator.skip
      query.all.tap do |docs|
        docs.extend(Pagination::Decorator)
        docs.paginator(paginator)
      end
    end
  end
end
end
```

```
require 'forwardable'
module Plucky
  module Pagination
    module Decorator
      extend Forwardable

      def_delegators :@paginator,
                    :total_entries, :total_pages,
                    :current_page, :per_page,
                    :previous_page, :next_page,
                    :skip,          :limit,
                    :offset,       :out_of_bounds?

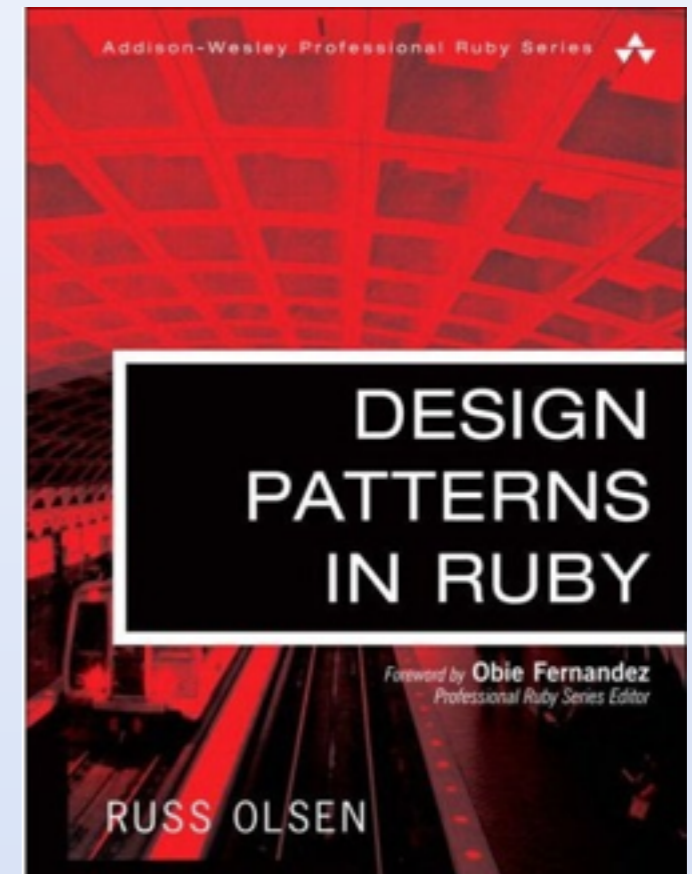
      def paginator(p=nil)
        return @paginator if p.nil?
        @paginator = p
        self
      end
    end
  end
end
end
```

# Identity Map

A man with two watches  
never knows the time

# Excercises for the Listener

Read Ruby Design Patterns



# Patterns

are not just for the enterprise

# APIs

or how to eat your own dog food

MongoMapper is

# Powered by Plugins

associations, callbacks, clone, descendants, dirty, equality, identity\_map, inspect, keys, logger, modifiers, pagination, persistence, protected, rails, serialization, timestamps, userstamps, validations

```
module MongoMapper
  module Plugins
    def plugins
      @plugins ||= []
    end

    def plugin(mod)
      extend mod::ClassMethods      if mod.const_defined?(:ClassMethods)
      include mod::InstanceMethods  if mod.const_defined?(:InstanceMethods)
      mod.configure(self)              if mod.respond_to?(:configure)
      plugins << mod
    end
  end
end
```

```
module MongoMapper
  module Document
    def self.included(model)
      model.class_eval do
        extend Plugins

        plugin Plugins::Document
        plugin Plugins::Associations
        plugin Plugins::Clone
        plugin Plugins::Equality
        plugin Plugins::Indexes
        plugin Plugins::Keys
        # etc
      end
    end
  end
end
```

```
module ActsAsListFu
  module ClassMethods
    def reorder(ids)
      # reorder ids...
    end
  end
end
```

```
module InstanceMethods
  def move_to_top
    # move to top
  end
end
```

```
def self.configure(model)
  model.key :position, Integer, :default => 1
end
end
```

```
class Foo
  include Mongomapper::Document
  plugin ActsAsListFu
end
```

```
Foo.reorder(...)  
Foo.new.move_to_top
```

# APIs

or how to eat your own dog food



**I want to make things, not  
just glue things together.**

**Mike Taylor**

Whatever Happened to Programming

Create

**Steal**

Think



**Good artists copy,  
great artists steal.**

**Pablo Picasso**

# What Have I Stolen?

I am glad you asked.

**httparty**

```
class Twitter  
  include HTTParty  
  base_uri 'twitter.com'  
end
```

```
module Scrobbler
  module REST
    class Connection
      def initialize(base_url, args = {})
        @base_url = base_url
        @username = args[:username]
        @password = args[:password]
      end

      def get(resource, args = nil)
        request(resource, "get", args)
      end

      def post(resource, args = nil)
        request(resource, "post", args)
      end

      # removed to shorten...
    end
  end
end
```

**happymapper**

```
class Status
  include HappyMapper

  element :id, Integer
  element :text, String
  element :created_at, Time
  element :source, String
  element :truncated, Boolean
  element :in_reply_to_status_id, Integer
  element :in_reply_to_user_id, Integer
  element :favorited, Boolean
  has_one :user, User
end
```

# RailsTips

One man, feverishly posting everything he learns. [RSS subscribe »](#)

[About](#) [Smorgasbord](#) [Projects](#) [Dude](#)

August 09, 2008  
Posted by John



Expert Web Design  
and Development.

[Ads from Ruby Row](#)

## Ruby Object to XML Mapping Library

While watching the opening ceremonies for the Olympics tonight, I came across a library named ROXML. An excerpt from the [ROXML website](#):

*ROXML is a Ruby library designed to make it easier for Ruby developers to work with XML. Using simple annotations, it enables Ruby classes to be custom-mapped to XML. ROXML takes care of the marshalling and unmarshalling of mapped attributes so that developers can focus on building first-class Ruby classes. As a result, ROXML simplifies the development of RESTful applications, Web Services, and XML-RPC.*

I've been wanting something like this for a while so I decided to give it a go. Below is an example of how to use it with the Delicious API and [HTTParty](#).

```
%w[rubygems roxml httparty pp].each { |x| require x }  
  
# <post href="http://code.google.com/p/sparrow/" hash="  
class Post  
  include ROXML  
  
  [:href, :hash, :description, :tag, :time, :others, :e  
    xml_attribute attribute  
  end  
end  
  
# <posts user="jnunemaker" tag="ruby">  
#   <post href="http://code.google.com/p/sparrow/" hash
```

### About

Authored by John Nunemaker (Noo-neh-maker), a web developer and programmer who has fallen deeply in love with Ruby. [More about John.](#)

### Need Web Help?

Hire me at [Ordered List](#)

Powered by  
*Harmony*



### Syndication

[RSS RailsTips Articles](#) - An assortment of howto's and thoughts on Ruby and Rails.

[RSS Rails Quick Tips](#) - Ruby and Rails related links that I find. Never more than 5 a day.

*PeepCode*  
screencasts

# August 9, 2008

## RailsTips

One man, feverishly posting everything he learns. [subscribe »](#)

[About](#) [Smorgasbord](#) [Projects](#) [Dude](#)

November 17, 2008  
Posted by John



Expert Web Design  
and Development.

[Ads from Ruby Row](#)

### HappyMapper, Making XML Fun Again

As much as [I write about XML](#), you would swear it is all I do, but I promise it is not. In fact, I do not really use XML that often, but I will admit that I am intrigued by it. A while back, you may remember, [I posted about ROXML](#), a ruby object to xml mapping library. I liked the idea but not the implementation. Soon after, I started playing around with what I have named HappyMapper, a ruby object to xml mapping library.

I wrote nearly 95% of it in a weekend and then let it sit. I let it sit so long that it started to rot. Today it hit me that I do not have to finish something in order to release it. The thing that wasn't working was xml with a default namespace. For good reasons I am sure, libxml-ruby does not like having default namespaces. I thought to myself, you know, this library is cool even without namespace junk. I mean who even uses namespaces other than Amazon. I started to package it for release and then I noticed a few nitpicky things. I tweaked them and five hours later I had also fixed the namespace issue and changed the API a bit. So much for releasing unfinished code in hopes that someone smarter than I would finish it up...



Logo created by Peter Cooper

#### Examples

But I digress, you do not care about all that, right? How about some examples? Twitter's xml seems to be popular on this here blawg, so I will start with that. Given this xml sample from twitter:

```
<statuses type="array">
```

#### About

Authored by John Nunemaker (Noo-neh-maker), a web developer and programmer who has fallen deeply in love with Ruby. [More about John.](#)

**Need Web Help?**  
Hire me at [Ordered List](#)

Powered by  
*Harmony*

#### Syndication

[RailsTips Articles](#) - An assortment of howto's and thoughts on Ruby and Rails.

[Rails Quick Tips](#) - Ruby and Rails related links that I find. Never more than 5 a day.

*PeepCode*  
screencasts

# November 17, 2008

**mongomapper**

**plucky**

**gem whois**

**canable**



**I steal.**

**John Nunemaker**

**What have I not stolen?**

# What have I not stolen?



Create

Steal

**Think**

**Make decisions**

```
class Account
  def add_user(user)
    user = user.is_a?(User) ? user : User.find(user)
    self.memberships << user.id
  end
end
```

```
class Account  
  def add_user(user)  
    self.memberships << user.id  
  end  
end
```

# Extraction vs Prediction

# Refactor

Addison-Wesley Professional Ruby Series



# REFACTORING

RUBY EDITION

JAY FIELDS ■ SHANE HARVIE ■ MARTIN FOWLER  
*with* KENT BECK

Query  
# normalized-criteria  
# normalized-options  
# normalized-key  
# normalized-value  
# normalized-unit  
# normalized-fields  
# normalized-order

Options-then-criteria  
# merge


Criteria  
# merge

↓ # modifier?  
# object-id-key?  
# separate-criteria-and-options

- don't allow merge if not  
the same collection

**Write**

## RailsTips

One man, feverishly posting everything he learns.  [subscribe »](#)

[About](#) [Smorgasbord](#) [Projects](#) [Dude](#)

May 10, 2010  
Posted by John

### MongoSF MongoMapper Video

In which I share a video of me presenting on MongoMapper.

Tags: [mongomapper](#) and [presentation](#) | [3 comments](#)

May 05, 2010  
Posted by John

### Improve Your Presentations In Under \$50

In which I provide some tips for presenting based on recent experiences.

Tags: [presentation](#) | [9 comments](#)

April 18, 2010  
Posted by John

### I Have No Talent (redux)

In which I turn a post into a keynote.

Tags: [presentation](#) | [9 comments](#)

March 30, 2010  
Posted by John

### Because Gem Names Are Like Domains in the 90's

In which I release a new gem that adds the whois command to Rubygems.

Tags: [gems](#) and [testing](#) | [19 comments](#)

March 26, 2010

### A Nunemaker Joint

#### About

Authored by John Nunemaker (Noo-neh-maker), a web developer and programmer who has fallen deeply in love with Ruby. [More about John.](#)

#### Need Web Help?


Hire me at [Ordered List](#)

Powered by  
*Harmony*



#### Syndication

 [RailsTips Articles](#) - An assortment of howto's and thoughts on Ruby and Rails.

 [Rails Quick Tips](#) - Ruby and Rails related links that I find. Never more than 5 a day.

*PeepCode*  
screencasts

**Create**

**Steal**

**Think**

# Thank you!

[john@orderedlist.com](mailto:john@orderedlist.com)

[@jnunemaker](https://twitter.com/jnunemaker)

**RailsConf Baltimore, MD**

June 8, 2010

**John Nunemaker**

Ordered List

