Ruby
Plus Rails
Plus Your Application
Minus Rails

Brian Guthrie
ThoughtWorks®
bguthrie@thoughtworks.com
http://twitter.com/bguthrie

The Last Ten Percent

Brian Guthrie
ThoughtWorks®
bguthrie@thoughtworks.com
http://twitter.com/bguthrie

How to do complex things in Ruby

(without shooting yourself in the foot)

2010

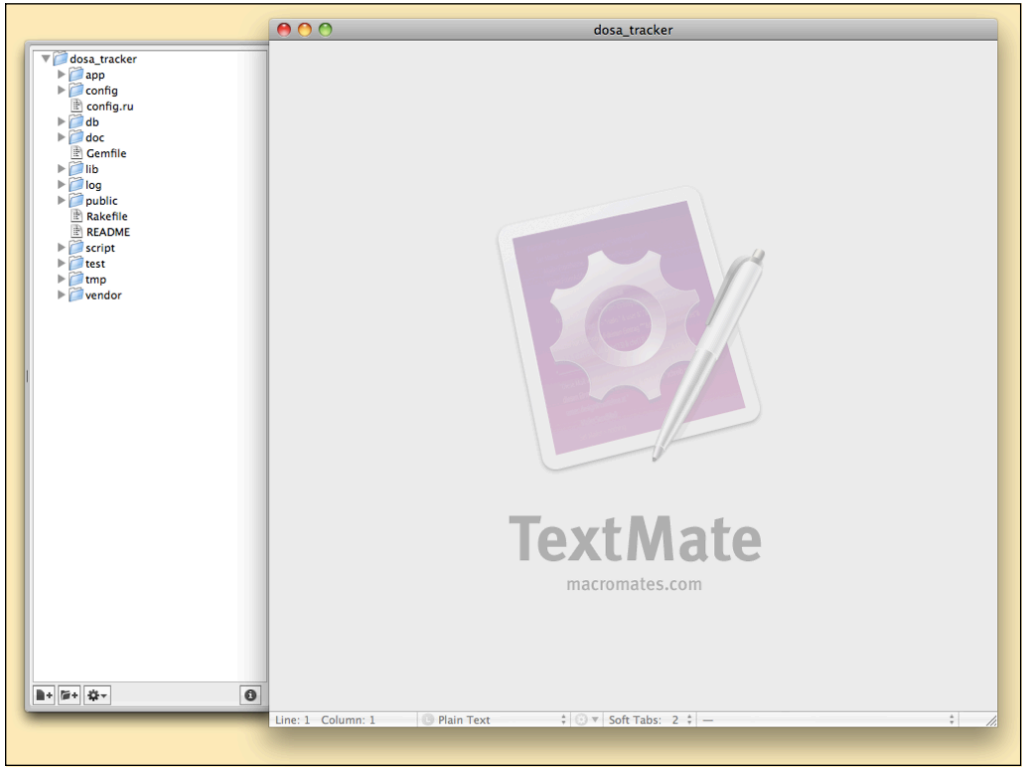
How to do **simple** things in Ruby

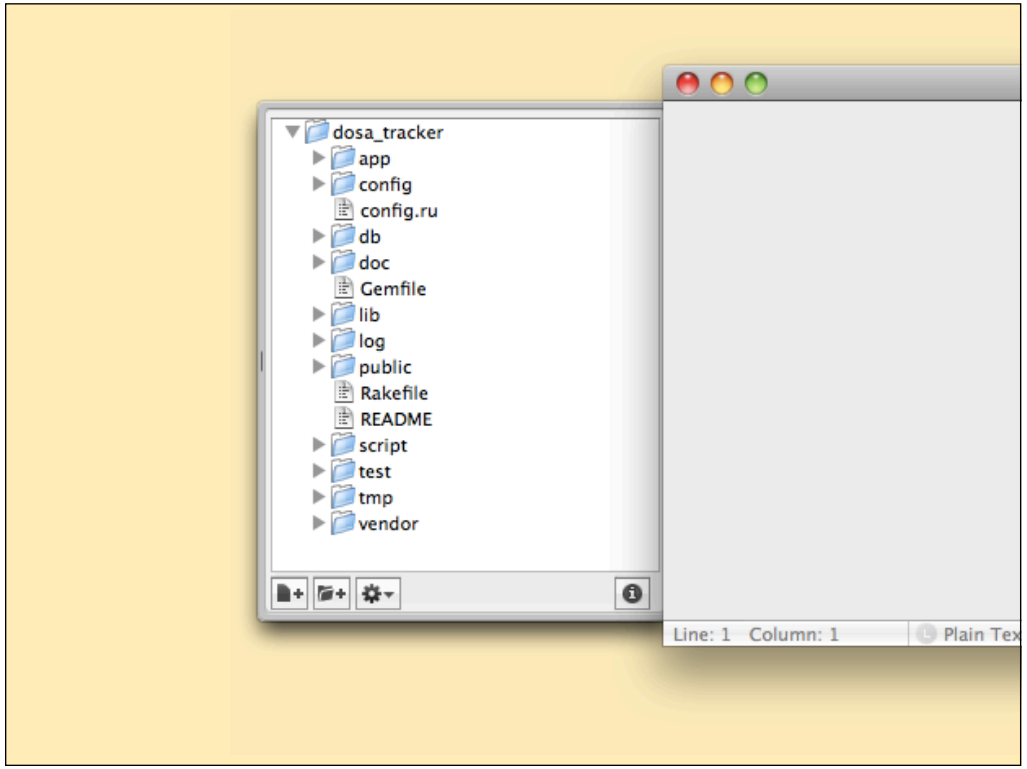
(with care and craft)

2011

the last

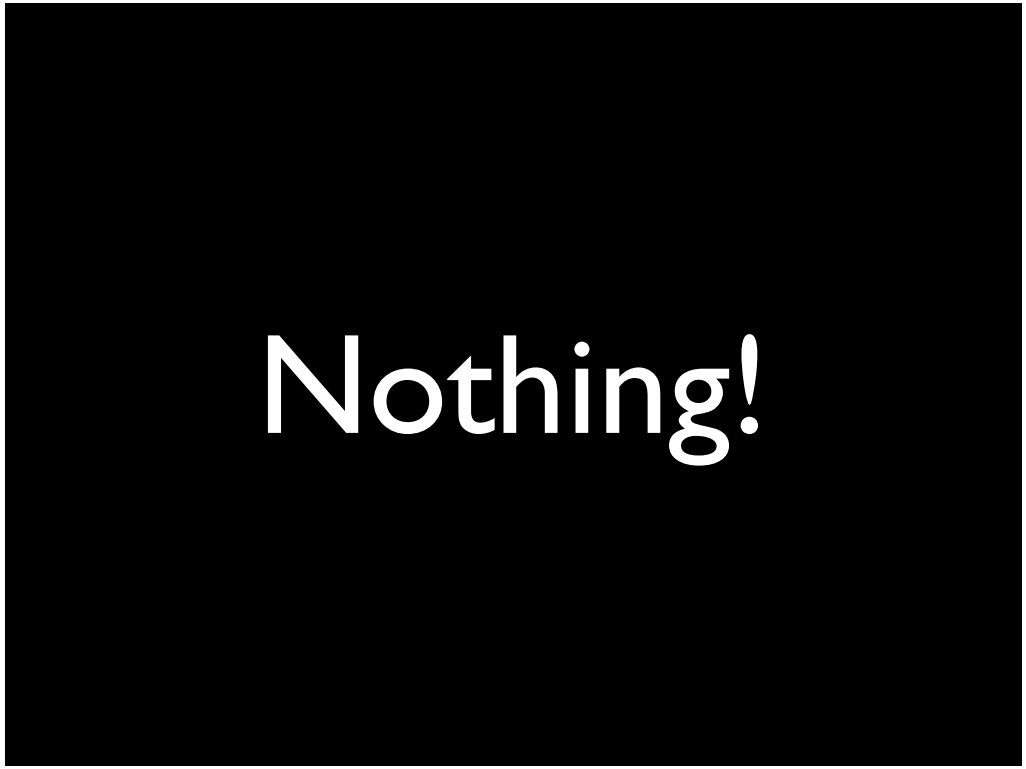
10%


dosa_tracker
dosa_tracker
app
config
config.ru
db
doc
Gemfile
lib
log
public
Rakefile
README
script
test
tmp
vendor
TextMate
macromates.com
Line: 1 Column: 1
Plain Text
Soft Tabs: 2

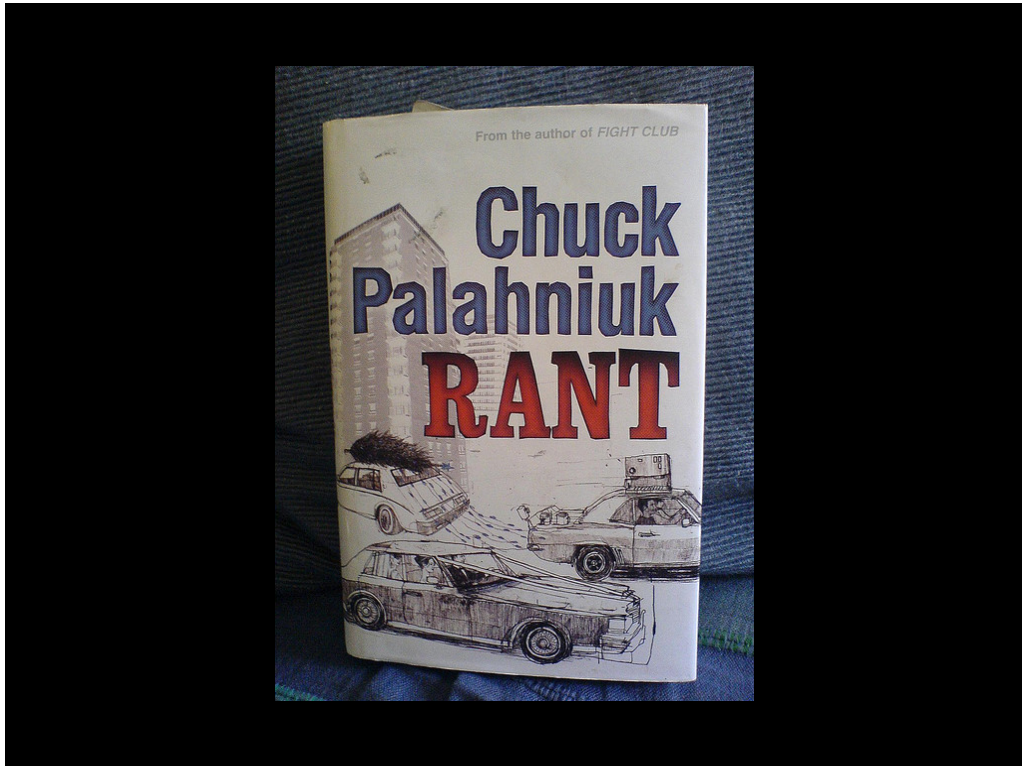

dosa_tracker
app
config
config.ru
db
doc
Gemfile
lib
log
public
Rakefile
README
script
test
tmp
vendor
Line: 1 Column: 1
Plain Tex

So what's hard about this?

(Is it time to go check out the talk in the other room? That guy seems cool.)


Nothing!

Every good talk needs a rant


From the author of FIGHT CLUB
Chuck
Palahniuk
RANT


not complex

There's an **army** of programmers
looking for Rails bugs

Your code? Just you.

Craft is *even more important.*


not
cookie
cutter

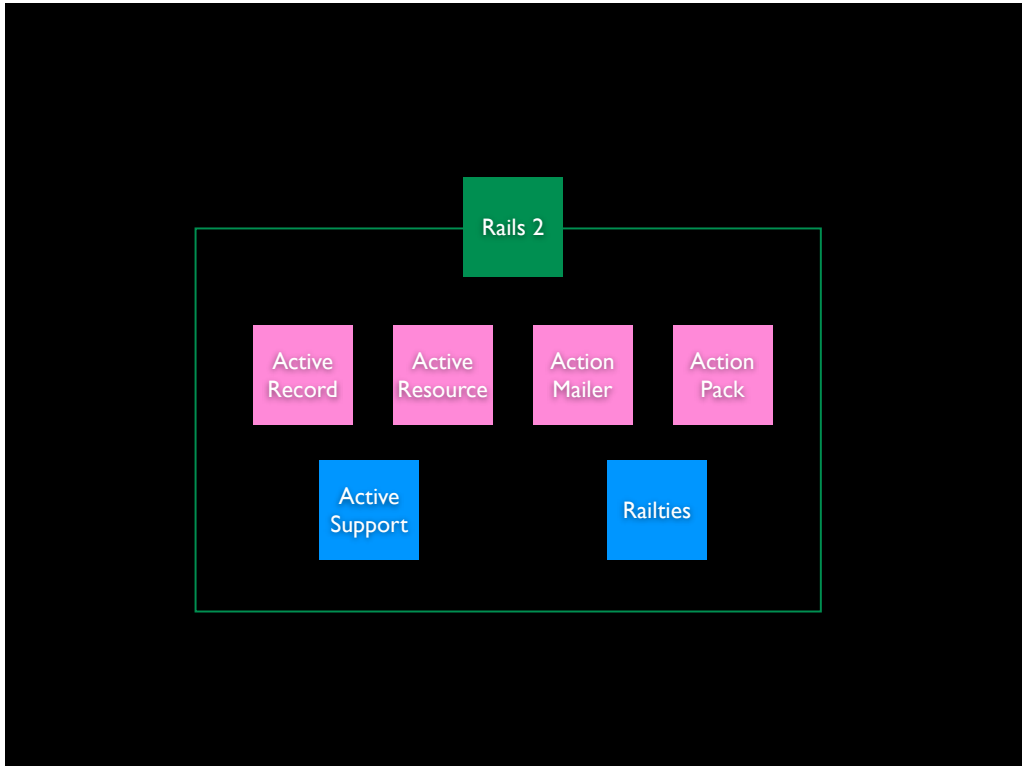
Most Ruby programmers are Rails programmers first

Learning what Rails *doesn't* do
is as important as learning what it *does*

For those craftspeople here:
Thank you

Gaps in Rails:
Where's the problem?

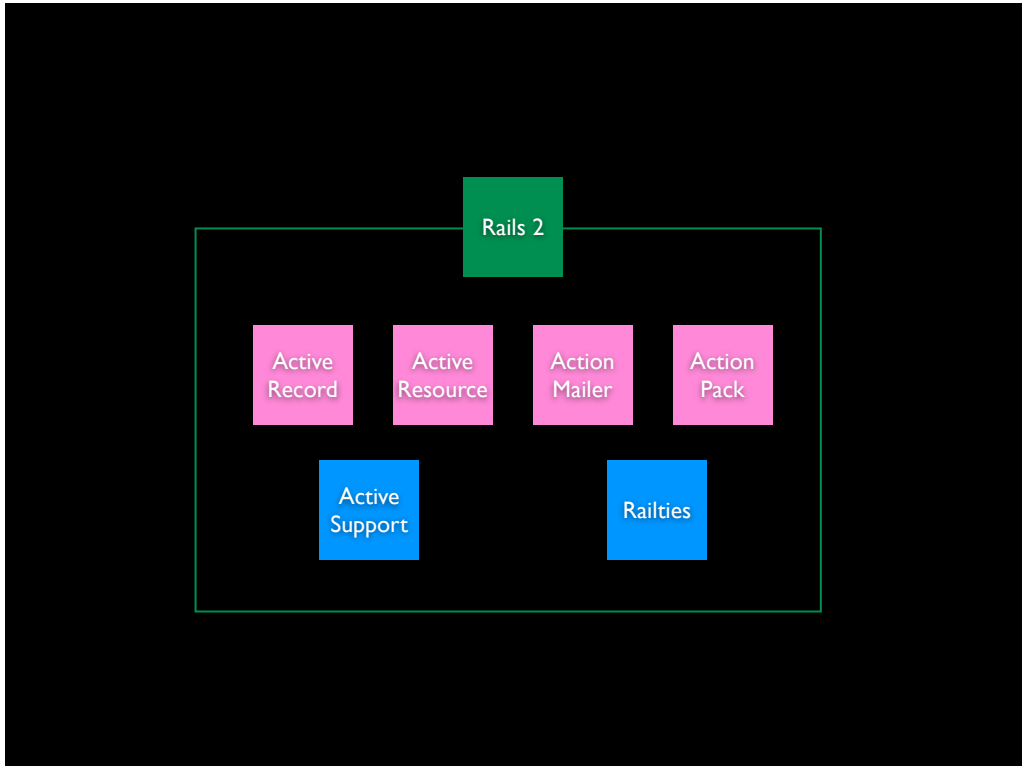
> Prior to Rails 3.0, if a plugin or gem developer wanted to have an object interact with Action Pack helpers, it was required to either copy chunks of code from Rails, or monkey patch entire helpers to make them handle objects that did not exactly conform to the Active Record interface. This would result in code duplication and fragile applications that broke on upgrades.
>
> *- ActiveModel README*

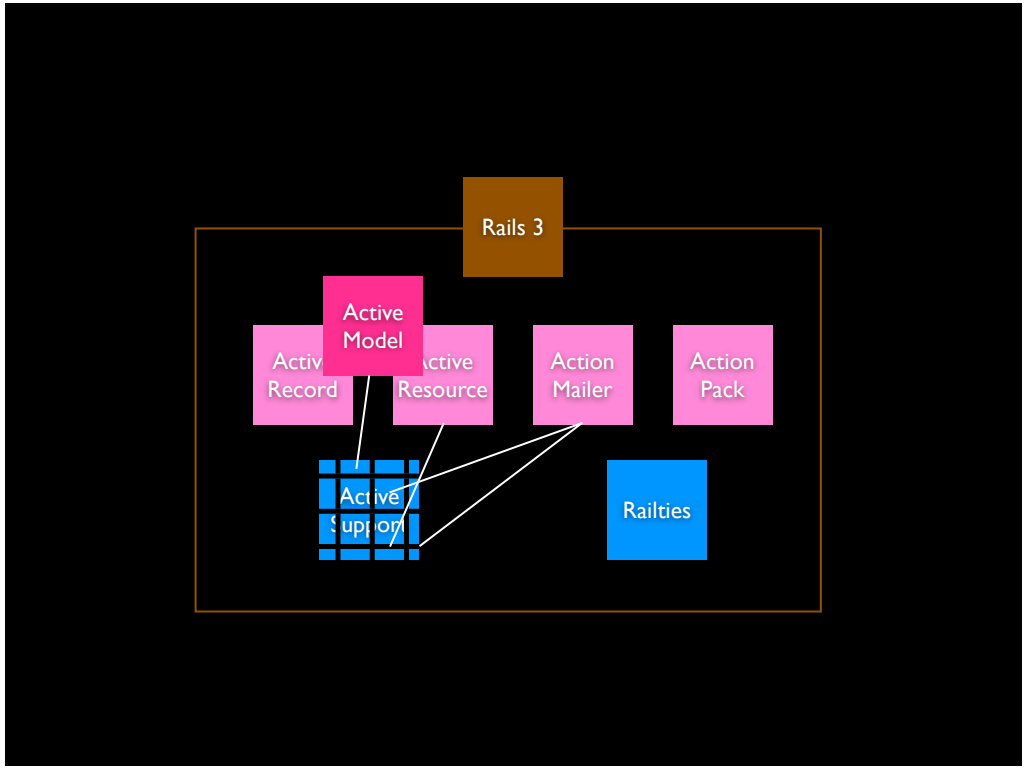

Rails 2
Active Record
Active Resource
Action Mailer
Action Pack
Active Support
Railties

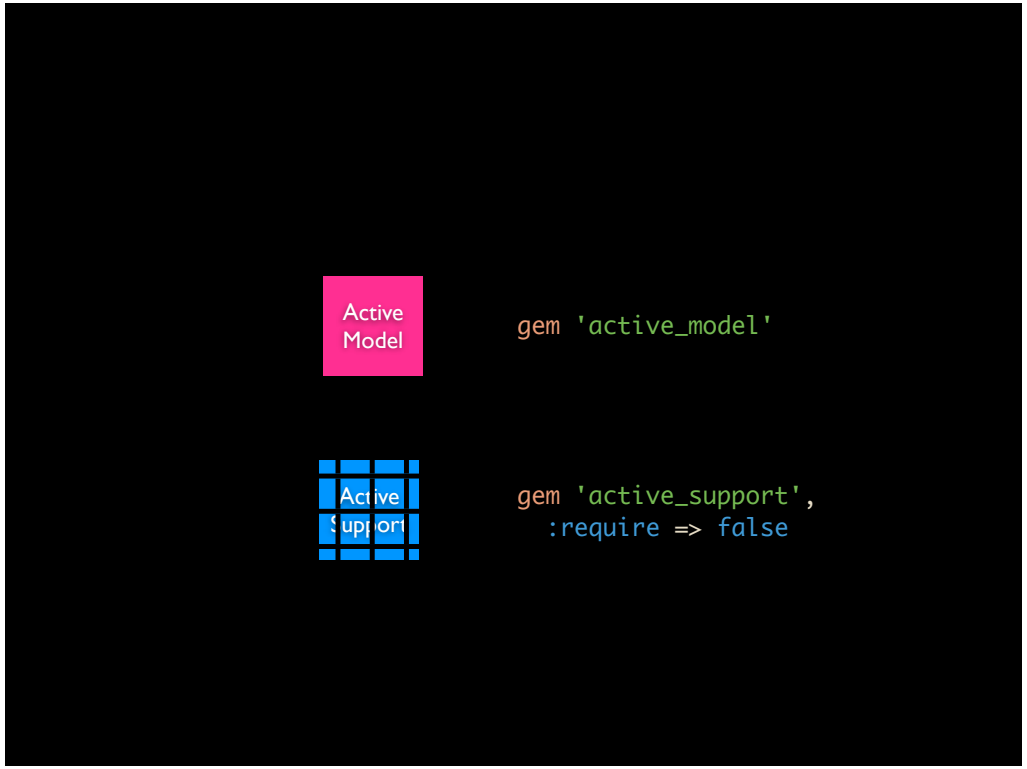
Presentation	Logic	Data
`form_for @user do`	`@user = User.find(params[:id])`	
`form_for @fb_user do`	`@fb_user = FbUser.find(params[:id])`	facebook

- Domain models that aren't database tables
- Importing and transforming data
- Web service clients
- Presenter objects

(not a comprehensive list)

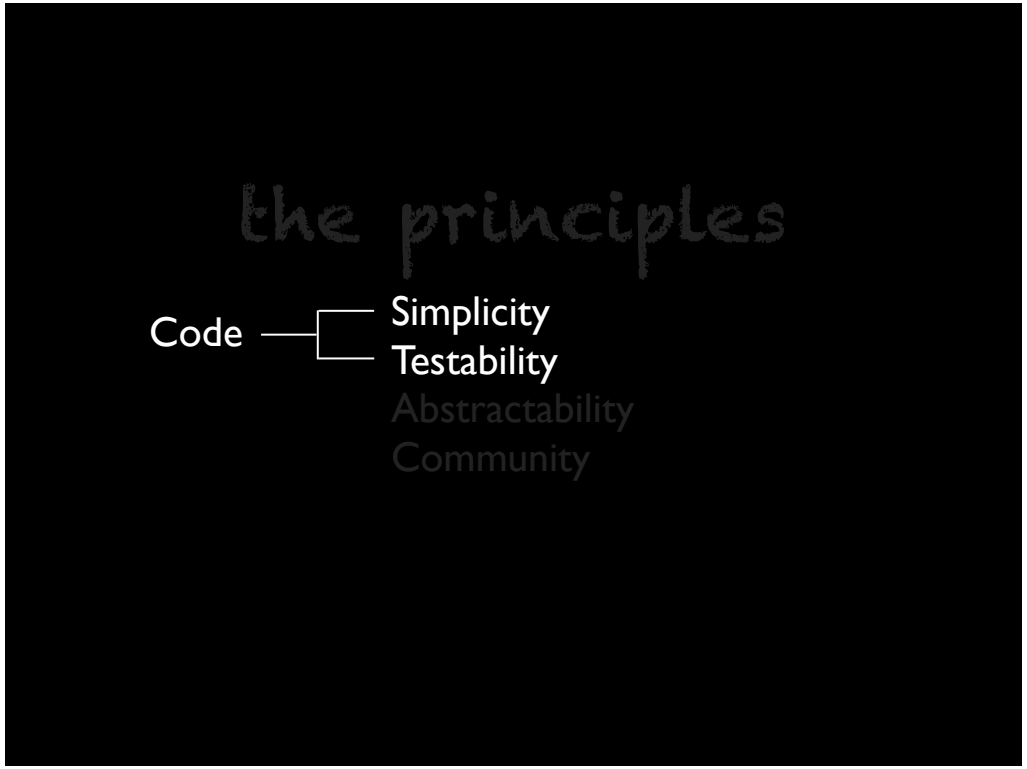

Rails 2
Active Record
Active Resource
Action Mailer
Action Pack
Active Support
Railties


Rails 3
Active Model
Active Record
Active Resource
Action Mailer
Action Pack
Active Support
Railties


Active Model
gem 'active_model'
Active Support
gem 'active_support',
:require => false

the principles

Simplicity
Testability
Abstractability
Community


the principles
Code
Simplicity
Testability
Abstractability
Community

S

Single responsibility principle
An object should have only a single responsibility.

O

Open/closed principle
Software entities … should be open for extension, but closed for modification.

L

Liskov substitution principle
Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.

I

Dependency inversion principle
Depend upon Abstractions. Do not depend upon concretions.

D

Interface segregation principle
Many client specific interfaces are better than one general purpose interface.

S **Single responsibility principle**
An object should have only a single responsibility.

O **Open/closed principle**
Software entities … should be open for extension, but closed for modification.

L **Liskov substitution principle**
Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.

I **Dependency inversion principle**
Depend upon Abstractions. Do not depend upon concretions.

D **Interface segregation principle**
Many client specific interfaces are better than one general purpose interface.

SD Card Principles

S **Single responsibility principle**
An object should have only a single responsibility.

O **Open/closed principle**
Software entities … should be open for extension, but closed for modification.

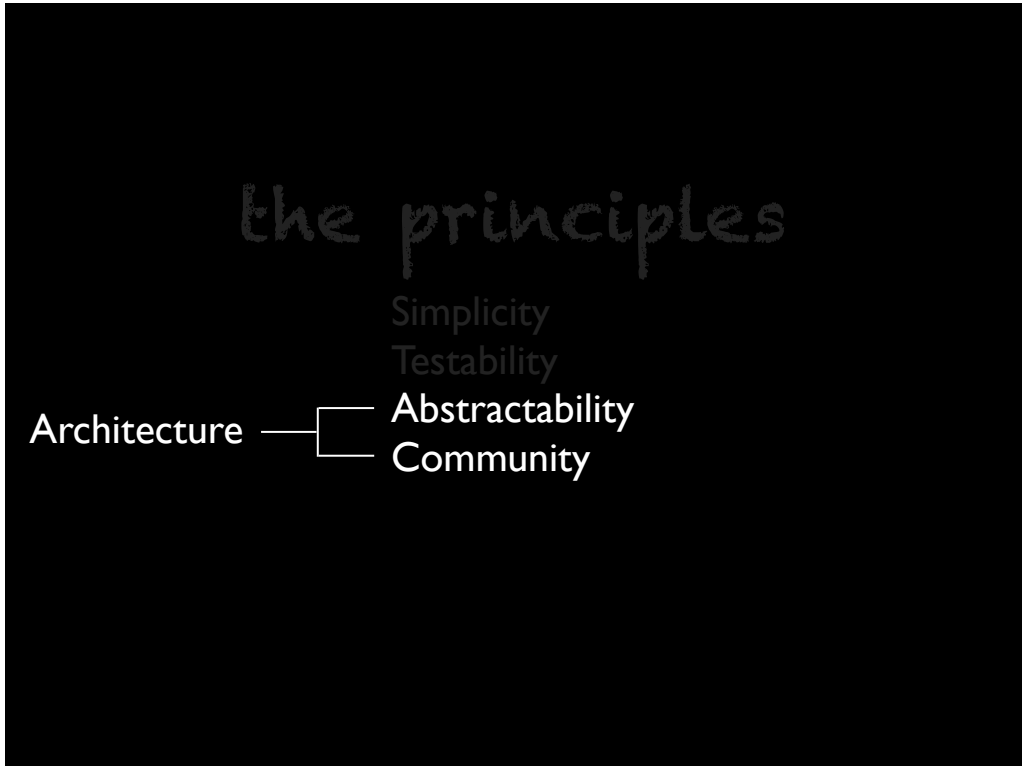
L **Liskov substitution principle**
Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.

I **Dependency inversion principle**
Depend upon Abstractions. Do not depend upon concretions.

D **Interface segregation principle**
Many client specific interfaces are better than one general purpose interface.

S O L I D

Mini SD Card Principles


the principles
Simplicity
Testability
Architecture
Abstractability
Community

Idiomaticity

Idiom (plural idioms)

1. A manner of speaking, a way of expressing oneself.
2. An artistic style (for example, in art, architecture, or music); an instance of such a style.
3. An expression peculiar to or characteristic of a particular language, especially when the meaning is illogical or separate from the meanings of its component words.
4. (linguistics) A communicative system under study, which could be called either a dialect or a language, when its status as a language or dialect is irrelevant.
5. (programming) A programming construct or phraseology generally held to be the most efficient, elegant or effective means to achieve a particular result or behavior.

http://en.wiktionary.org/wiki/idiom

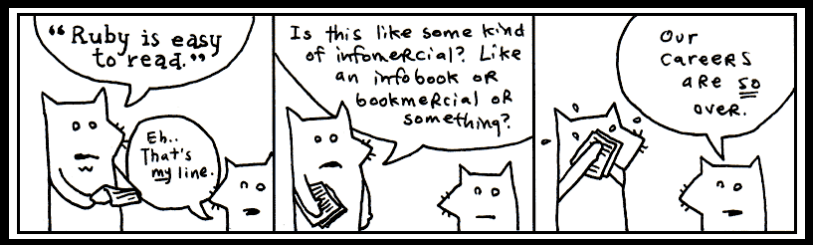
Idiomaticity

Idiom (plural idioms)

1. A manner of speaking, a way of expressing oneself.
2. An artistic style (for example, in art, architecture, or music); an instance of such a style.
3. An expression peculiar to or characteristic of a particular language, especially when the meaning is illogical or separate from the meanings of its component words.
4. (linguistics) A communicative system under study, which could be called either a dialect or a language, when its status as a language or dialect is irrelevant.
5. (programming) A programming construct or phraseology generally held to be the most efficient, elegant or effective means to achieve a particular result or behavior.

http://en.wiktionary.org/wiki/idiom

Some of the best Ruby programmers have gone against commonly-accepted style



Keep it simple when you can
Break the rules when you must
(or when it sounds like fun)

The best* Ruby code is clean, simple, and follows community idioms.

* Easiest to maintain.

the recipe

Build an object
Test that object
Expose a familiar API

Mmm... Masala Ruby

the recipe

Build an object
Test that object
Expose a familiar API

Antipattern: Bad Modeling


Dosa Tracker
(I smell a startup opportunity)

CSV File

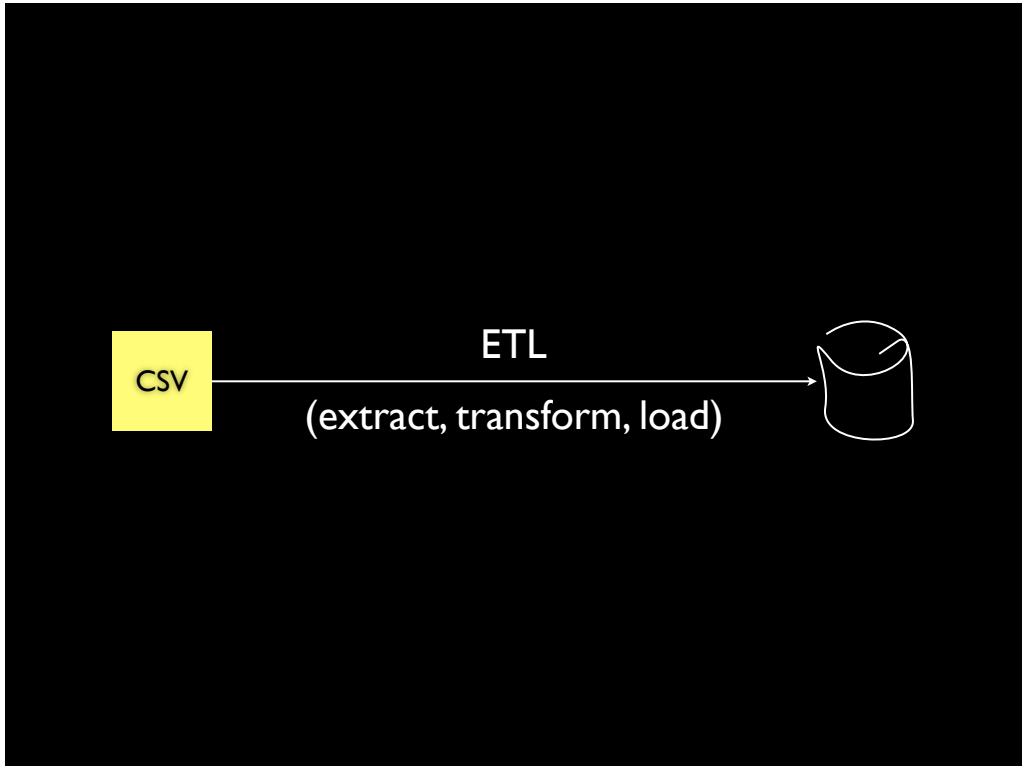
```
name,city,address,district,review,dosa types
Sukh Sagar,Chennai,2nd Ave,Anna Nagar,plain|paper|masala|mysore masala|rava
Sri Sai Sagar,Bangalore,"5th Cross, 18th Main Road, HAL 2nd
Stage",Indiranagar,plain|paper|masala|rava masala|mysore masala|coconut
```

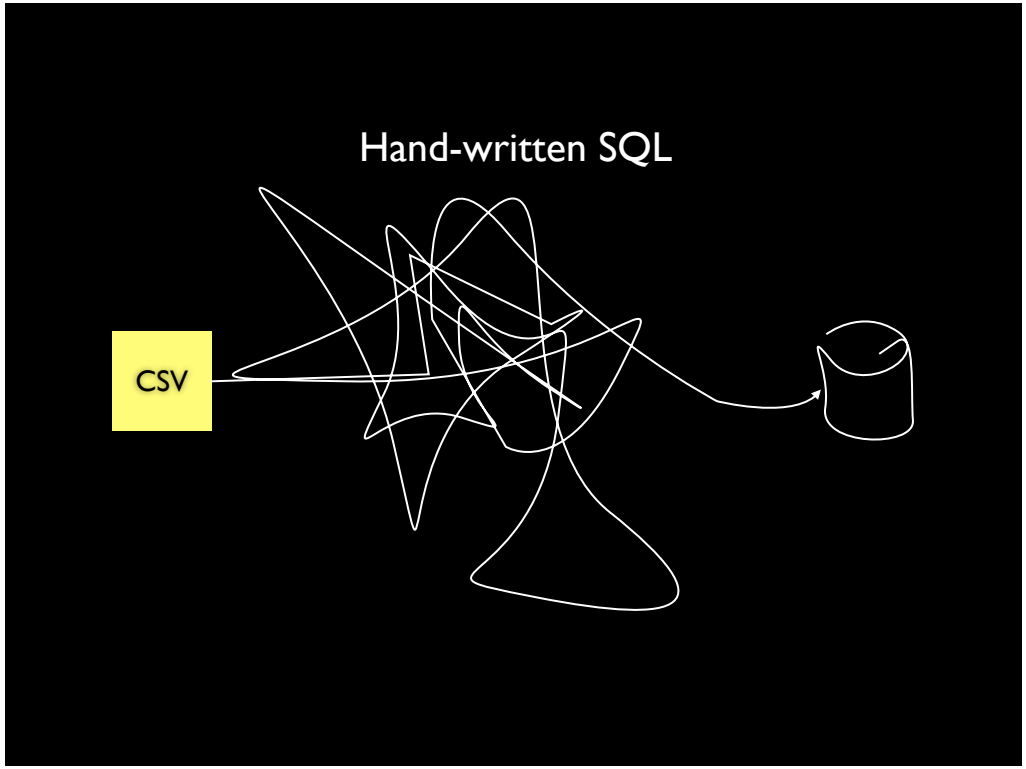
Columns

```
name,city,address,district,review,dosa types
Sukh Sagar,Chennai,2nd Ave,Anna Nagar,plain|paper|masala|mysore masala|rava
Sri Sai Sagar,Bangalore,"5th Cross, 18th Main Road, HAL 2nd
Stage",Indiranagar,plain|paper|masala|rava masala|mysore masala|coconut
```

Data

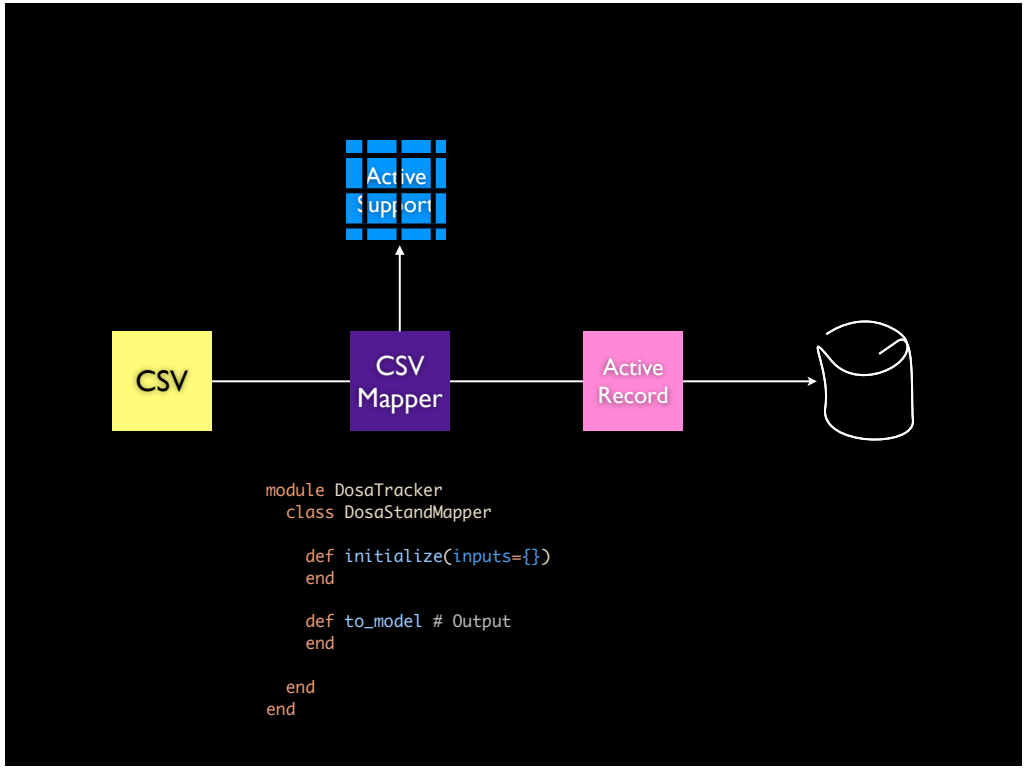
```
name,city,address,district,review,dosa types
Sukh Sagar,Chennai,2nd Ave,Anna Nagar,plain|paper|masala|mysore masala|rava
Sri Sai Sagar,Bangalore,"5th Cross, 18th Main Road, HAL 2nd
Stage",Indiranagar,plain|paper|masala|rava masala|mysore masala|coconut
```


CSV
ETL
(extract, transform, load)


Hand-written SQL
CSV

Slight improvement


```
module ABunchOfSuperAwesomeDataTransformationMethods

  def some_conversion_that_wont_make_sense_six_months_from_now(inputs)
  end

end
```




```ruby
module DosaTracker
  class DosaStandMapper

    def initialize(inputs={})
    end

    def to_model # Output
    end

  end
end
```

```ruby
module DosaTracker
  class DosaStandMapper

    def initialize(inputs={})
      @inputs = inputs
    end

    def location
      [ @inputs["city"], @inputs["address"], @inputs["landmark"] ].join(", ")
    end

  end
end
```

```ruby
require 'spec_helper'

describe DosaTracker::DosaStandMapper do

  context "#location" do
    it "should include the city and address" do
      DosaTracker::DosaStandMapper.new(
        'city' => 'Chennai',
        'address' => '2nd Avenue, Anna Nagar'
      ).location.should match("Chennai, 2nd Avenue, Anna Nagar")
    end
  end

end
```

```
context "#rating" do
  [
    [ "pretty good", 4 ],
    [ "bad", 1 ],
    [ "okay", 3 ],
    [ "awesome", 5 ],
    [ "meh", 2 ]
  ].each do |review, rating|

    it "should match a review of '#{review}' to a rating of #{rating} stars" do
      DosaTracker::DosaStandMapper.new(
        'review' => "it was sort of #{review}"
      ).rating.should == rating
    end

  end
end
```

```ruby
def rating
  case @inputs["review"]
  when /bad/; 1
  when /meh/; 2
  when /okay/; 3
  when /pretty good/; 4
  when /awesome/; 5
  end
end
```

Tip

Keep objects testable by reducing internal dependencies.

Each method tested individually.

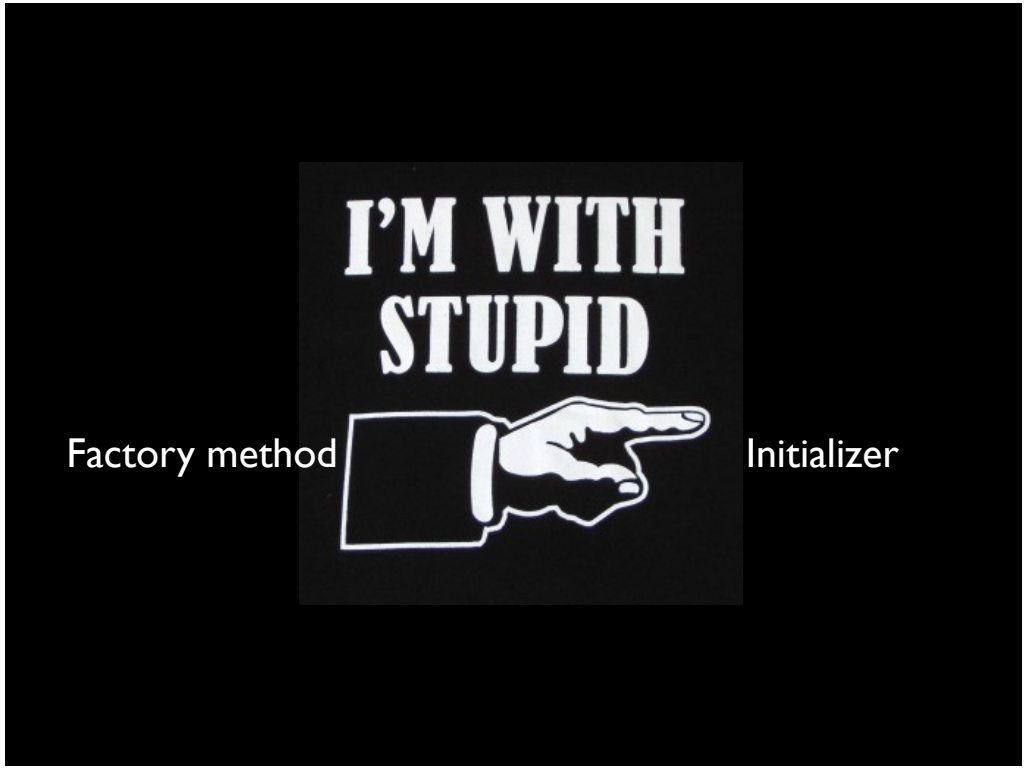
```
  context ".build" do

    EXAMPLE_CSV = <<-CSV
name,city,address,district,review,dosa types
Sukh Sagar,Chennai,2nd Ave,Anna Nagar,plain|paper|masala|mysore masala|rava
Sri Sai Sagar,Bangalore,"5th Cross, 18th Main Road, HAL 2nd Stage",Indiranagar,plain|
paper|masala|rava masala|mysore masala|coconut
CSV

    it "should instantiate an array of mappers based on the file it's given" do
      csv_file = Rails.root.join("test", "tmp.csv")
      csv_file.open("w") { |io| io << EXAMPLE_CSV }

      mappers = DosaTracker::DosaStandMapper.build(csv_file.open)
      mappers.size.should == 2
      mappers[0].name.should == 'Sukh Sagar'
    end
  end
```

```ruby
def self.build(io)
  CSV.read(io, :headers => :first_row).map do |row|
    self.new(row.to_hash)
  end
end
```

Tip

Initializers are dumb, factory methods are smart.

Follow Rails factory patterns where appropriate.


I'M WITH
STUPID
Factory method
Initializer

```
  context "#dosa_types" do
    it "map a single dosa type to a list with a single item" do
      DosaTracker::DosaStandMapper.new(
        'dosa types' => 'rava'
      ).dosa_types.should == [ DosaType.new(:name => "rava") ]
    end

    it "map pipe-delimited dosa types to a list of multiple types" do
      DosaTracker::DosaStandMapper.new(
        'dosa types' => 'rava|masala|paper|butter'
      ).dosa_types.should == [
        DosaType.new(:name => "rava"),
        DosaType.new(:name => "masala"),
        DosaType.new(:name => "paper"),
        DosaType.new(:name => "butter")
      ]
    end
  end


  def dosa_types
    @inputs["dosa types"].split("|").map do |name|
      DosaType.new(:name => name)
    end
  end
```

Tip

Don't stub in the class you're testing.

It's a smell. Reduce dependencies instead.

```
context "#to_model" do
  it "should expose a model object with the newly-populated data" do
    DosaTracker::DosaStandMapper.new(
      'name' => 'Sukh Sagar',
      'city' => 'Chennai',
      'address' => '2nd Avenue, Anna Nagar',
      'review' => 'Pretty good',
      'dosa types' => 'plain|paper|masala|mysore masala|rava'
    ).to_model.location.should match("Chennai, 2nd Avenue, Anna Nagar")
  end
end
```

```ruby
class DosaStand < ActiveRecord::Base
  # Domain logic here
end

module DosaTracker
  class DosaStandMapper
    def to_attributes
      {
        name: self.name,
        rating: self.rating,
        location: self.location,
        dosa_types: self.dosa_types
      }
    end

    def to_model
      DosaStand.new(self.to_attributes)
    end
  end
end
```

Tip

Abstract when you've got a pattern.

But not before.

Tip

When you've abstracted a pattern, make it a gem.

This is how good ideas are born. You don't have to release it.

Antipattern: Too much view logic

Active Model

No integration dependencies

Mix and match

Quack like a duck

Keep it consistent

Hero of Rails 3

```
        autoload :AttributeMethods
        autoload :BlockValidator, 'active_model/validator'
        autoload :Callbacks
        autoload :Conversion
        autoload :DeprecatedErrorMethods
        autoload :Dirty
        autoload :EachValidator, 'active_model/validator'
        autoload :Errors
        autoload :Lint
        autoload :MassAssignmentSecurity
        autoload :Name, 'active_model/naming'
        autoload :Naming
        autoload :Observer, 'active_model/observing'
        autoload :Observing
        autoload :Serialization
        autoload :TestCase
        autoload :Translation
        autoload :Validations
        autoload :Validator
```


Mix and match to suit your needs

```ruby
class DosaSearchController < ApplicationController

  def new
    @search = DosaSearch.new
  end

  def show
    @search = DosaSearch.new(params[:search])
    @results = @search.results
  end

end
```

```ruby
describe DosaSearch do
  include Test::Unit::Assertions
  include ActiveModel::Lint::Tests

  def model
    subject
  end

  ActiveModel::Lint::Tests.public_instance_methods.map{|m| m.to_s}.grep(/^test/).each do |m|
    example m.gsub('_',' ') do
      send m
    end
  end
end
```

```ruby
class DosaSearch
  include ActiveModel::Conversion

  def initialize(attrs={})
    @attributes = attrs
  end

  def persisted?
    false
  end
end
```

Tip

If you accept attributes, mass-assign.

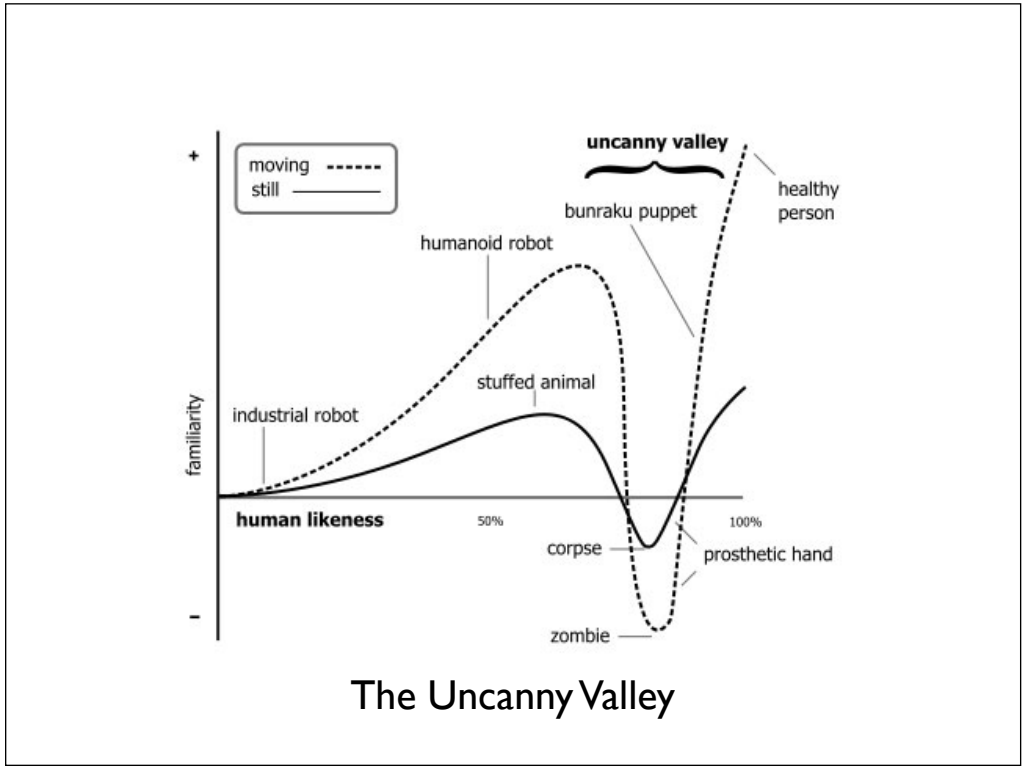
It's simple and straightforward.

```ruby
context "attributes" do
  it "should assign city" do
    DosaSearch.new(:city => "Chennai").city.should == "Chennai"
  end
end

context "validations" do
  it "should not be valid if a city is not specified" do
    DosaSearch.new.should have(1).error_on(:city)
  end

  it "should be valid if it has a city" do
    DosaSearch.new(:city => "Mysore").should be_valid
  end
end
```

```ruby
class DosaSearch
  include ActiveModel::Conversion
  include ActiveModel::Validations

  attr_reader :attributes
  attr_accessor :city

  validates_presence_of :city

  def initialize(attrs={})
    @attributes = attrs

    @attributes.each do |key, value|
      send "#{key}=", value
    end
  end

  def persisted?
    false
  end
end
```

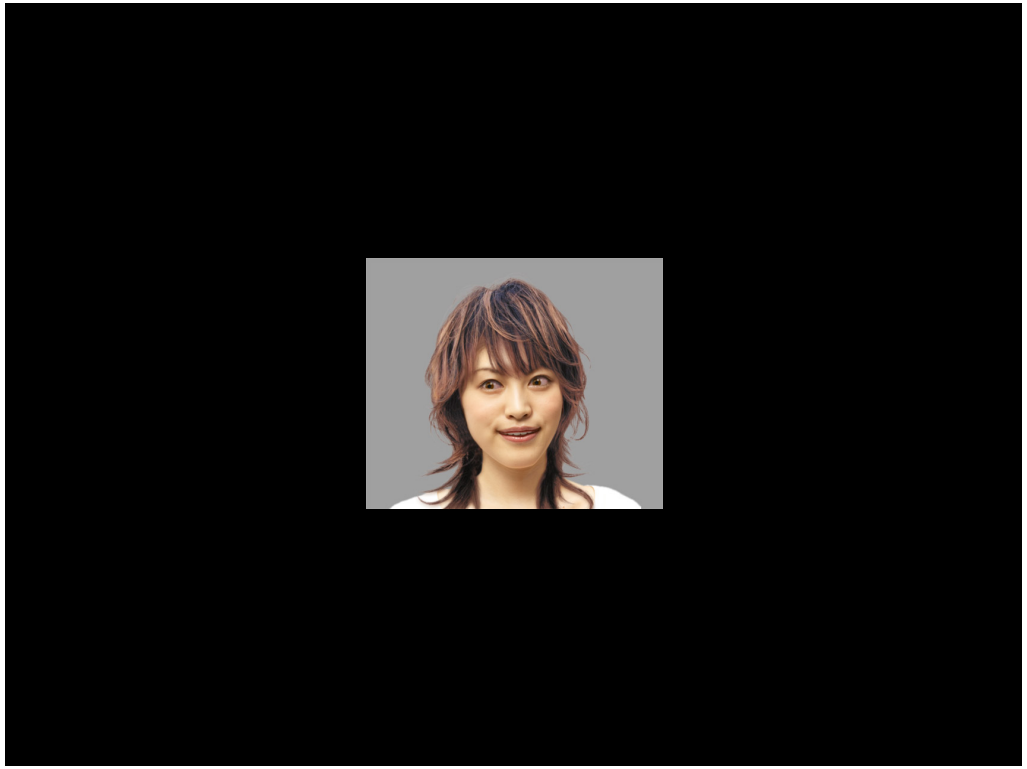
```ruby
class DosaSearchController < ApplicationController

  def new
    @search = DosaSearch.new
  end

  def show
    @search = DosaSearch.new(params[:search])
    @results = @search.results
  end

end
```

Antipattern: Slavish Adherence
(with zombies)



The Uncanny Valley

Masahiro Mori – 1970



Masahiro Mori in 1970

```ruby
class Person
  include Validatable
  validates_presence_of :name
  attr_accessor :name
end
```

```ruby
class Person
  include Validatable
  validates_presence_of :name, :times => 1
  attr_accessor :name
end
```

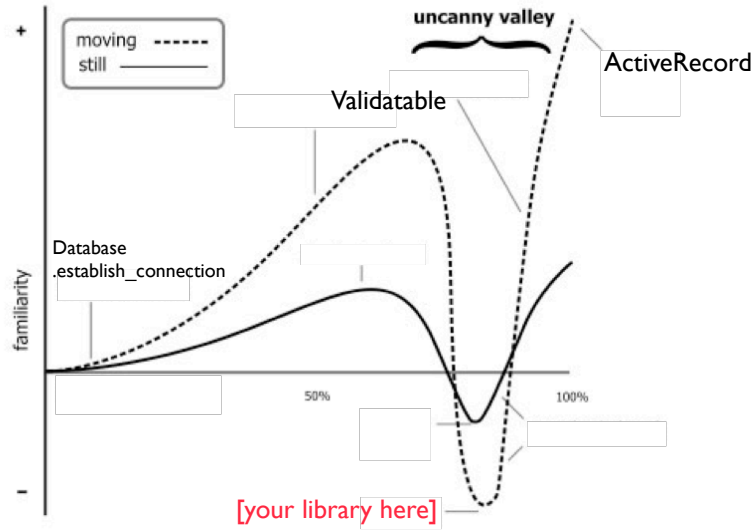
```ruby
class MortgageApplication
  include Validatable
  validates_presence_of :ssn, :groups => [:saving, :underwriting]
  validates_presence_of :name, :groups => :underwriting
  attr_accessor :name, :ssn
end
```

```ruby
class MortgageApplication
  include Validatable
  validates_presence_of :ssn, :groups => [:saving, :underwriting]
  validates_presence_of :name, :groups => :underwriting
  attr_accessor :name, :ssn
end

>> application = MortgageApplication.new
>> application.ssn = 377990118
>> application.valid_for_saving?
=> true
application.valid_for_underwriting?
=> false
```

Validations! Everywhere!

[your library here]

Validations was a great gem.
It solved a real need.


+
-
familiarity
moving
still
uncanny valley
ActiveRecord
Validatable
Database
.establish_connection
50%
100%
[your library here]

http://www.flickr.com/photos/vogelium/2105821119

Post-Rails?

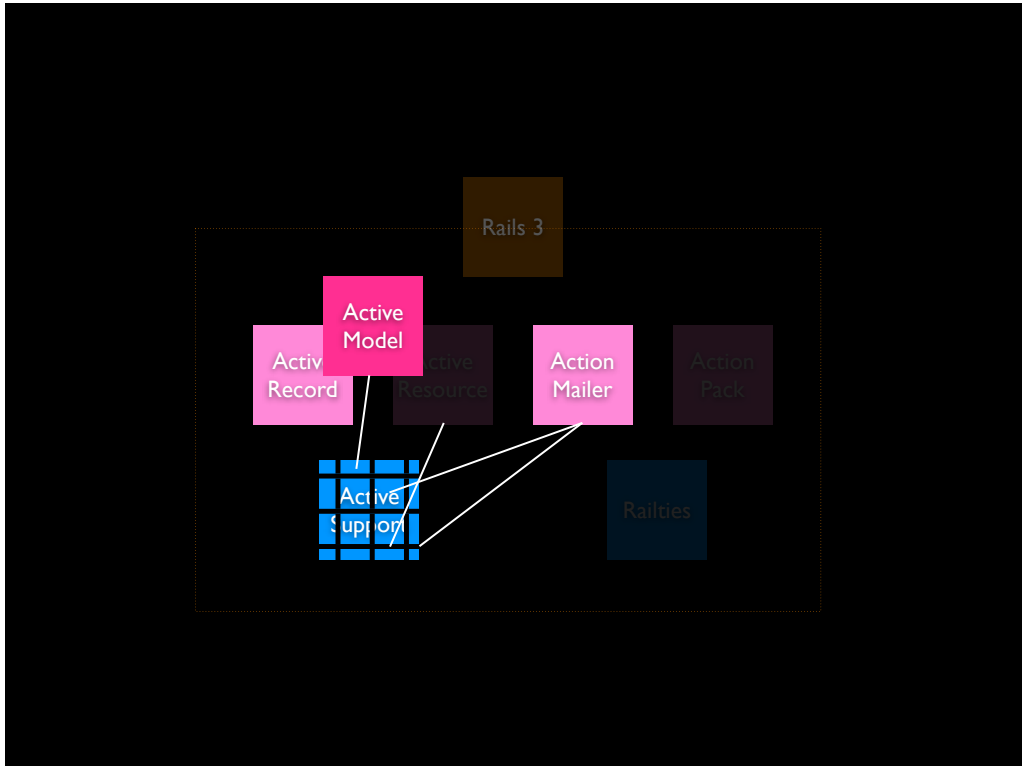
Rails isn't going anywhere

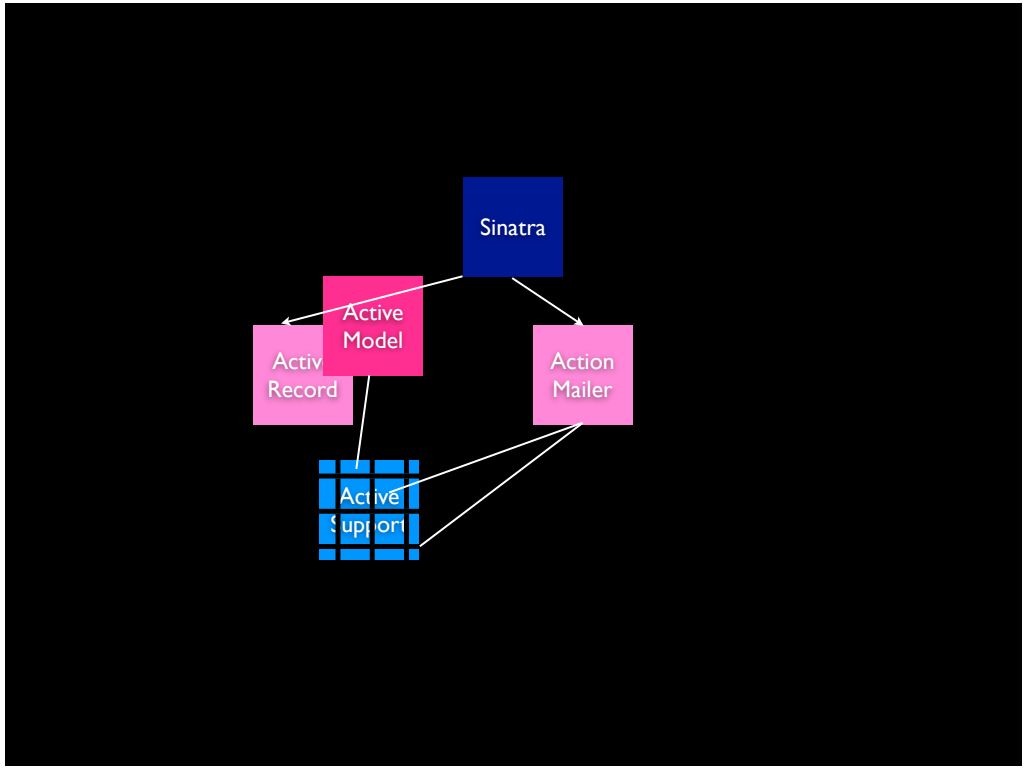
What is Ruby on Rails anyway?
Routing?
Requires?
A set of conventions?

Post-Rails:
use individual pieces without being tied to the whole

ActiveRecord anywhere
First-class Sinatra
Unintrusive ActiveSupport

More generally, code is now more portable


Rails 3
Active Model
Action Mailer
Railties

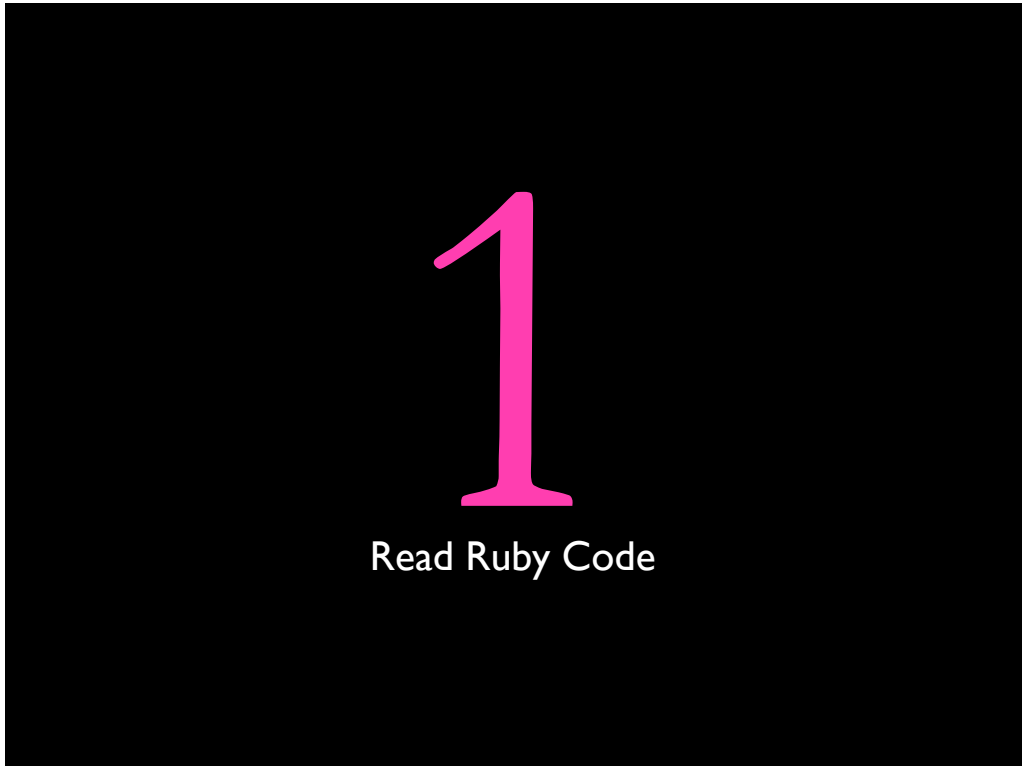

Sinatra
Active Model
Action Mailer

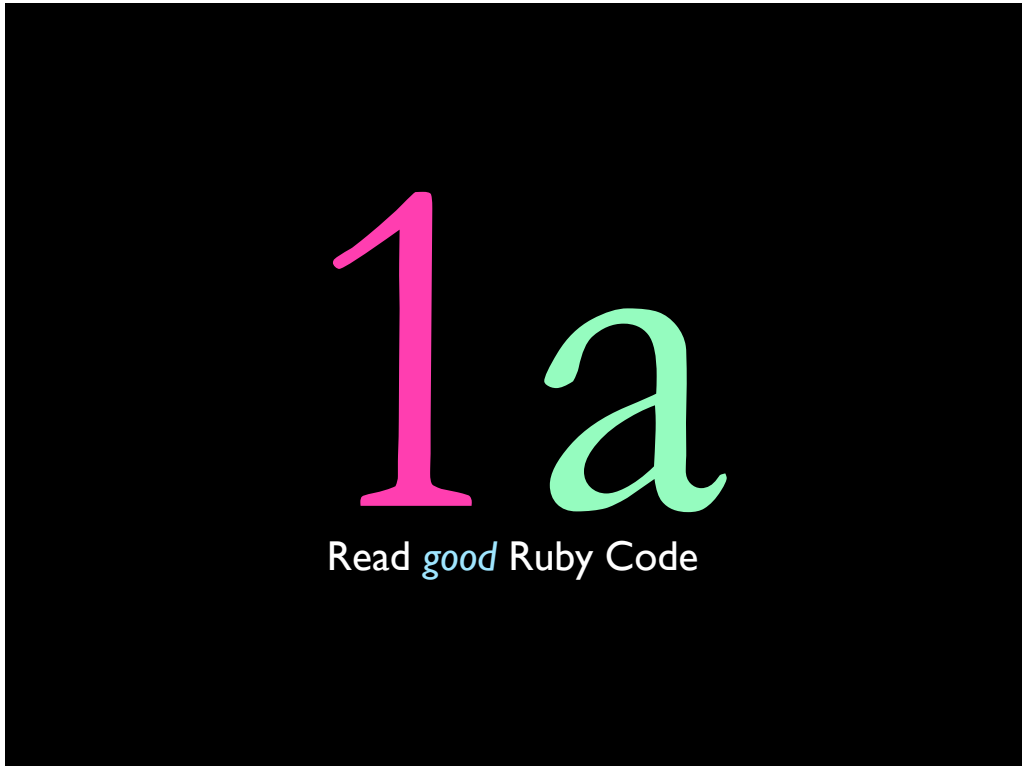
Padrino

What happens when stuff gets modular

How to learn Ruby
(secret)

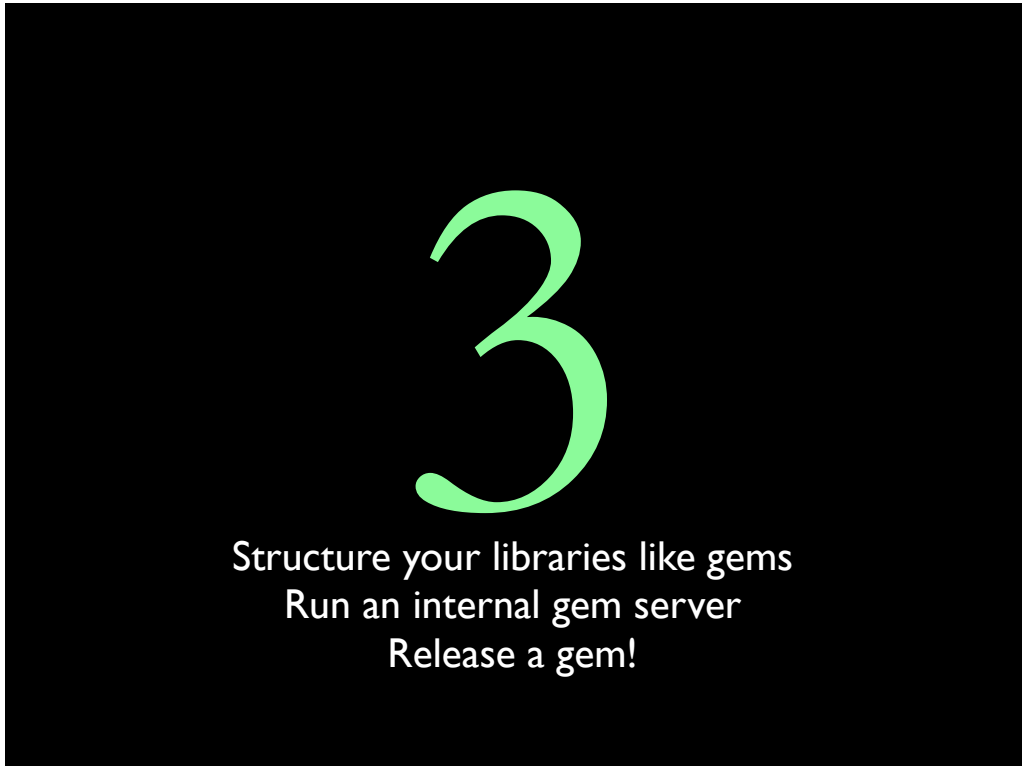
“Dude, this is RubyConf. How many people here do you really think don’t already know Ruby?”


1
Read Ruby Code


1a
Read good Ruby Code


2
Write Ruby Code


2a
Write Ruby Code
(without Rails)


3
Structure your libraries like gems
Run an internal gem server
Release a gem!

Robots do what every other robot does

Don’t be a robot!

Brian Guthrie
ThoughtWorks®
bguthrie@thoughtworks.com
http://twitter.com/bguthrie

questions?

http://www.flickr.com/photos/jonnya/4212907371/