Before I Begin...
Double Dream Hands:
SO INTENSE

Wednesday, May 18, 2011
ZOMG!!!
HAPPY
CONF
Aaron Patterson
WWFMD?
Corey Haines
Corey Haines

Eric Ries
Venture Advisor

Aaron Patterson
Señor Software Engineer
RailsConf is the largest official conference dedicated to everything Ruby on Rails. Featuring keynotes, sessions, tutorials, events, and of course lots of hallway hacking, RailsConf is the meeting place for the Ruby on Rails community.

**Featured Speakers**

David Heinemeier Hansson 37signals

Guy Steele Oracle Labs

Richard Gabriel IBM Research

Chad Dickerson ETSY

Aaron Patterson AT&T Interactive

Glenn Vanderburg InfoEther

**Diamond Sponsors**

**Sponsorship Opportunities**

For information on exhibiting and sponsorship opportunities at the conference, contact Yvonne Romaine at yromaine@oreilly.com

**Stay Connected**

Twitter

Facebook
RailsConf is the largest official conference dedicated to everything Ruby on Rails. Featuring keynotes, sessions, tutorials, events, and of course lots of hallway hacking, RailsConf is the meeting place for the Ruby on Rails community.

**Featured Speakers**

- **David Heinemeier Hansson** 37signals
- **Guy Steele** Oracle Labs
- **Richard Gabriel** IBM Research
- **Chad Dickerson** ETSY
- **Aaron Patterson** AT&T Interactive
- **Glenn Vanderburg** InfoEther

**Diamond Sponsors**

- livingSocial

**Learn New Skills, Techniques, and Ideas**

**Stay Connected**

- Twitter
- Facebook
Señor Software Engineer
Señor Software Engineer
Agenda

New Features

Real Talk!

Development Pro Tips™
Implementation and thoughts on API design

I will be very TECHNICAL
I will be very CRITICAL because my audience is smart because I care.
New Features!
git log --since="1 year ago"
API Strategy
rails generate VC
def halts?(function)
Prepared Statements
select * from 'users'
where id = 10
select * from 'users'
where id = ?
DATABASE

Records

RAILS APP
4 Steps!
4 Steps!
2 Steps!
Impact
SQLite3
SQLite3 Simple Prepared Statement

\begin{align*}
y &= 8.556 \times 10^{-5} \times 2.2868x \\
y &= 0.0002 \times 2.187x
\end{align*}

Number of Queries

- Cached
- Non-Cached
- Trend 1
- Trend 2
8665 q/s
12987 q/s
$\Delta \ 4322 \text{ q/s}$
SQLite3 Complex SQL Statement

$y = 0.0003e^{2.2227x}$

$y = 0.0029e^{2.3111x}$
339 q/s
4184 q/s
Δ 3845 q/s
PostgreSQL
$y = 0.0002e^{2.3018x}$

$y = 0.0004e^{2.1424x}$
Δ 924 q/s
The graph shows the performance of PostgreSQL Complex Query with and without cache. The equations for the trends are:

1. For the cached trend: $y = 0.0005e^{2.173x}$
2. For the no cache trend: $y = 0.0047e^{2.2153x}$
307 q/s
3322 q/s
3015 q/s
6410 q/s
5154 q/s
$\Delta -1256 \text{ q/s}$
626 q/s
1198 q/s
△ 572 q/s
No Query Planning
Two Network Roundtrips
Problem?
```ruby
select updated_at from pirates
```

```
['2011-05-13 22:03:55']
```

```
"2011-05-13 22:03:55"
```
select updated_at from pirates

Prepared Statement


[["2011-05-13 22:03:55"]]

Wednesday, May 18, 2011
select updated_at from pirates


[["2011-05-13 22:03:55"]]

Regular Statement
API Changes
User.find 1
User.find 1
Serialized Attributes
class User < ActiveRecord::Base
  serialize :preferences
end
user = User.find 1
user.preferences = { :hello => 'world!' }
user.save!
Why YAML?
class User < ActiveRecord::Base
  serialize :preferences, Base64Encoder.new
end
class Base64Encoder
  def load(value)
    return unless value
    value.unpack('m').last
  end

  def dump(text)
    [text].pack('m')
  end
end
class JSONEncoder
  def load(value)
    return unless value
    JSON.load value
  end

  def dump(text)
    JSON.dump(text)
  end
end
class MarshalEncoder
  def load(value)
    return unless value
    Marshal.load value
  end

  def dump(text)
    Marshal.dump(text)
  end
end
class XMLEncoder
  def load(value)
    return unless value
    Nokogiri.XML value
  end

  def dump(doc)
    doc.to_xml
  end
end
Official NoSQL Support
Official NoSQL Support

IT'S OFFICIAL!

Wednesday, May 18, 2011
irb(main):020:0> }

}
2 Caveats
PostgreSQL Only
You might hurt yourself
class User < ActiveRecord::Base
  class HStore
    def load value
      return unless value
      eval "#{value}";
    end

    def dump value
      return unless value
      value.map { |xs| xs.join '=>' }.join ', '
    end
  end

  serialize :preferences, HStore.new
end
class BCryptCoder
  def load(value)
    return unless value
    BCrypt::Password.new value
  end

  def dump(value)
    BCrypt::Password.create(value).to_s
  end
end
class User < ActiveRecord::Base
  serialize :crazy_column, BCryptCoder.new
end
## stdlib serialization API

<table>
<thead>
<tr>
<th></th>
<th>Dumping</th>
<th>Loading</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>YAML</strong></td>
<td>dump</td>
<td>load</td>
</tr>
<tr>
<td><strong>JSON</strong></td>
<td>dump</td>
<td>load</td>
</tr>
<tr>
<td><strong>Marshal</strong></td>
<td>dump</td>
<td>load</td>
</tr>
</tbody>
</table>
YOU choose

Storage type

Column name
Good Abstractions yield Good Features
CONSISTENCY is FREEDOM and FLEXIBILITY
has_secure_password
class User < ActiveRecord::Base
  has_secure_password
end
SECURITY!

user = User.find 1
user.password = 'lolwut'
user.save!
Advantages

- Introduces a new method
- Saves password in `password_digest`
- Uses BCrypt
Disadvantages

❤️ Introduces a new method
❤️ Saves password in `password_digest`
❤️ Uses BCrypt

API is not flexible
Not reusable
GO GREEN!
Write Something

REUSABLE!
Streaming Responses
Rack API

class MyApp
  def call(env)
    [200, { 'X-ZOMG' => 'hi' }, ['web page!']]  
  end
end
class PoniesController
  def call(env)
    body << render(:content)
    body << render(:layout)

    [200, {}, body]
  end
end
Delayed Evaluation
Body\#each
class DelayedBody
  def each
    yield render(:layout)
    yield render(:content)
  end
end

class PoniesController
  def call(env)
    [200, {}, DelayedBody.new]
  end
end
Middleware

Middleware
Middleware 😞
Middleware Chain

Connection Manager

Application
Middleware Chain

call(env)

Connection Manager

call(env)

Application
Middleware Chain

[200, {}, body]

Connection Manager

[200, {}, body]

Application
Connection Manager

When ERb is evaluated, db connection does not exist
Connection Manager

Open DB Connection

When ERb is evaluated, db connection does not exist
Connection Manager

Open DB Connection

Delegate

When ERb is evaluated, db connection does not exist
Connection Manager

Open DB Connection

Delegate

Close DB Connection

When ERb is evaluated, db connection does not exist
Connection Manager

- Open DB Connection
- Delegate
- Close DB Connection
- Return Delegate Values

When ERb is evaluated, db connection does not exist.
Problem?
Problem?
class ConnectionBodyProxy
  def initialize(delegate)
    @delegate = delegate
  end
  def each(&blk)
    @delegate.each(&blk)
  end
  def close
    # close connection
  end
end
class ConnectionManager
  def initialize(app); @app = app; end

  def call(env)
    # OPEN CONNECTION
    response = @app.call(env)
    response
  end
end
~25 middleware
Embrace Diversity
Types of Middleware

- Generators (our application)
- Filters (gzip, etc)
- Lifecycle Handlers (connection management)
Real Talk!
Rails is getting slower
Time for N Requests

Requests vs. Time

Rails 2.3

Rails 3.0

Wednesday, May 18, 2011
Avg Requests / time

- Rails 2.3
- Rails 3.0

Requests:
- 10: 162.5
- 100: 387.5
- 1000: 500
- 10000: 1000

Average Requests over time:
- Rails 2.3:
  - 10: 162.5
  - 100: 387.5
  - 1000: 500
  - 10000: 1000
- Rails 3.0:
  - 10: 10
  - 100: 100
  - 1000: 1000
  - 10000: 10000
Middleware
Increased

Wednesday, May 18, 2011
GC Pressure
Rails 2.3 Stack
51 deep
Rails 3.0 Stack
60 deep
Rails 3.1 Stack
67 deep
TL;DR
Embrace Diversity
We need a change
class Application
  def call(request, response)
    response.write(request, "hello", 200, {})

    10.times do |body|
      response.write(request, body)
    end

    response.close
  end
end
class Filter
  def initialize(filter)
    @filter = filter
  end

  def write(request, body, status = nil, headers = nil)
    @filter.write(request, body, status, headers)
  end

  def close
    @filter.close
  end
end
class Handler
  def before(request)
    # Connect to database
  end
  
  def after(request)
    # Disconnect
  end
end
git log --until="1 year from now"
We Must Change
Memory
Stack Depth
Backwards
Compat
Development
Pro Tip™
Final Words
A Message for Rails-Core

- We will not be here forever
- Rails will outlive us
- Encourage refactoring
- Stop "yes men"

Wednesday, May 18, 2011
Not all features are Tangible
Good Abstraction yield Reusable Code
Reusable Code
yield
New Features
Your Homework:
GO GREEN!
Refactor Rails!
GO FORTH AND CODE!
THANKS! ♥
QUESTIONS?