@engineyard

Rails 4
Changes and New Features

J. Austin Hughey
Field Application Engineer
Engine Yard

@jaustinhughey
@openhackatx

Overview

- Multiple separations and deprecations
- HTTP semantics changes
- More security features
- Lots of cool PostgreSQL integration

Changes

- Ruby 1.9.3 minimum
  - 2.0 recommended
  - Rails 5.x will require >= 2.0 so might as well upgrade now
- Many deprecated items are now separate gems
  - May not be compatible with Rails 4.1
  - Use only as a bridge; make sure to get rid of uses of the old stuff
- PATCH verb (instead of PUT)
  - PATCH :update, article: { author: “foo”, title: “bar”, body: “blah” }
- “Strong Parameters”
- Thread Safe by default

Saying goodbye...

QuickTime™ and a
GIF decompressor
are needed to see this picture.

Removed in 4.0

- vendor/plugins - use gems instead
- ActiveResource
  - https://github.com/rails/activeresource
- Hash-based/dynamic finder methods
  - https://github.com/rails/activerecord-deprecated_finders
- ActiveRecord::SessionStore
  - https://github.com/rails/activerecord-session_store
- Observers
  - https://github.com/rails/rails-observers
- Page and Action Caching
  - https://github.com/rails/actionpack-action_caching
  - https://github.com/rails/actionpack-page_caching

PATCH

HTTP PATCH

- HTTP says that a PUT request represents a complete representation of a resource.
- Ergo, we've been using PUT wrong. We rarely pass a whole resource to a controller on edits - just the changed bits.
- Solution: use PATCH instead. PATCH sends up just what's changed.

THREAD SAFETY

- config.thread_safe is on by default
- Still should try a truly threaded interpreter/server
- JRuby/Rubinius + Puma, Passenger Enterprise

STRONG PARAMETERS

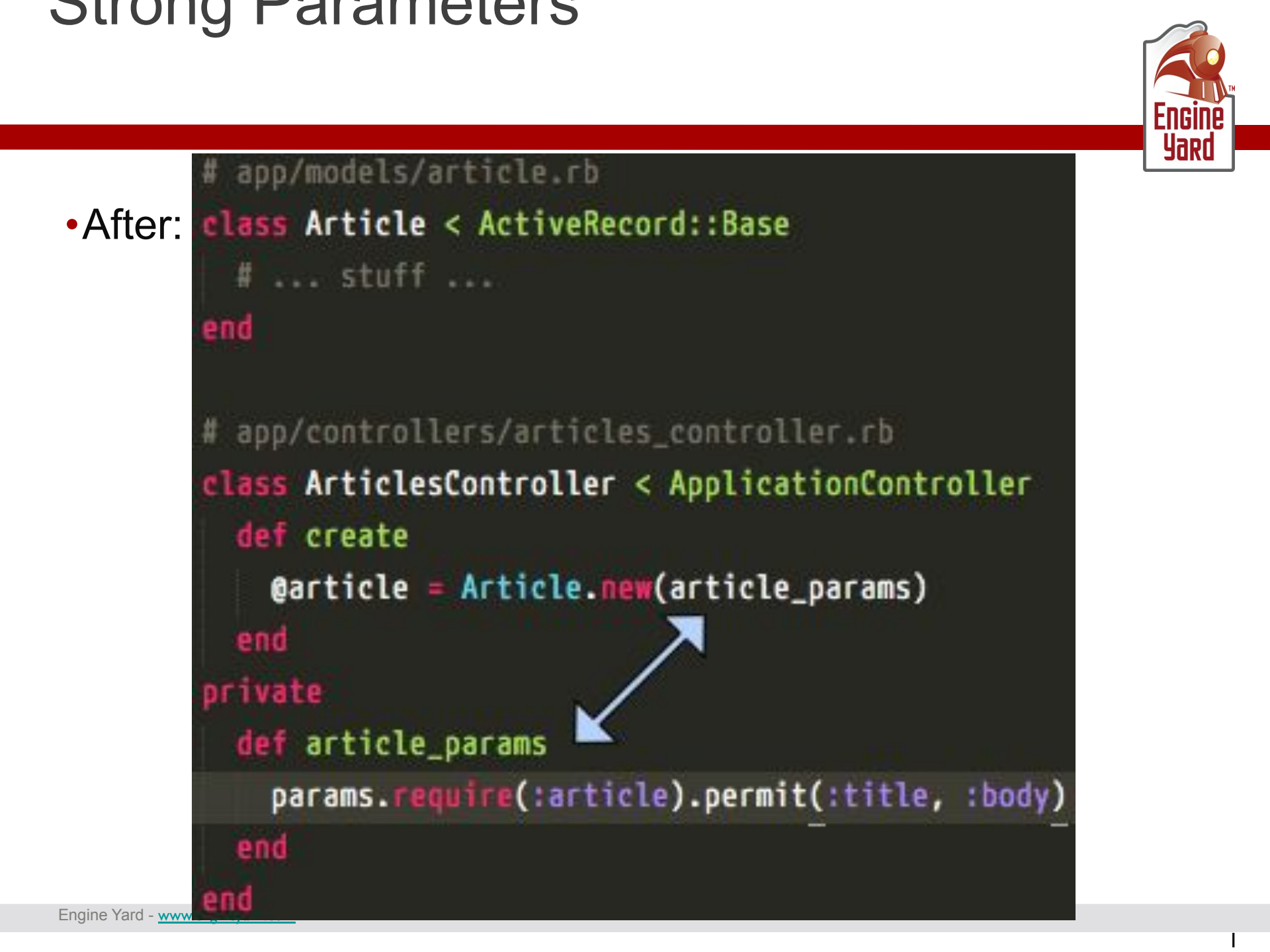
Strong Parameters

- Before:

```
class Article < ActiveRecord::Base
  attr_accessible :title, :body
end

article = Article.new(params[:article])
```

Strong Parameters



- After:

```ruby
# app/models/article.rb
class Article < ActiveRecord::Base
  # ... stuff ...
end

# app/controllers/articles_controller.rb
class ArticlesController < ApplicationController
  def create
    @article = Article.new(article_params)
  end
private
  def article_params
    params.require(:article).permit(:title, :body)
  end
end
```

Strong Parameters

- Why is this better?
  - Puts sanitization focus on user input vector - the controller
  - Frees up the developer to work with the data model uninhibited
- Criticisms:
  - Breaks the idea that you should be able to throw ANYTHING at an object and it knows what to do with it.
  - Nested attributes can be a pain in the rear.

Encrypted Cookies

- New cookie store: "encrypted_cookie_store"
- Now the default in Rails 4
- Encrypts cookies before being sent to the client, decrypts received cookies
- Prevents user tampering
- Not a complete security solution.
- MIGHT annoy the NSA.

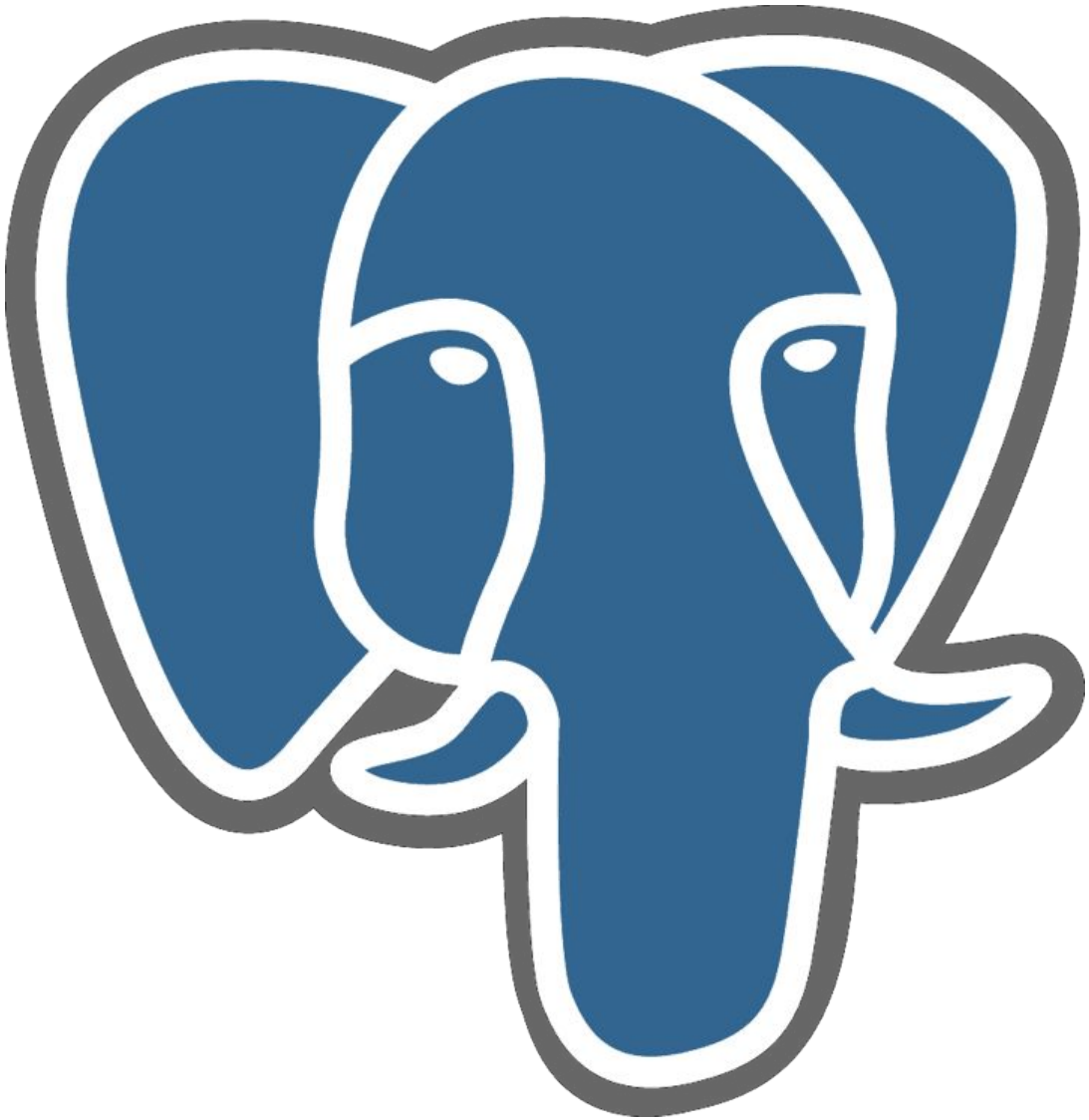
Image credit: Electronic Frontier Foundation - eff.org

Default Headers

Include default headers with each response coming from Rails.

```
config.action_dispatch.default_headers = {
'X-Frame-Options' => 'SAMEORIGIN',
'X-XSS-Protection' => '1; mode=block',
'X-Content-Type-Options' => 'nosniff'
}
```

THE ELEPHANT IN THE ROOM



Rails <3 PostgreSQL

- Rails 4 includes support for PostgreSQL datatypes:
  - hstore
  - arrays
  - INET
  - CIDR
  - MACADDR
  - uuid

PostgreSQL hstore

- CREATE EXTENSION hstore;
  - Or `enable_extension "hstore"` in migrations
- Like serialized columns, but more efficient (not a text field)
- GIST or GIN indexes
  - Read the PostgreSQL docs to figure out which is right for you
- Querying is a little weird
  - User.where(“preferences @> ‘theme=>black’”)
- Available in 3.2 through activerecord-postgres-hstore gem

PostgreSQL Array


```
create_table :foos do |t|
  t.integer  :int_array,    array: true
  t.string   :string_array, array: true
end

foo = Foo.new
foo.int_array = [1, 2, 3, 4, 5]
foo.save
```

INET, CIDR, MACADDR


```
create_table :networks do |t|
  t.cidr    :cidr_address
  t.inet    :ip_address
  t.macaddr :mac_address
end
```

- cidr, inet both come out as a native Ruby IPAddr object
- macaddr treated as a string

Using a UUID

- Enable the uuid-ossp extension
- create_table :name, id: :uuid { |t| ... }

```
class CreateUsers < ActiveRecord::Migration
  def change
    enable_extension "uuid-ossp"

    create_table :users, id: :uuid do |t|
      t.string          :email,           null: false
      t.string          :password_digest, null: false
      t.string          :name,            null: false
      t.timestamps
    end

    # Throw down some indexes for quick searching
    add_index :users, :email
    add_index :users, :password_digest
    add_index :users, :name
  end
end
```

TURBOLINKS

ZOOM ZOOM!

Turbolinks

- Swaps out <body> contents with what should've been rendered by the server
- Avoids the need to reload all the CSS/JS again
- On by default, easily disabled
- Makes everything look faster

- CAVEAT EMPTOR:

May break some of your javascript

Various event listeners may need to be changed

Speed improvement depends on how much JS/CSS you have

Disabling Turbolinks

- Remove from Gemfile
- Remove from application.js
- `bundle`

https://github.com/rails/turbolinks

CACHE MONEY

Cache Digests

- Forget bumping version numbers in your cache.
- On application start, computes MD5 sum of cache content and stores the sum as a key; when the content changes, the MD5 sum changes thus invalidating the cache.

BEFORE

```
<% cache [‘v3’, comment] do %>
  My comment: <%= comment.body %>
<% end %>
```

AFTER

```
<% cache comment do %>
  My comment: <%= comment.body %>
<% end %>
```

NEW DEFAULT TEST LOCATIONS

APERTURE LABORATORIES

FOR SCIENCE

New Default Test Locations

Then	Now
test/units	test/models
test/units/helpers	test/helpers
test/functional	test/controllers
test/functional	test/mailers


LIVE STREAMING

Is it live?

- Stream response to the browser
- Needs multi-threaded application server
  - e.g. Puma, Thin, Passenger Enterprise
  - Putting it behind a non-GIL addled interpreter also advised
- Not a lot of examples in the wild yet
- May not work on IE. :-(

```
class MyController < ApplicationController  include
ActionController::Live  def index    100.times {
response.stream.write "hello world\n"    }
response.stream.close  endend
```

Example from http://tenderlovemaking.com/2012/07/30/is-it-live.html

Stuff NOT Shipping

- Background Queuing
- Asynchronous ActionMailer
- where.like / where.not_like

Upgrading

- PAY ATTENTION to deprecation warnings
- Have a *really* good set of tests and as high coverage as possible
- Take it in stages, by sprints
- 3.2 -> 4.0 will be easiest upgrade path

Thank You