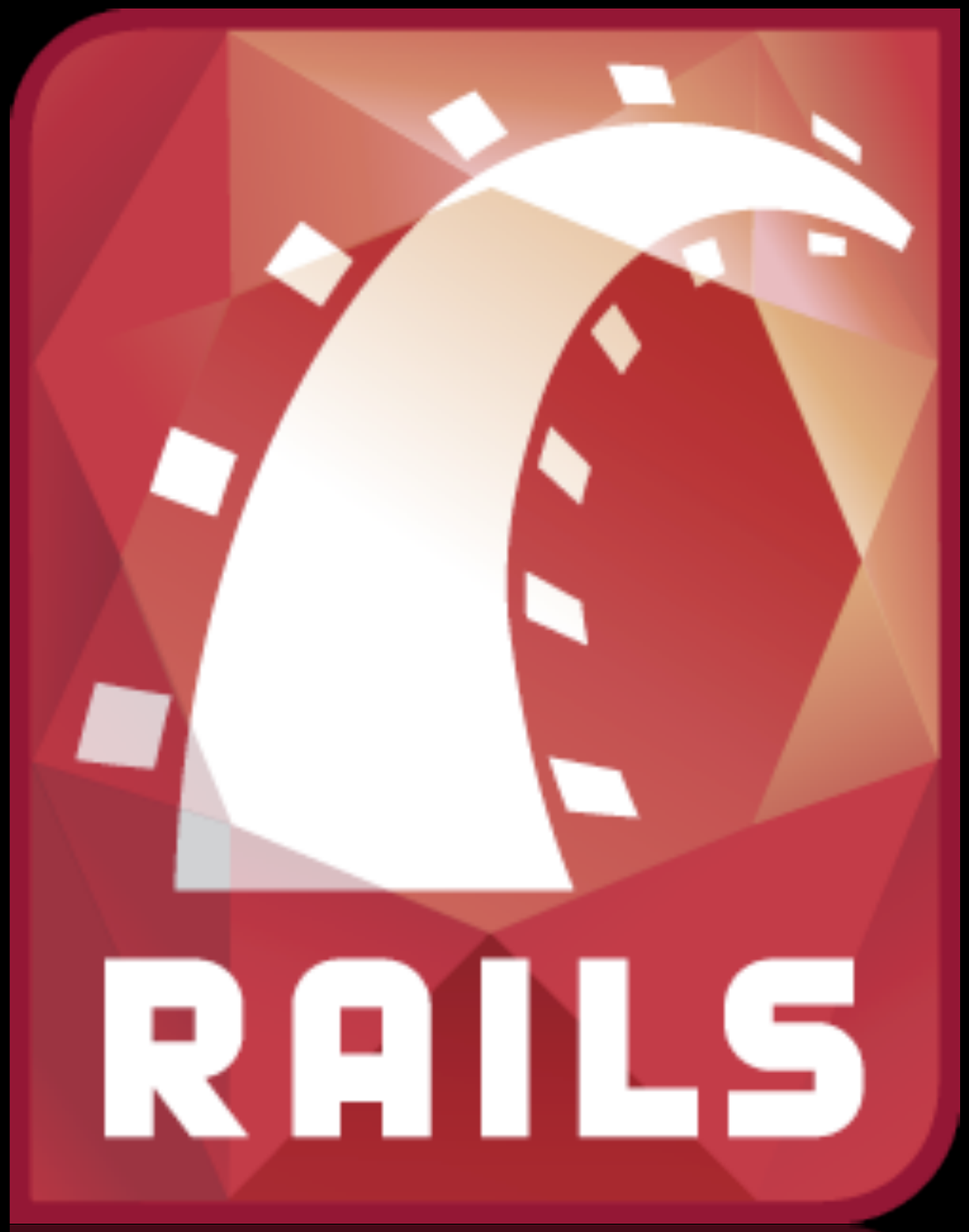
Merging

Merb into Rails

Me

Yehuda Katz


Engine
Yard™


RAILS

@carlhuda

Cloud

12/23

2008

Yehuda Katz

Yehuda Katz is a member of the **Ruby on Rails** core team, and lead developer of the **Merb** project. H
member of the **jQuery** Core Team, and a core contributor to **DataMapper**. He contributes to many ope
projects, like **Rubinius** and **Johnson**, and works on some he created himself, like **Thor**.

Rails and Merb Merge

December 23rd, 2008

Today is a fairly momentous day in the history of Ruby web frameworks. You will probably find the news I'm about to share with you fairly shocking, but I will attempt to explain the situation.

Before talking tech, and even going into the details of the announcement, I want to assure everyone that the incredible members of the thriving Merb community are top priority, and that this could not have been possible without every one of them.

Merb is an open, ever-changing project, and some its best ideas have come from not-core regular-Joe community members. It's gotten where it has because of the community, and the community will get us even further in the future. Your ideas, feedback and even complaints will be 100% welcome in the future, just as they have been in the past. I believe in the tremendous value an open community and just generally open attitude bring to the table, and am counting on those things to continue ushering in the future of Ruby.

On to the news: beginning today, the Merb team will be working with the Rails core team on a joint project. The plan is to merge in the things that made Merb different. This will make it possible to use Rails 3 for the same sorts of use-cases that were compelling for Merb users. Effectively, Merb 2 is Rails 3.

What does that mean exactly?

- Rails will become more modular, starting with a rails-core, and including the ability to opt in or out of specific components. We will focus on reducing coupling across Rails, and making it possible to replace parts of Rails without disturbing other parts. This is exactly what Merb means when it touts “modularity”.
- We will port all of our performance improvements into Rails. This includes architectural decisions that are big performance wins. This project will also include the creation of one or more benchmarking applications, so we can more clearly see what optimizations have real-world impact.
- As of Rails 3, Rails will have a defined public API with a test suite for that API. This was one of the major differentiators of Merb. This will allow users and plugin developers to have a clearer, more stable API to build against. It should also significantly reduce plugin breakage from release to release.
- Rails will be retrofitted to make it easy to start with a “core” version of Rails (like Merb’s current core generator), that starts with all modules out, and makes it easy to select just the parts that are important for your app. Of course, Rails will still ship with the “stack” version as the default (just as Merb does since 1.0), but the goal is to make it easy to do with Rails what people do with Merb today.
- Rails will be modified to more easily support DataMapper or Sequel as first-class ORMs. While ActiveRecord will ship as the default ORM, the plan is to make it drop-dead easy to drop in other ORMs without feature degradation (to the extent possible, of course).
- Rails will continue their recent embrace of Rack, which is a really exciting development in the Ruby community that Merb got in on early and which we believe will improve the state of modular, sharable logic between applications.
- In general, we will take a look at features in Merb that are not in Rails (the most obvious example is the more robust router) and find a way to bring them into Rails.

So how'd we do?

Rails will become more modular, starting with a rails-core, and including the ability to opt in or out of specific components.

ActiveSupport

active_support/core_ext/conversions.rb

```
module ActiveSupport
  module CoreExtensions
    module Array
      module Conversions
        def to_sentence(options = {})
        ...
          options.assert_valid_keys :words_connector,
            :two_words_connector,
            :last_word_connector, :locale
          ...
        end
      ...
    end
  end
end
```

active_support/core_ext/conversions.rb

```
module ActiveSupport
  module CoreExtensions
    module Array
      module Conversions
        def to_sentence(options = {})
        ...
          options.assert_valid_keys :words_connector,
            :two_words_connector,
            :last_word_connector, :locale
        ...
        end
      ...
    end
  end
end
```

Where?

active_support/core_ext/conversions.rb

```
require 'active_support/core_ext/hash/keys'
require 'active_support/core_ext/hash/reverse_merge'
require 'active_support/inflector'

class Array
  def to_sentence(options = {})
    ...
    options.assert_valid_keys :words_connector,
      :two_words_connector, :last_word_connector, :locale
    ...
  end
  ...
end
```

active_support/core_ext/conversions.rb

```ruby
require 'active_support/core_ext/hash/keys'
require 'active_support/core_ext/hash/reverse_merge'
require 'active_support/inflector'

class Array
  def to_sentence(options = {})
    ...
    options.assert_valid_keys :words_connector,
      :two_words_connector, :last_word_connector, :locale
    ...
  end
  ...
end
```

```
require 'active_support/core_ext/conversions'
```

abstract_controller/logging.rb

```
require 'active_support/core_ext/logger'
require 'active_support/benchmarkable'

module AbstractController
  module Logger
    ...
  end
end
```

abstract_controller/logging.rb

```
require 'active_support/core_ext/logger'
require 'active_support/benchmarkable'

module AbstractController
  module Logger
    ...
  end
end
```

Rails will become more modular, starting with a rails-core, and including the ability to opt in or out of specific components.

```
action_controller$ ls
assertions
base.rb
benchmarking.rb
caching
caching.rb
cgi_ext
cgi_ext.rb
cgi_process.rb
cookies.rb
dispatcher.rb
failsafe.rb
filters.rb
flash.rb
headers.rb
helpers.rb
http_authentication.rb
integration.rb
layout.rb
middleware_stack.rb
middlewares.rb
mime_responds.rb
mime_type.rb
mime_types.rb
params_parser.rb
performance_test.rb
polymorphic_routes.rb
rack_lint_patch.rb
record_identifier.rb
reloader.rb
request.rb
request_forgery_protection.rb
rescue.rb
resources.rb
response.rb
routing
routing.rb
session
session_management.rb
status_codes.rb
streaming.rb
string_coercion.rb
templates
test_case.rb
test_process.rb
translation.rb
uploaded_file.rb
url_rewriter.rb
vendor
verification.rb
```

```
action_controller$ ls
assertions
base.rb
benchmarking.rb
caching
caching.rb
cgi_ext
cgi_ext.rb
cgi_process.rb
cookies.rb
dispatcher.rb
failsafe.rb
filters.rb
flash.rb
headers.rb
helpers.rb
http_authentication.rb
integration.rb
layout.rb
middleware_stack.rb
middlewares.rb
mime_responds.rb
mime_type.rb
mime_types.rb
params_parser.rb
performance_test.rb
polymorphic_routes.rb
rack_lint_patch.rb
record_identifier.rb
reloader.rb
request.rb
request_forgery_protection.rb
rescue.rb
resources.rb
response.rb
routing
routing.rb
session
session_management.rb
status_codes.rb
streaming.rb
string_coercion.rb
templates
test_case.rb
test_process.rb
translation.rb
uploaded_file.rb
url_rewriter.rb
vendor
verification.rb
```

base.rb

```
action_controller$ ls
assertions
base.rb
benchmarking.rb
caching
caching.rb
cgi_ext
cgi_ext.rb
cgi_process.rb
cookies.rb
dispatcher.rb
failsafe.rb
filters.rb
flash.rb
headers.rb
helpers.rb
http_authentication.rb
integration.rb
layout.rb
middleware_stack.rb
middlewares.rb
mime_responds.rb
mime_type.rb
mime_types.rb
params_parser.rb
performance_test.rb
polymorphic_routes.rb
rack_lint_patch.rb
record_identifier.rb
reloader.rb
request.rb
request_forgery_protection.rb
rescue.rb
resources.rb
response.rb
routing
routing.rb
session
session_management.rb
status_codes.rb
streaming.rb
string_coercion.rb
templates
test_case.rb
test_process.rb
translation.rb
uploaded_file.rb
url_rewriter.rb
vendor
verification.rb
```

```
action_controller$ ls
assertions
base.rb
benchmarking.rb
caching
caching.rb
cgi_ext
cgi_ext.rb
cgi_process.rb
cookies.rb
dispatcher.rb
failsafe.rb
filters.rb
flash.rb
headers.rb
helpers.rb
http_authentication.rb
integration.rb
layout.rb
middleware_stack.rb
middlewares.rb
mime_responds.rb
mime_type.rb
mime_types.rb
params_parser.rb
performance_test.rb
polymorphic_routes.rb
rack_lint_patch.rb
record_identifier.rb
reloader.rb
request.rb
request_forgery_protection.rb
rescue.rb
resources.rb
response.rb
routing
routing.rb
session
session_management.rb
status_codes.rb
streaming.rb
string_coercion.rb
templates
test_case.rb
test_process.rb
translation.rb
uploaded_file.rb
url_rewriter.rb
vendor
verification.rb
```

cgi_ext.rb

```
action_controller$ ls
assertions
base.rb
benchmarking.rb
caching
caching.rb
cgi_ext
cgi_ext.rb
cgi_process.rb
cookies.rb
dispatcher.rb
failsafe.rb
filters.rb
flash.rb
headers.rb
helpers.rb
http_authentication.rb
integration.rb
layout.rb
middleware_stack.rb
middlewares.rb
mime_responds.rb
mime_type.rb
mime_types.rb
params_parser.rb
performance_test.rb
polymorphic_routes.rb
rack_lint_patch.rb
record_identifier.rb
reloader.rb
request.rb
request_forgery_protection.rb
rescue.rb
resources.rb
response.rb
routing
routing.rb
session
session_management.rb
status_codes.rb
streaming.rb
string_coercion.rb
templates
test_case.rb
test_process.rb
translation.rb
uploaded_file.rb
url_rewriter.rb
vendor
verification.rb
```

```
action_controller$ ls
assertions
base.rb
benchmarking.rb
caching
caching.rb
cgi_ext
cgi_ext.rb
cgi_process.rb
cookies.rb
dispatcher.rb
failsafe.rb
filters.rb
flash.rb
header.rb
helpers.rb
http_authentication.rb
integration.rb
layout.rb
middleware_stack.rb
middlewares.rb
mime_responds.rb
mime_type.rb
mime_types.rb
params_parser.rb
performance_test.rb
polymorphic_routes.rb
rack_lint_patch.rb
record_identifier.rb
reloader.rb
request.rb
request_forgery_protection.rb
rescue.rb
resources.rb
response.rb
routing
routing.rb
session
session_management.rb
status_codes.rb
streaming.rb
string_coercion.rb
templates
test_case.rb
test_process.rb
translation.rb
uploaded_file.rb
url_rewriter.rb
vendor
verification.rb
```

middlewares.rb

action_mailer/base.rb

```
module ActionMailer
 class Base
   include AdvAttrAccessor,
           PartContainer,
           Quoting,
           Utils

    if Object.const_defined?(:ActionController)
      include ActionController::UrlWriter
      include ActionController::Layout
    end
    ...
  end
end
```

helpers.rb

:(

ActionController

Dispatcher

Rack Controller

Generic Controller

Dispatcher
ActionDispatch

Rack Controller
ActionController

Generic Controller
AbstractController

```
action_controller$ ls
base.rb                    caching
caching.rb                 deprecated
deprecated.rb              dispatch
legacy                     metal
metal.rb                   middleware.rb
notifications.rb           testing
polymorphic_routes.rb      url_rewriter.rb
record_identifier.rb       translation.rb
vendor
```

```
action_dispatch$ ls
http
middleware
routing
routing.rb
testing
```

```
abstract_controller$ ls
base.rb
callbacks.rb
exceptions.rb
helpers.rb
layouts.rb
logger.rb
rendering_controller.rb
```

```
module ActionMailer
  class Base
    include AdvAttrAccessor,
            PartContainer,
            Quoting,
            Utils

    include AbstractController::RenderingController
    include AbstractController::LocalizedCache
    include AbstractController::Layouts

    include AbstractController::Helpers
    ...
  end
end
```

```
require "action_dispatch"
```

```
require "action_dispatch"
require "abstract_controller"
```

```
require "action_dispatch"
require "abstract_controller"
require "action_controller"
```

Dependencies

action_controller/base.rb

```ruby
def render(options = nil, extra_options = {}, &block) #:doc:
  raise DoubleRenderError, "Can only render or redirect once per action"
if performed?

  validate_render_arguments(options, extra_options, block_given?)

  if options.nil?
    options = { :template => default_template, :layout => true }
  elsif options == :update
    options = extra_options.merge({ :update => true })
  elsif options.is_a?(String) || options.is_a?(Symbol)
    case options.to_s.index('/')
    when 0
      extra_options[:file] = options
    when nil
      extra_options[:action] = options
    else
      extra_options[:template] = options
    end

    options = extra_options
  elsif !options.is_a?(Hash)
    extra_options[:partial] = options
    options = extra_options
  end

  layout = pick_layout(options)
  response.layout = layout.path_without_format_and_extension if layout
  logger.info("Rendering template within #
{layout.path_without_format_and_extension}") if logger && layout

  if content_type = options[:content_type]
    response.content_type = content_type.to_s
  end

  if location = options[:location]
    response.headers["Location"] = url_for(location)
  end

  if options.has_key?(:text)
    text = layout ? @template.render(options.merge(:text => options
[:text], :layout => layout)) : options[:text]
    render_for_text(text, options[:status])

  else
    if file = options[:file]
      render_for_file(file, options[:status], layout, options[:locals] ||
{})

    elsif template = options[:template]
      render_for_file(template, options[:status], layout, options[:locals]
|| {})

    elsif inline = options[:inline]
      render_for_text(@template.render(options.merge(:layout => layout)),
options[:status])

    elsif action_name = options[:action]
      render_for_file(default_template(action_name.to_s), options
[:status], layout)

    elsif xml = options[:xml]
      response.content_type ||= Mime::XML
      render_for_text(xml.respond_to?(:to_xml) ? xml.to_xml : xml, options
[:status])

    elsif js = options[:js]
      response.content_type ||= Mime::JS
      render_for_text(js, options[:status])

    elsif options.include?(:json)
      json = options[:json]
      json = ActiveSupport::JSON.encode(json) unless json.is_a?(String)
      json = "#{options[:callback]}(#{json})" unless options
[:callback].blank?
      response.content_type ||= Mime::JSON
      render_for_text(json, options[:status])

    elsif options[:partial]
      options[:partial] = default_template_name if options[:partial] ==
true
      if layout
        render_for_text(@template.render(:text => @template.render
(options), :layout => layout), options[:status])
      else
        render_for_text(@template.render(options), options[:status])
      end

    elsif options[:update]
      @template.send(:_evaluate_assigns_and_ivars)

      generator =
ActionView::Helpers::PrototypeHelper::JavaScriptGenerator.new(@template,
&block)
      response.content_type = Mime::JS
      render_for_text(generator.to_s, options[:status])

    elsif options[:nothing]
      render_for_text(nil, options[:status])

    else
      render_for_file(default_template, options[:status], layout)
    end
  end
end
```

action_controller/base.rb

```
def render(options = nil, extra_options = {}, &block) #:doc:
  raise DoubleRenderError, "Can only render or redirect once per action"
if performed?

  validate_render_arguments(options, extra_options, block_given?)

  if options.nil?
    options = { :template => default_template, :layout => true }
  elsif options == :update
    options = extra_options.merge({ :update => true })
  elsif options.is_a?(String) || options.is_a?(Symbol)
    case options.to_s.index('/')
    when 0
      extra_options[:file] = options
    when nil
      extra_options[:action] = options
    else
      extra_options[:template] = options
    end

    options = extra_options
  elsif options.is_a?(Hash)
    extra_options = options.dup
    options = extra_options
  end

  layout = pick_layout(options)
  response.layout = layout.path_without_format_and_extension if layout
  logger.info("Rendering template within #
{layout.path_without_format_and_extension}") if logger && layout

  if content_type = options[:content_type]
    response.content_type = content_type.to_s
  end

  if location = options[:location]
    response.headers["Location"] = url_for(location)
  end

  if options.has_key?(:text)
    text = layout ? @template.render(options.merge(:text => options
[:text], :layout => layout)) : options[:text]
    render_for_text(text, options[:status])

  else
    if file = options[:file]
      render_for_file(file, options[:status], layout, options[:locals] ||
{})

    elsif template = options[:template]
      render_for_file(template, options[:status], layout, options[:locals]
|| {})

    elsif inline = options[:inline]
      render_for_text(@template.render(options.merge(:layout => layout)),
options[:status])

    elsif action_name = options[:action]
      render_for_file(default_template(action_name.to_s), options
[:status], layout)

    elsif xml = options[:xml]
      response.content_type ||= Mime::XML
      render_for_text(xml.respond_to?(:to_xml) ? xml.to_xml : xml, options
[:status])

    elsif js = options[:js]
      response.content_type ||= Mime::JS
      render_for_text(js, options[:status])

    elsif options.include?(:json)
      json = options[:json]
      unless json.is_a?(String)
        json = json.to_json(options)
      end
      json = "#{options[:callback]}(#{json})" unless options[:callback].blank?
      response.content_type = Mime::JSON
      render_for_text(json, options[:status])

    elsif options[:partial]
      options[:partial] = default_template_name if options[:partial] ==
true
      if layout
        render_for_text(@template.render(:text => @template.render
(options), :layout => layout), options[:status])
      else
        render_for_text(@template.render(options), options[:status])
      end

    elsif options[:update]
      @template.send(:_evaluate_assigns_and_ivars)

      generator =
ActionView::Helpers::PrototypeHelper::JavaScriptGenerator.new(@template,
&block)
      response.content_type = Mime::JS
      render_for_text(generator.to_s, options[:status])

    elsif options[:nothing]
      render_for_text(nil, options[:status])

    else
      render_for_file(default_template, options[:status], layout)
    end
  end
end
```

def render

action_controller/base.rb

```
def render(options = nil, extra_options = {}, &block) #:doc:
  raise DoubleRenderError, "Can only render or redirect once per action"
if performed?

  validate_render_arguments(options, extra_options, block_given?)

  if options.nil?
    options = { :template => default_template, :layout => true }
  elsif options == :update
    options = extra_options.merge({ :update => true })
  elsif options.is_a?(String) || options.is_a?(Symbol)
    case options.to_s.index('/')
    when 0
      extra_options[:file] = options
    when nil
      extra_options[:action] = options
    else
      extra_options[:template] = options
    end

    options = extra_options
  elsif !options.is_a?(Hash)
    extra_options[:partial] = options
    options = extra_options
  end

  layout = pick_layout(options)
  response.layout = layout.path_without_format_and_extension if layout
  logger.info("Rendering template within #
{layout.path_without_format_and_extension}") if logger && layout

  if content_type = options[:content_type]
    response.content_type = content_type.to_s
  end

  if location = options[:location]
    response.headers["Location"] = url_for(location)
  end

  if options.has_key?(:text)
    text = layout ? @template.render(options.merge(:text => options
[:text], :layout => layout)) : options[:text]
    render_for_text(text, options[:status])

  else
    if file = options[:file]
      render_for_file(file, options[:status], layout, options[:locals] ||
{})

    elsif template = options[:template]
      render_for_file(template, options[:status], layout, options[:locals]
|| {})

    elsif inline = options[:inline]
      render_for_text(@template.render(options.merge(:layout => layout)),
options[:status])

    elsif action_name = options[:action]
      render_for_file(default_template(action_name.to_s), options
[:status], layout)

    elsif xml = options[:xml]
      response.content_type ||= Mime::XML
      render_for_text(xml.respond_to?(:to_xml) ? xml.to_xml : xml, options
[:status])

    elsif js = options[:js]
      response.content_type ||= Mime::JS
      render_for_text(js, options[:status])

    elsif options.include?(:json)
      json = options[:json]
      json = ActiveSupport::JSON.encode(json) unless json.is_a?(String)
      json = "#{options[:callback]}(#{json})" unless options
[:callback].blank?
      response.content_type ||= Mime::JSON
      render_for_text(json, options[:status])

    elsif options[:partial]
      options[:partial] = default_template_name if options[:partial] ==
true
      if layout
        render_for_text(@template.render(:text => @template.render
(options), :layout => layout), options[:status])
      else
        render_for_text(@template.render(options), options[:status])
      end

    elsif options[:update]
      @template.send(:_evaluate_assigns_and_ivars)

      generator =
ActionView::Helpers::PrototypeHelper::JavaScriptGenerator.new(@template,
&block)
      response.content_type = Mime::JS
      render_for_text(generator.to_s, options[:status])

    elsif options[:nothing]
      render_for_text(nil, options[:status])

    else
      render_for_file(default_template, options[:status], layout)
    end
  end
end
```

action_controller/base.rb

```
def render(options = nil, extra_options = {}, &block) #:doc:
  raise DoubleRenderError, "Can only render or redirect once per action"
if performed?

  validate_render_arguments(options, extra_options, block_given?)

  if options.nil?
    options = { :template => default_template, :layout => true }
  elsif options == :update
    options = extra_options.merge({ :update => true })
  elsif options.is_a?(String) || options.is_a?(Symbol)
    case options.to_s.index('/')
    when 0
      extra_options[:file] = options
    when nil
      extra_options[:action] = options
    else
      extra_options[:template] = options
    end

    options = extra_options
  elsif options.is_a?(Hash)
    extra_options = options.dup
    options = extra_options
  end

  layout = pick_layout(options)
  response.layout = layout.path_without_format_and_extension if layout
  logger.info("Rendering template within #
{layout.path_without_format_and_extension}") if logger && layout

  if content_type = options[:content_type]
    response.content_type = content_type.to_s
  end

  if location = options[:location]
    response.headers["Location"] = url_for(location)
  end

  if options.has_key?(:text)
    text = layout ? @template.render(options.merge(:text => options
[:text], :layout => layout)) : options[:text]
    render_for_text(text, options[:status])

  else
    if file = options[:file]
      render_for_file(file, options[:status], layout, options[:locals] ||
{})

    elsif template = options[:template]
      render_for_file(template, options[:status], layout, options[:locals]
|| {})

    elsif inline = options[:inline]
      render_for_text(@template.render(options.merge(:layout => layout)),
options[:status])

    elsif action_name = options[:action]
      render_for_file(default_template(action_name.to_s), options
[:status], layout)

    elsif xml = options[:xml]
      response.content_type ||= Mime::XML
      render_for_text(xml.respond_to?(:to_xml) ? xml.to_xml : xml, options
[:status])

    elsif js = options[:js]
      response.content_type ||= Mime::JS
      render_for_text(js, options[:status])

    elsif options.include?(:json)
      json = options[:json]
      json = ActiveSupport::JSON.encode(json) unless json.is_a?(String)
      json = "#{options[:callback]}(#{json})" unless options[:callback].blank?
      response.content_type ||= Mime::JSON
      render_for_text(json, options[:status])

    elsif options[:partial]
      options[:partial] = default_template_name if options[:partial] ==
true
      if layout
        render_for_text(@template.render(:text => @template.render
(options), :layout => layout), options[:status])
      else
        render_for_text(@template.render(options), options[:status])
      end

    elsif options[:update]
      @template.send(:_evaluate_assigns_and_ivars)

      generator =
ActionView::Helpers::PrototypeHelper::JavaScriptGenerator.new(@template,
&block)
      response.content_type = Mime::JS
      render_for_text(generator.to_s, options[:status])

    elsif options[:nothing]
      render_for_text(nil, options[:status])

    else
      render_for_file(default_template, options[:status], layout)
    end
  end
end
```

layout = pick_layout(options)

action_controller/layouts.rb

```
def pick_layout(options)
  if options.has_key?(:layout)
    case layout = options.delete(:layout)
    when FalseClass
      nil
    when NilClass, TrueClass
      active_layout if action_has_layout? &&
        candidate_for_layout?(:template => default_template_name)
    else
      active_layout(layout, :html_fallback => true)
    end
  else
    active_layout if action_has_layout? &&
      candidate_for_layout?(options)
  end
end
```

Base

Layouts

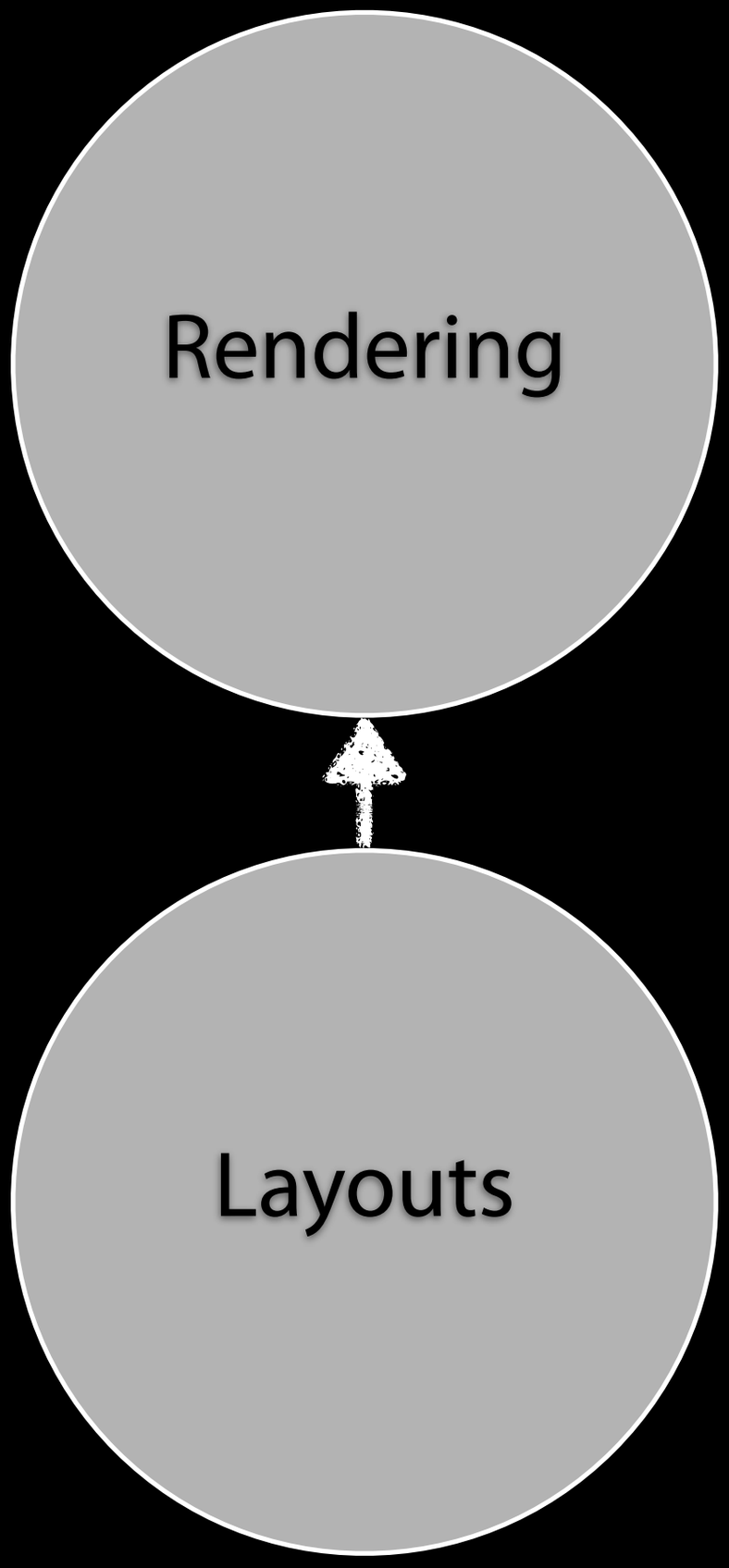
Base

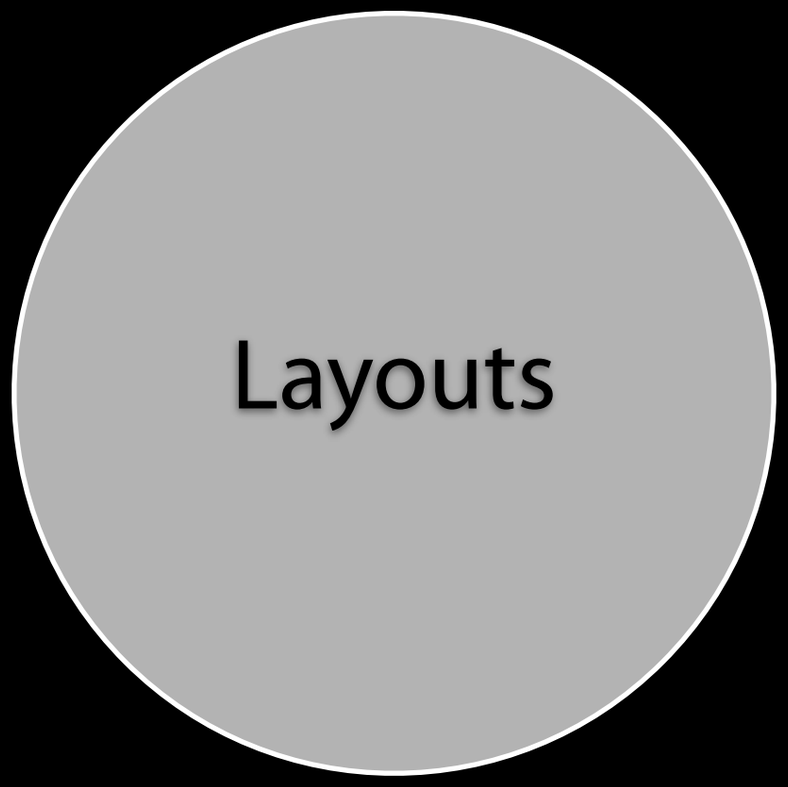
Layouts

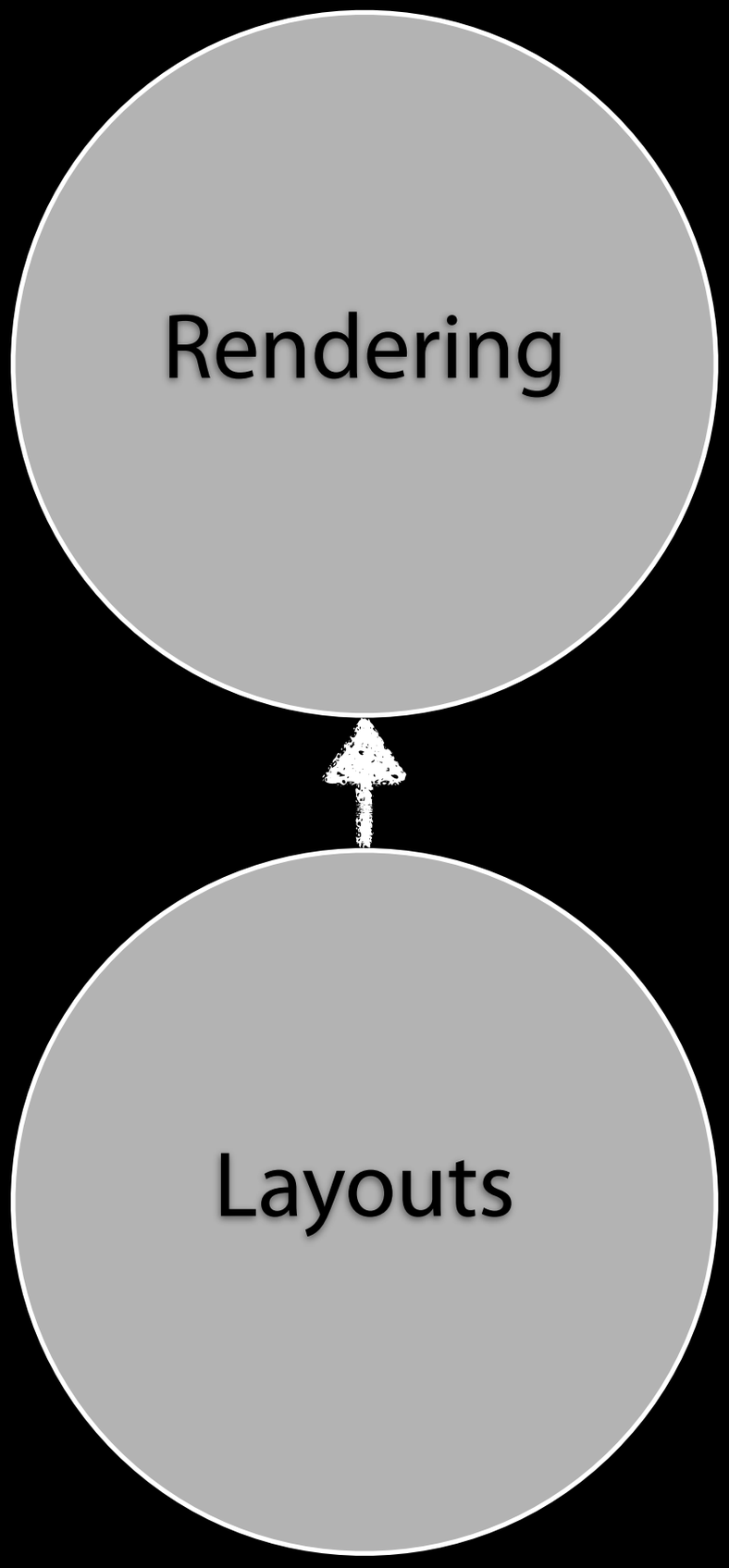
Base

Rendering

Layouts


Rendering
Layouts


Layouts


Rendering
Layouts

action_controller/base.rb

```
def render(action = nil, options = {}, &blk)
  options = _normalize_options(action, options, &blk)
  super(options)
end
```

action_controller/base.rb

```
def _normalize_options(action = nil, options = {}, &blk)
  if action.is_a?(Hash)
    options, action = action, nil
  elsif action.is_a?(String) || action.is_a?(Symbol)
    key = case action = action.to_s
    when %r{^/} then :file
    when %r{/}  then :template
    else             :action
    end
    options.merge! key => action
  elsif action
    options.merge! :partial => action
  end

  if options.key?(:action) && options[:action].to_s.index("/")
    options[:template] = options.delete(:action)
  end

  if options[:status]
    options[:status] = interpret_status(options[:status]).to_i
  end

  options[:update] = blk if block_given?
  options
end
```

metal/rendering_controller.rb

```
def render(options)
  super
  self.content_type ||=
    option[:_template].mime_type.to_s
  response_body
end
```

metal/layouts.rb

```
module ActionController
  module Layouts
    extend ActiveSupport::Concern

    include ActionController::RenderingController
    include AbstractController::Layouts
    ...
  end
end
```

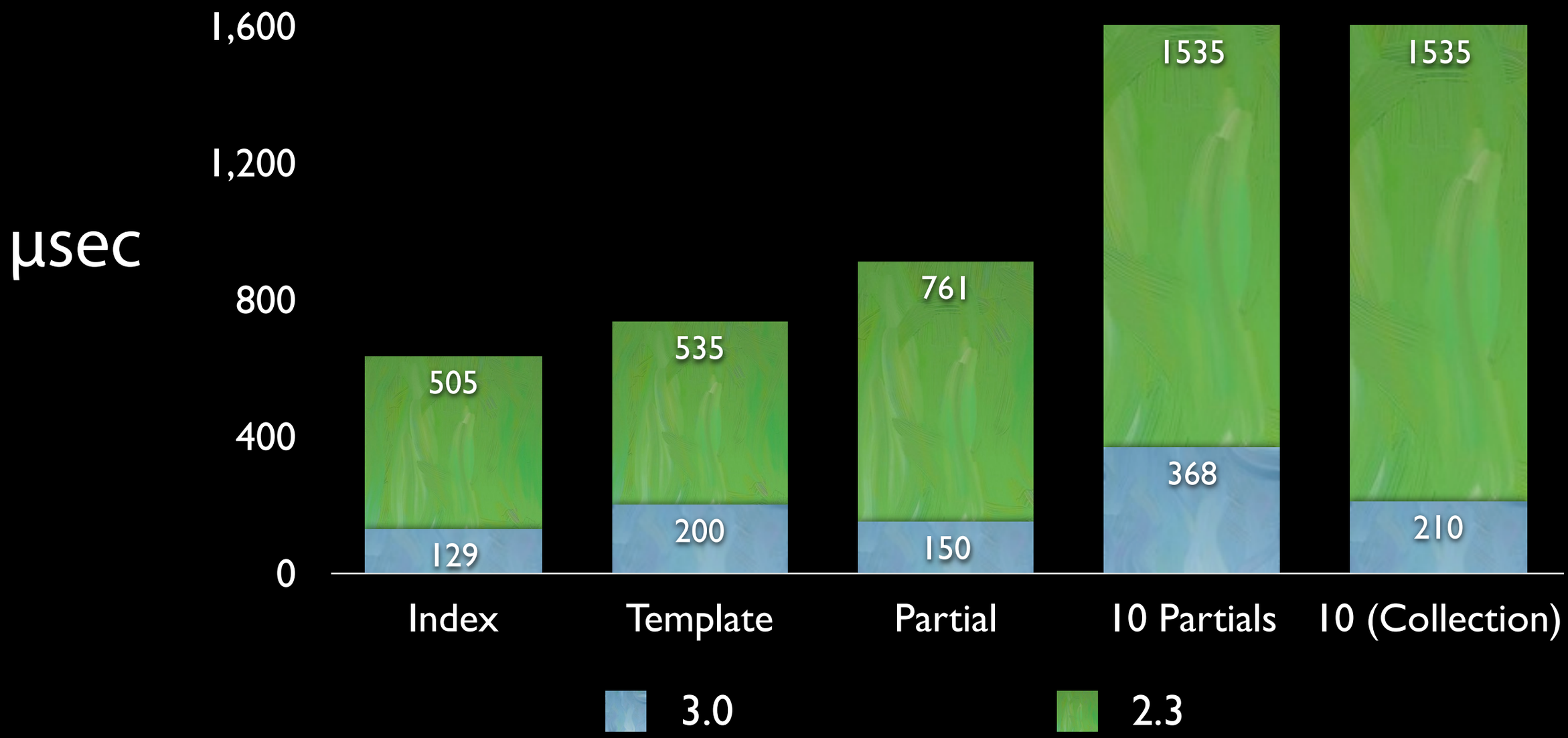
metal/layouts.rb

```
module ActionController
  module Layouts
    extend ActiveSupport::Concern

    include ActionController::RenderingController
    include AbstractController::Layouts
    ...
  end
end
```

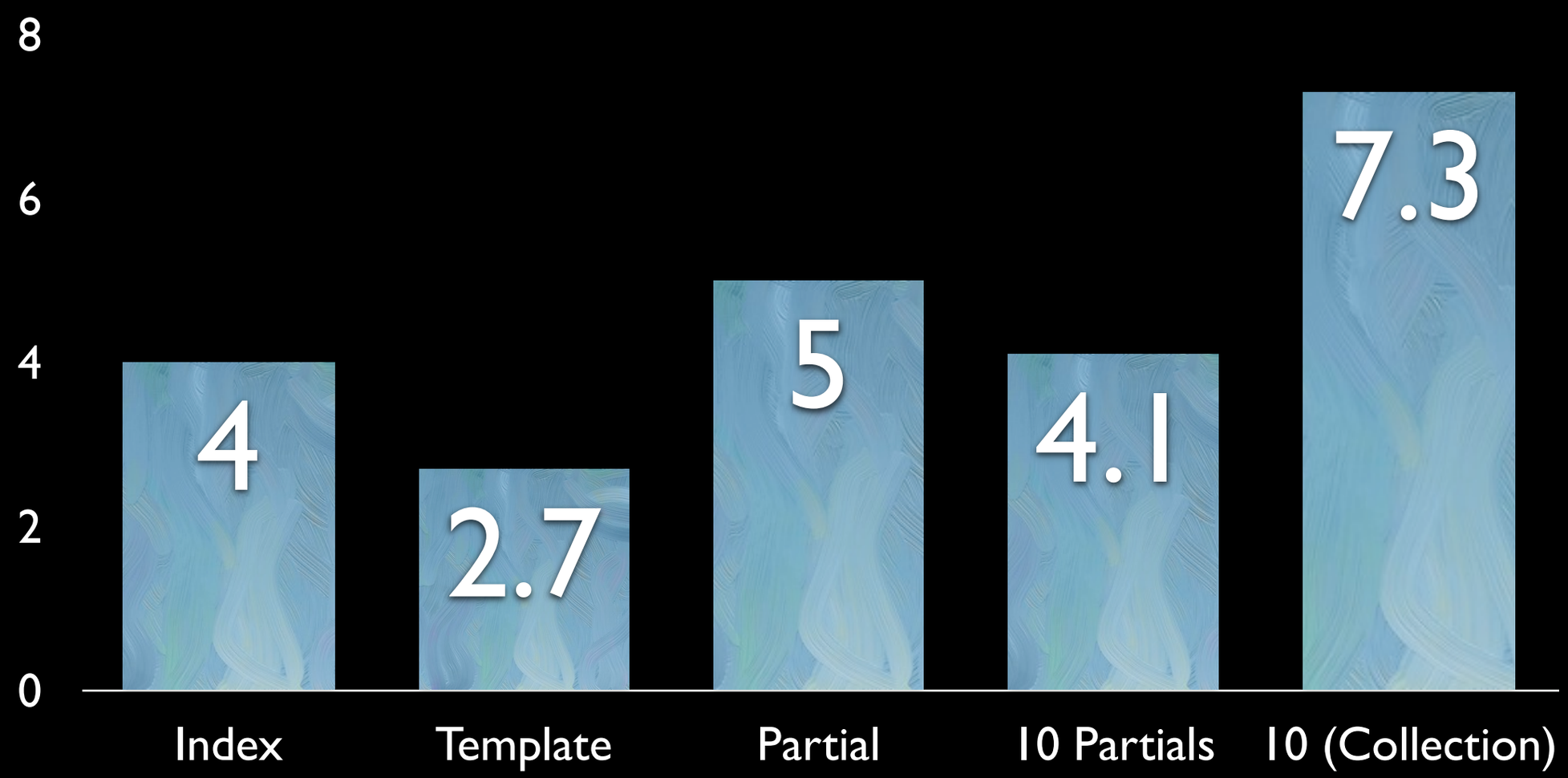
Rails will become more modular, starting with a rails-core, and including the ability to opt in or out of specific components.

We will port all of our performance improvements into Rails

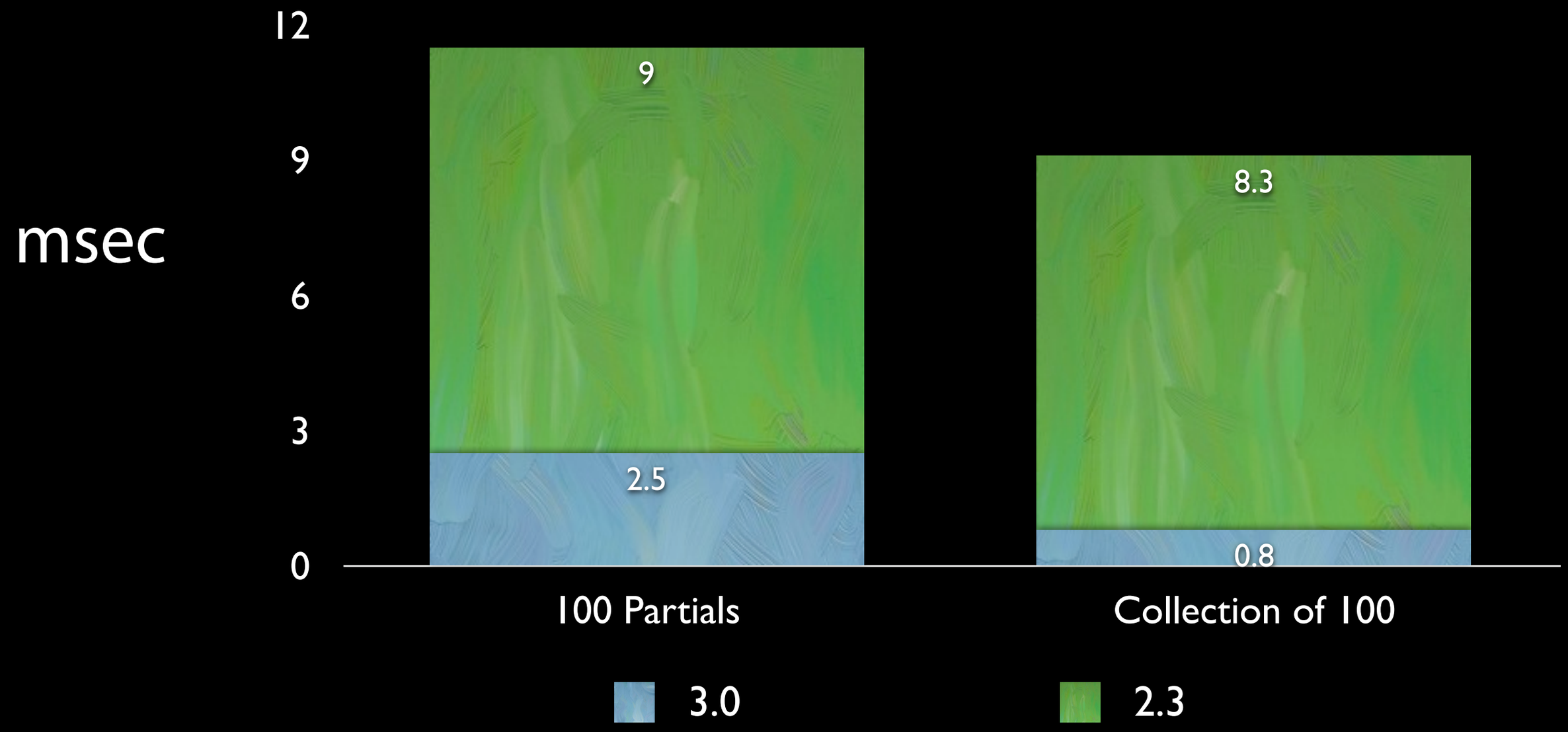
Rails 2.3 vs. 3.0



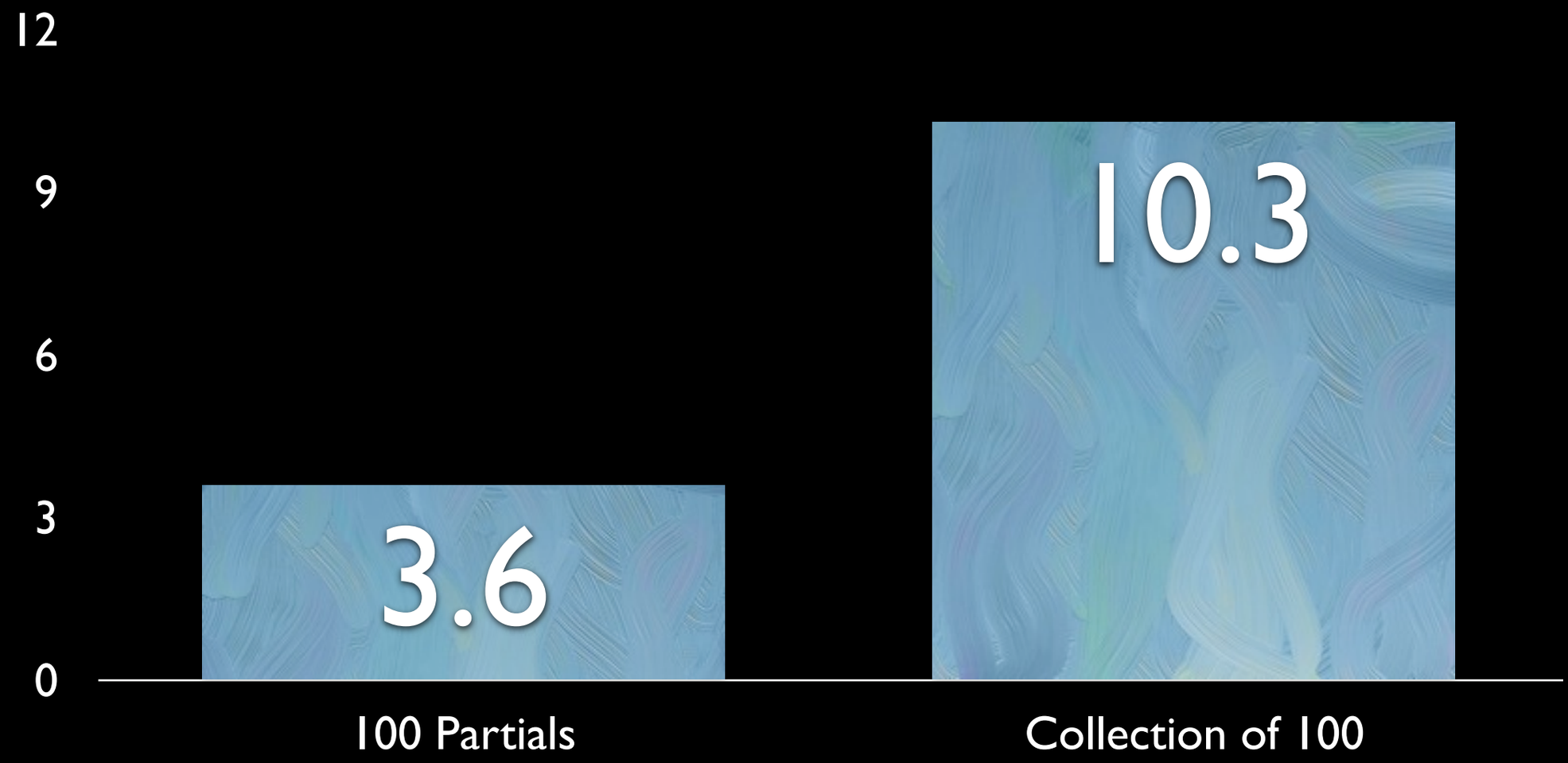
Times Faster



Rails 2.3 vs. 3.0



Times Faster



merb-core/controller/mixins/render.rb

```
sent_template = with.map do |temp|
  locals[as] = temp unless named_local

  if template_method && self.respond_to?(template_method)
    locals[:collection_index] += 1
    send(template_method, locals)
  else
    raise TemplateNotFound,
      "Could not find template at #{template_location}.*"
  end
end.join
```

action_view/partials.rb

```
index = 0
options[:collection].map do |object|
  _partial_path ||= partial ||
    ActionController::RecordIdentifier.partial_path(
      object, controller.class.controller_path)
  template = _pick_partial_template(_partial_path)
  local_assigns[template.counter_name] = index
  result = template.render_partial(self, object,
    local_assigns.dup, as)
  index += 1
  result
end.join(spacer).html_safe!
```

action_view/partials.rb

```
index = 0
options[:collection].map do |object|
  _partial_path ||= partial ||
    ActionController::RecordIdentifier.partial_path(
      object, controller.class.controller_path)
  template = _pick_partial_template(_partial_path)
  local_assigns[template.counter_name] = index
  result = template.render_partial(self, object,
    local_assigns.dup, as)
  index += 1
  result
end.join(spacer).html_safe!
```

action_view/render/partials.rb

```
@collection.each do |object|
  locals[counter_name] += 1
  locals[as] = object

  segments << template.render(@view, locals)
end
```

Rails will be modified to more easily support DataMapper or Sequel as first-class ORMs.

action_view/context.rb

```
def convert_to_model(object)
  object.respond_to?(:to_model) ?
    object.to_model :
    object
end
```

action_view/helpers/form_helper.rb

```
def apply_form_for_options!(object_or_array, options)
  object = object_or_array.is_a?(Array) ?
           object_or_array.last : object_or_array

  object = convert_to_model(object)

  html_options =
    if object.respond_to?(:new_record?) && object.new_record?
      { :class  => dom_class(object, :new),
        :id => dom_id(object), :method => :post }
    else
      { :class  => dom_class(object, :edit),
        :id => dom_id(object, :edit), :method => :put }
    end

  options[:html] ||= {}
  options[:html].reverse_merge!(html_options)
  options[:url] ||= polymorphic_path(object_or_array)
end
```

action_view/helpers/form_helper.rb

```
def apply_form_for_options!(object_or_array, options)
  object = object_or_array.is_a?(Array) ?
           object_or_array.last : object_or_array

  object = convert_to_model(object)

  html_options =
    if object.respond_to?(:new_record?) && object.new_record?
      { :class  => dom_class(object, :new),
        :id => dom_id(object), :method => :post }
    else
      { :class  => dom_class(object, :edit),
        :id => dom_id(object, :edit), :method => :put }
    end

  options[:html] ||= {}
  options[:html].reverse_merge!(html_options)
  options[:url] ||= polymorphic_path(object_or_array)
end
```

object = convert_to_model(object)

```ruby
module DataMapper::Resource
  def to_model
    self
  end
end
```

```ruby
class Sequel::Model
  def to_model
    return Proxy.new(self)
  end
end
```

```
config.generators do |g|
  g.orm             :datamapper
  g.template_engine :haml
  g.test_framework  :rspec
end
```

In general, we will take a look at features in Merb that are not in Rails (the most obvious example is the more robust router) and find a way to bring them into Rails

Bundler

```
# environment.rb
```

```
config.gem "activemerchant"
```

```
# Gemfile

gem "activemerchant"
```

```
clear_sources
bundle_path "vendor/bundler_gems"

source "http://gemcutter.org"
source "http://gems.github.com"

gem "rails", "2.3.4"
gem "rack",  "1.0.1"

gem "clearance",          "0.8.2"
gem "will_paginate",      "2.3.11"
gem "sinatra",            "0.9.4"
gem "xml-simple",         "1.0.12"
gem "gchartrb",           "0.8",
    :require_as => "google_chart"
gem "ddollar-pacecar", "1.1.6",
    :require_as => "pacecar"
gem "net-scp",            "1.0.2"
```

```
only :test do
  gem "shoulda",      "2.10.2"
  gem "factory_girl", "1.2.3"
  gem "webrat",       "0.5.3"
  gem "cucumber",     "0.3.101"
  gem "rr",           "0.10.4"
  gem "redgreen",     "1.2.2"
  gem "fakeweb",      "1.2.6"
  gem "rack-test",    "0.5.0",
      :require_as => "rack/test"
end
```

```
only [:staging, :production] do
  gem "rack-cache",          "0.5.2",
      :require_as => "rack/cache"

  gem "aws-s3",              "0.6.2",
      :require_as => "aws/s3"

  gem "ambethia-smtp-tls", "1.1.2",
      :require_as => "smtp-tls"

  gem "memcache-client",   "1.7.5",
      :require_as => "memcache"
end
```

Specify Dependencies

Specify Dependencies

```
gem bundle
```

Specify Dependencies

```
gem bundle
```

```
git push
```

Specify Dependencies

`gem bundle`

`git push`

`git clone`

Specify Dependencies

`gem bundle`

`git push`

`git clone`

`gem bundle`

Specify Dependencies

`gem bundle`

`git push`

`git clone`

`gem bundle`

It works!

Specify Dependencies

`script/bundle`

`git push`

`git clone`

`script/bundle`

It works!

Works with compiled gems

Idempotent

Live Today

Router

Backward Compatible

```
controller :sessions do
  get  'login', :to => :new,
    :as => :login

  post 'login', :to => :create

  delete 'logout', :to => :destroy,
    :as => :logout
end
```

```
match 'openid/login',
   :via => [:get, :post],
   :to => "openid#login"
```

```
match '/export/:id/:file',
    :to => :export,
    :as => :export_download,
    :constraints => { :file => /.*/ }
```

```ruby
get 'admin',
  :to => "queenbee#index",
  :constraints =>
    {:ip => /192\.168\.1\.\d\d\d/}
```

```ruby
constraints(:ip => /192\.168\.1\.\d\d\d/) do
  get 'admin', :to => "queenbee#index"
end
```

```ruby
class IpRestrictor
  def self.matches?(request)
    request.ip =~ /192\.168\.1\.1\d\d/
  end
end
```

```ruby
constraints IpRestrictor do
  get 'admin/accounts',
    :to => "queenbee#accounts"
end
```

```
match 'sprockets.js',
  :to => SprocketsApp
```

```
match 'account/login',
    :to => redirect("/login")
```

```ruby
def redirect(path, options = {})
  status = options[:status] || 301
  lambda { |env|
    req = Rack::Request.new(env)
    url = req.scheme + '://' + req.host + path
    [status,
     {'Location' => url, 'Content-Type' => 'text/html'},
     ['Moved Permanently']]
  }
end
```

```
:to => "queenbee#index"
```

```
:to =>
  QueenbeeController.action(:index)
```

```ruby
resources :posts do
  get  :archive, :toggle_view,
       :on => :collection
  post :preview, :on => :member

  resource :subscription

  resources :comments do
    post :preview, :on => :collection
  end
end
```

One more thing...

```
match ":controller(/:action(/:id))(.:format)"
```

```
match ":controller(/:action(/:id))(.:format)"
```

AbstractController

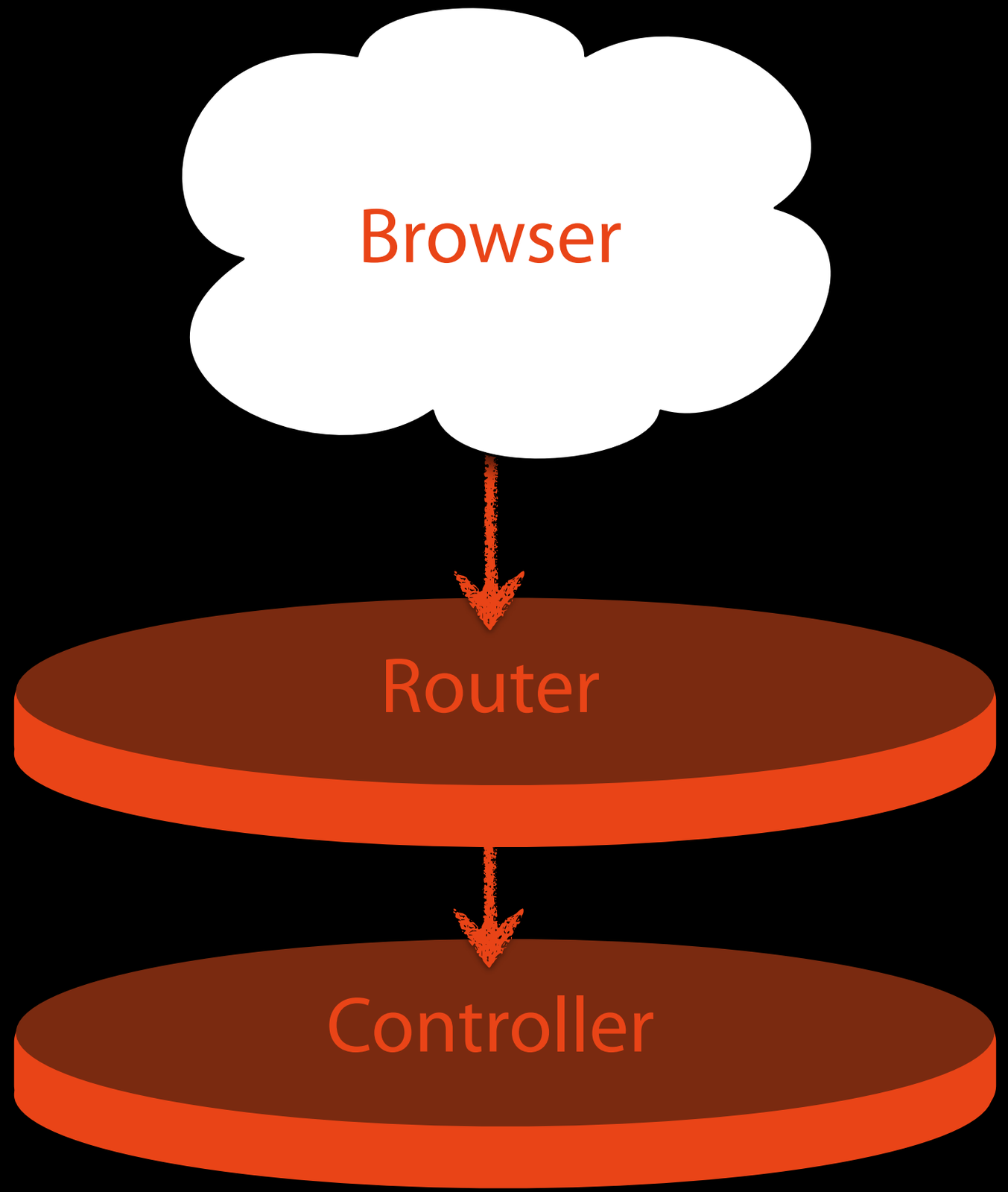
display

```
class Posts < Merb::Controller
  provides :html, :json, :xml

  def index
    @posts = Post.all
    display @posts
  end
end
```

respond_with

```ruby
class PostsController < ApplicationController
  respond_to :html, :json, :xml

  def index
    @posts = Post.find(:all)
    respond_with @posts
  end
end
```

Rails will continue their recent embrace of Rack, which is a really exciting development in the Ruby community which we believe will improve the state of modular, sharable logic between applications


Browser
Router
Controller

action_controller/routing/route_set.rb

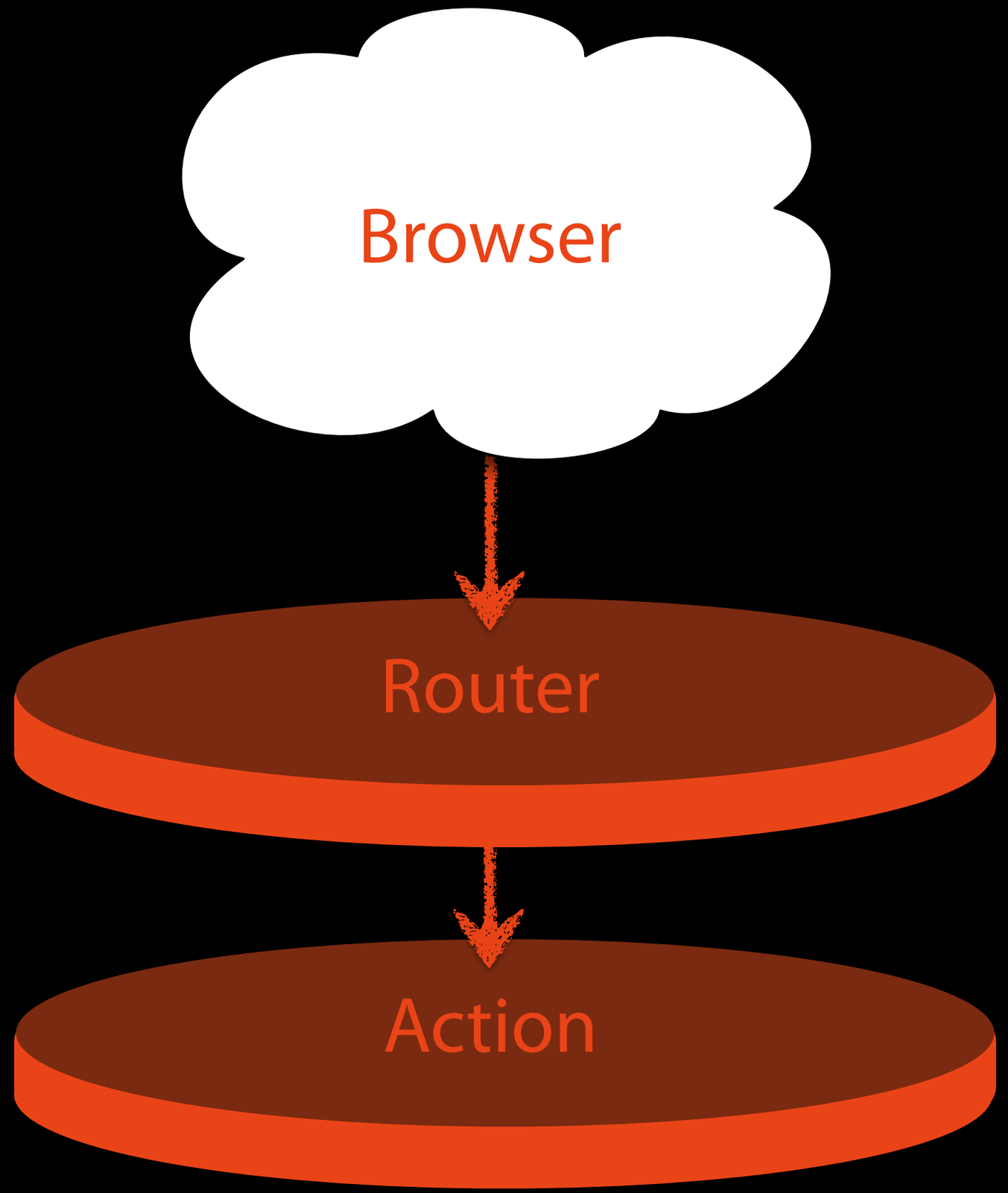
```
def call(env)
  request = Request.new(env)
  app = Routing::Routes.recognize(request)
  app.call(env).to_a
end

def recognize(request)
  params = recognize_path(request.path,
    extract_request_environment(request))
  request.path_parameters =
    params.with_indifferent_access

  "#{params[:controller].to_s.camelize"\
  "Controller".constantize
end
```

action_controller/base.rb

```
def assign_names
  @action_name = (params['action'] || 'index')
end
```

action_controller/base.rb

```
def perform_action
  if action_methods.include?(action_name)
    send(action_name)
    default_render unless performed?
  elsif respond_to? :method_missing
    method_missing action_name
    default_render unless performed?
  else
    begin
      default_render
    rescue ActionView::MissingTemplate => e
      # Was the implicit template missing, or was it another template?
      if e.path == default_template_name
        raise UnknownAction, "No action responded to #{action_name}."\
          "Actions: #{action_methods.sort.to_sentence(:locale=>:en)}",
          caller
      else
        raise e
      end
    end
  end
end
```


Browser
Router
Action

action_dispatch/routing/route_set.rb

```ruby
def call(env)
  params = env[PARAMETERS_KEY]
  merge_default_action!(params)
  split_glob_param!(params) if @glob_param
  params.each do |key, value|
    if value.is_a?(String)
      params[key] = URI.unescape(value)
    end
  end

  if env['action_controller.recognize']
    [200, {}, params]
  else
    controller = controller(params)
    controller.action(params[:action]).call(env)
  end
end
```

action_dispatch/routing/route_set.rb

```
def call(env)
  params = env[PARAMETERS_KEY]
  merge_default_action!(params)
  split_glob_param!(params) if @glob_param
  params.each do |key, value|
    if value.is_a?(String)
      params[key] = URI.unescape(value)
    end
  end

  if env['action_controller.recognize']
    [200, {}, params]
  else
    controller = controller(params)
    controller.action(params[:action]).call(env)
  end
end
```

```
controller.
  action(params[:action]).
  call(env)
```

```
PostsController.action(:index)
```

```
class PostsController < ApplicationController
  use MyMiddleware
end
```

```ruby
class MyMiddleware < ActionController::Middleware
  include ActionController::RenderingController
  include ActionController::Helpers

  helper do
    def in_views
      "WOW!"
    end
  end

  def call(env)
    @make_available = "YES!"
    render "some_template"
  end
end
```

```
class Mephisto < Rails::Application
  config.frameworks = [:active_record]
end
```

```
# config.ru
run Mephisto
```

- Modular
- Performant
- ORM Agnostic
- Improved Rack Use
- Imported Merb Features

- ☑ Modular
- ☐ Performant
- ☐ ORM Agnostic
- ☐ Improved Rack Use
- ☐ Imported Merb Features

- ☑ Modular
- ☑ Performant
- ☐ ORM Agnostic
- ☐ Improved Rack Use
- ☐ Imported Merb Features

- ☑ Modular
- ☑ Performant
- ☑ ORM Agnostic
- ☐ Improved Rack Use
- ☐ Imported Merb Features

- ☑ Modular
- ☑ Performant
- ☑ ORM Agnostic
- ☑ Improved Rack Use
- ☐ Imported Merb Features

- ☑ Modular
- ☑ Performant
- ☑ ORM Agnostic
- ☑ Improved Rack Use
- ☑ Imported Merb Features

Now Let's Ship It


Questions?