Controllers

OWNING RAILS MASTERCLASS

Routing

```ruby
match "/posts(/:year(/:month))" => "home#index",
      year: /\d{4}/, month: /\d{2}/,
      defaults: { year: 2011 },
      as: :posts

match "/about" => "home#about", via: [:get, :post]
get "/about" => "home#about"

match 'photos/*other' => 'photos#unknown'

get "/about_us" => redirect("/about")
get "/users/:name" => redirect("/%{name}")
get "/users/:name/dashboard" => redirect {|params, request|}

scope "admin", as: "admin" do
  get "dashboard" => "admin#show"
end
namespace "admin" do
  # ...
end

resources :photos do
  member do
    get 'preview'
  end
  collection do
    get 'search'
  end
end
resource :account, only: [:show, :update]

root to: "home#index"
```

Constraints

Creates the posts_path(year, month) helper

/photos/aye/12 => params[:other] = "aye/12"

prefix routes with /admin

prefix helpers with admin_ and controllers with Admin::

Same as

/photos/1/preview

/photos/search

Verb	Path	Action
GET	/photos	index
GET	/photos/new	new
POST	/photos	create
GET	/photos/:i	show
GET	/photos/:i/edit	edit
PUT	/photos/:i	update
DELETE	/photos/:i	destroy

Rendering

```ruby
render :index, layout: "admin"
render "controller/action"
render "/some/file", status: :not_found
render text: "hello", content_type: "text/plain"
render nothing: true
render inline: "<%= erb %>"
render json: @photo
render xml: @photo, location: photo_url(@photo)
render js: "alert('Hello Rails');"
head :ok

respond_with @photo, responder: FlashResponder,
                     message: "An option!" do |format|
  format.html { redirect_to @photo }
end

class FlashResponder < ActionController::Responder
  def to_html
    message = options[:message] || "Updated, nice!"
    controller.flash.notice = message if post?
    super
  end
end
```

:ok, :bad_request, :created, :not_found or 404

Option passed to the responder

In views

```ruby
render "partial",
       layout: "special"
render @photo
render @photos
```

Strong Parameters

```ruby
params.require(:post).
       permit(:title, :body)
```

Views

OWNING RAILS MASTERCLASS

Forms

```erb
<%= form_for @photo, html:   { multipart: true },
                     url:    photo_url,
                     remote: true  do |f| %>
  <%= f.label :title %>: <%= f.text_field :title %>
  <%= f.text_area :description, size: "60x12" %>
  <%= f.check_box :private %>
  <%= f.radio_button :rating, 1 %>
  <%= f.password_field :password %>
  <%= f.number_field :number %>
  <%= f.range_field :range %>
  <%= f.search_field :range %>
  <%= f.telephone_field :range %>
  <%= f.email_field :range %>
  <%= f.url_field :range %>
  <%= f.hidden_field :parent_id %>
  <%= f.file_field :data %>
  <%= f.select :city_id, [['Montreal', 1], ['New York', 2]] %>
  <%= f.collection_select :author_id, Author.all, :id, :name %>
  <%= f.date_select :publish_on %>
  <%= f.datetime_select :publish_at %>

  <%= f.fields_for :address do |af| %>
    <%= af.text_field :street %>
    <%= af.text_field :zip_code %>
  <% end %>

  <%= f.submit "Create", disable_with: "Loading..." %>
<% end %>
```

Optional with @photo

Send with AJAX

Block

```erb
<%= my_block_helper do %>
  <h1>Title</h1>
<% end %>

def my_block_helper(&block)
  capture(&block)
end
```

Links

```erb
<%= link_to "Show", @photo, remote:  true,
                            method:  :delete,
                            confirm: "Sure?" %>
<%= link_to @photo do %>
  <h1>Title</h1>
<% end %>
```

URLs

```erb
<%= url_for(:back) %>
<%= url_for(@photo) %>
<%= url_for(Photo.new) %>
```

/photos/1

/photos

Rendering

```erb
<%= render @photo %>
<%= render @photos %>
```

renders photos/_photo.html.erb

renders photos/_photo.html.erb for each item

Escaping

```erb
<%= raw "<p>raw html</p>" %>
```

Everything is escaped by default

Models

OWNING RAILS MASTERCLASS

Query

```
Ticket.where(status: "opened").
       where("comments_count > ?", comments_count).
       where(created_at: 1.year.ago..Time.now).
       where(owner_id: [1,3,5]).
       includes(:comments).
       order("created_at desc").
       limit(5).
       first

User.find_by_name("Bob")
Ticket.where(status: "opened").new(title: "...")
Ticket.where(...).create(title: "...")
Ticket.where(...).first_or_create
Ticket.where(...).first_or_initialize
User.find_or_create_by(name: "Don")
User.find_or_initialize_by(name: "Don")
```

where, select, group, order, limit, offset, joins, includes, lock, readonly, from, having

to_a, load, first, last, count, maximum(attr), minimum(attr), average(attr) or sum(attr)

Scopes

```
class Ticket < ActiveRecord::Base
  default_scope { order(:created_at) }
  scope :opened, -> status do
        where(status: status)
  end
end
```

Associations

```
class Project < ActiveRecord::Base
  belongs_to              :portfolio
  has_one                 :project_manager, class_name:    "User"
  has_many                :milestones,      dependent:     :destroy
                                            before_add:    :method,
                                            after_add:     -> { |project, milestone| },
                                            before_remove: [:multiple, :method_or_proc],
                                            after_remove:  method_or_proc
  has_many                :tasks,           through:       :milestones
  has_many                :assets,          as:            :attachable
  has_and_belongs_to_many :categories
end
class Asset < ActiveRecord::Base
  belongs_to              :attachable,      polymorphic: true
end
```

Options

:as specifies the polymorphic interface to use

Validation

```
class User < ActiveRecord::Base
  validates :name,     presence:     true,
                       uniqueness:   true,
                       format:       /\A[a-z]\w+\Z/i,
                       length:       1..10,
                       exclusion:    %w( admin god )
  validates :status,   inclusion:    %w( new registered banned ),
  validates :password, confirmation: true,

  validates :terms,    acceptance:   true,

  validates :age,      numericality: true,  only_integer: true,
                                            odd:          true
  validates :title,    title:        true
end
```

Common options

```
allow_blank: true
if:          :password?,
unless:      -> user { ... }
on:          create,
message:     "Gotta accept"
```

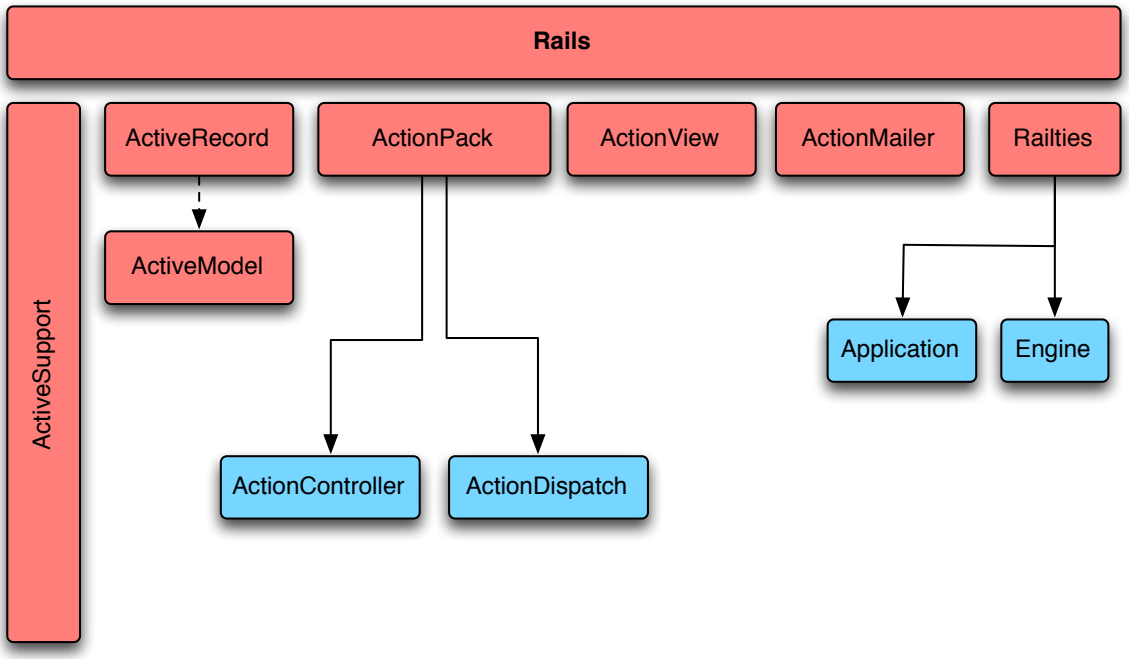
Custom validator

```
class TitleValidator < ActiveModel::EachValidator
  def validate_each(record, attribute, value)
    record.errors.add(attribute, 'must ...') if ...
  end
end
```

Callbacks

before, after, around — validation, save, create, update, destroy, initialize, find, rollback, commit

Gems



Where's the code?

Utility classes and core extensions	`ActiveSupport`	`bundle open activesupport`
Model logic that requires DB access	`ActiveRecord`	`bundle open activerecord`
Model logic that **doesn't** require DB access	`ActiveModel`	`bundle open activemodel`
Code handling the processing & routing of a requests	`ActionDispatch`	`bundle open actionpack`
Features you get access to inside your controllers	`ActionController`	`bundle open actionpack`
View and helpers code	`ActionView`	`bundle open actionview`
Framework code, holding all the components together. CLI, code generators, rake tasks.	`Rails`	`bundle open railties`