

# ROUTING API

## BASIC ROUTING

```
YourApp::Application.routes.draw do
  resources :posts
  match '/all' => 'posts#index'
  root :to => "home#index"
end
```

## OPTIONAL PARAMETERS

```
match '/posts(/:yy(/:mm))' => "posts#index"

class PostsController < ApplicationController
  def index
    # params[:yy]
    # params[:mm]
  end
end
```

## REDIRECTION

```
match '/sign_out' => redirect("/signout")
match '/users/:name' => redirect { |params| "/#{params[:name]}"}
match '/google' => redirect('http://google.com')
```

Gives you sign\_in\_path helper

## NAMED ROUTES

```
match '/sign_in' => 'session#new', :as => 'sign_in'
match '/reset_password/:token' => 'users#reset_password', :as => 'reset_password'
```

Gives reset\_password\_path('key')

## RACK ROUTING

```
get '/hello' => proc { |env| [200, {}, "Hello Rack"]}
get '/rack_endpoint' => PostsController.action(:index)
get '/rack_app' => CustomRackApp
```

## CONSTRAINTS

```
match '/:year' => "posts#index",
      :constraints => { :year => /\d{4}/, :ip => /192\.168\.\d{1,3}\./ }

constraints(:host => /localhost/) do
  resources :posts
end

constraints IpRestrictor do
  get 'admin/accounts' => "queenbee#accounts"
end
```

Responds to self.matches?(request)

Commented out by default

## LEGACY ROUTE

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

## SCOPE

```
scope ':token', :token => /\w{5}/ do
  resources :rooms do
    resources :meetings
  end
end

scope '(:locale)', :locale => /en|pl/ do
  resources :posts
  root :to => 'posts#index'
end
```

Requires token parameter to get to resources and must be 5 alphanumeric characters

Locale is optional for these routes

## BUNDLER COMMANDS

< > required  
[ ] optional

### \$ bundle

Make sure all dependencies in your Gemfile are available to your application. If they are not available, go install them.

### \$ bundle --without <group\_name>

Installs everything, except gems included in <group\_name>.

### \$ bundle --deployment

Isolates all gems into vendor/bundle, requires up-to-date Gemfile.lock, use gems in vendor/cache if they exist.

### \$ bundle check

Checks if the dependencies listed in Gemfile are satisfied by currently installed gems.

### \$ bundle show [gem\_name]

Shows all libraries which are included by the Gemfile and their dependencies. If [gem\_name] is given, shows where it is located in the filesystem.

### \$ bundle open <gem\_name>

Opens the gem source in the default editor.

### \$ bundle update [gem\_name]

Recreates Gemfile.lock and runs "bundle" to install new dependencies.

### \$ bundle package

Copies all project gems to vendor/cache/ — use this if you don't want to rely on external servers for deployment.

## GEMFILE SYNTAX

```
source "http://rubygems.org"

gem "hpricot", "0.6"
gem "sqlite3-ruby", :require => "sqlite3"
gem "local_gem", :path => "~/Sites/local_gem"
gem "rails", :git => "git://github.com/rails/rails.git"

git "git://github.com/rails/rails.git" do
  gem "railties"
  gem "active_model" } only these two gems will be fetched from the git repository
end

group :test do
  gem "webrat"
end
```

### Additional parameters

:branch => "branch\_name"  
:tag => "tag\_number"  
:ref => "ref\_number"

### Creating a gemset for your app

\$ rvm use \_ruby\_version\_  
\$ rvm gemset create \_name\_  
\$ rvm gemset use \_name\_  
\$ gem install bundler  
\$ bundle

<http://rvm.beginrescueend.com>



## WORKFLOW

### Developing a new application

1. \$ cd new\_app/
2. \$ bundle init #creates a new Gemfile
3. Add project dependencies
4. Check Gemfile and Gemfile.lock into VCS

### After adding or removing dependencies from Gemfile

1. \$ bundle
2. Commit Gemfile and Gemfile.lock

### After modifying existing dependency versions

1. \$ bundle update
2. Commit Gemfile and Gemfile.lock

### Deploying your application

```
$ bundle package # locally
Copies all app gems to vendor/cache

$ bundle # server
Installs gems from vendor/cache - no external server communication.
```

For more info, visit <http://gembundler.com>

# ACTIVE RELATION

## LAZY LOADING

```
@posts = Post.where(:published => true)
if params[:order]
  @posts = @posts.order(params[:order])
end
@posts.each { |p| ... }

posts = Post.order(params[:order])
@published = posts.where(:published => true)
@unpublished = posts.where(:published => false)
```

## CRUD METHODS

new(attributes)	delete(id_or_array)
create(attributes)	delete_all
create!(attributes)	update(ids, updates)
find(id_or_array)	update_all(updates)
destroy(id_or_array)	exists?
destroy_all	

```
sports_posts = Post.where(:category => 'sports')
new_sports_post = sports_posts.new
new_sports_post.category # => 'sports'

sports_posts.update_all(:published => false)
sports_posts.exists? # => true
```

## CHAIN METHODS

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

## CHAINING

```
@posts = Post.where(:published => true).order(params[:order])

@joe_posts = Post.where(:author => "Joe").include(:comments).limit(10).all
```

## (NAMED) SCOPES

```
class Post < ActiveRecord::Base
  default_scope order('title')
  scope :published, where(:published => true)
  scope :unpublished, where(:published => false)
end
```

Forces query execution and returns an array, not a relation

## DEPRECATED

```
find(id_or_array_of_ids, options)
find(:first, options)
find(:all, options)
first(options)
all(options)
update_all(updates, conditions, options)
```

All options are now sent using the chain methods

# XSS PROTECTION & UJS

## XSS PROTECTION

```
<%= @post.body %> → <%= raw(@post.body) %>
  safe by default           unsafe

<%= link_to raw("<span class='cart'>#{h @user_input}</span>"), cart_path %>
  not escaped      escaped    not escaped
```

## UNOBTRUSIVE JS

```
<%= link_to 'Show', @post, :remote => true %>
  → <a href="/posts/1" data-remote="true">Show</a>

<%= form_for(@post, :remote => true) do |f| %>
  → <form action="/posts" class="new_post" data-remote="true" id="new_post" method="posts">

<%= link_to 'Destroy', @post, :method => :delete %>
  → <a href="/posts/1" data-method="delete" rel="nofollow">Destroy</a>

<%= link_to 'Destroy', @post, :confirm => 'Are you sure?', :method => :delete %>
  → <a href="/posts/1" data-confirm="Are you sure?" data-method="delete" rel="nofollow">Destroy</a>

<%= f.submit :disable_with => "Please wait..." %>
  → <input data-disable-with="Please wait..." id="post_submit" name="commit" type="submit" value="Create Post" />
```

## HTML5 CUSTOM DATA ATTRIBUTES

<b>data-remote</b>	<b>data-confirm</b>
<b>data-method</b>	<b>data-disable-with</b>

Parsed by JavaScript drivers — defaults to rails.js

## DEPRECATED

```
link_to_remote
remote_form_for
observe_field
observe_form
form_remote_tag
button_to_remote
submit_to_remote
link_to_function
periodically_call_remote
```

If you need to use any of the deprecated helpers, visit  
[http://github.com/rails/prototype\\_legacy\\_helper](http://github.com/rails/prototype_legacy_helper)

## USING JQUERY

Go to <https://github.com/rails/jquery-ujs> for instructions

# ACTIONMAILER & ACTIONCONTROLLER

## RAILS MAIL GENERATOR

```
rails generate mailer UserMailer welcome forgot_password
```

**Creates:**

app/mailers/user\_mailer.rb  
app/views/user\_mailer/welcome.text.erb  
app/views/user\_mailer/forgot\_password.text.erb

## BASIC MAILER SYNTAX

```
class UserMailer < ActionMailer::Base
  def welcome(user, subdomain)
    @user = user
    @subdomain = subdomain
    mail(:from => 'admin@app.com',
         :to => @user.email,
         :subject => 'Welcome')
  end
end
```

These instance variables are available within your view

## DELIVERING MESSAGES

```
UserMailer.welcome(user, subdomain).deliver
```

OR

```
message = UserMailer.welcome(user, subdomain)
message.deliver
```

## DEFAULTS AND ATTACHMENTS

```
class UserMailer < ActionMailer::Base
  default :from => 'admin@test.com',
           :reply_to => 'noreply@test.com',
           "X-Time-Code" => Time.now.to_i.to_s

  def welcome(user, subdomain)
    @user = user
    @subdomain = subdomain
    attachments['test.pdf'] = File.read(Rails.root.join('public/test.pdf'))
    attachments['photo.jpg'] = { :content => generate_image() }
    mail(:to => @user.email, :subject => 'Welcome to TestApp') do |format|
      format.html { render 'other_html_welcome' }
      format.text { render 'other_text_welcome' }
    end
  end
end
```

**Defaults To:**  
welcome.html.erb  
welcome.text.erb

## RESPOND\_TO AND RESPOND\_WITH

```
class UsersController < ApplicationController
  respond_to :html, :json, :only => :index
  respond_to :xml, :except => :index

  def index
    respond_with(@users = User.all, :status => :ok)
  end

  def create
    @user = User.create(params[:user])
    respond_with(@user) do |format|
      format.html { redirect_to(users_path) }
    end
  end
end
```

# ACTIVE MODEL

## DIRTY

```
class Person
  include ActiveModel::Dirty
  define_attribute_methods [:name]

  def name
    @name
  end

  def name=(val)
    name_will_change!
    @name = val
  end

  def save
    @previously_changed = changes
    @changed_attributes.clear
  end
end
```

```
>> person = Person.find(id)
=> person.changed?
=> false
=> person.name = 'Bob'
=> person.changed?
=> true
=> person.name_changed?
=> true
=> person.name_was
=> 'Uncle Bob'
=> person.name_change
=> ['Uncle Bob', 'Bob']
=> person.name = 'Bill'
=> person.name_change
=> ['Uncle Bob', 'Bill']
```

## MODULES

AttributeMethods  
 Callbacks  
 Dirty  
 Errors  
 Naming  
 Observing  
 Serialization  
 Validations

## VALIDATIONS

```
class Person
  include ActiveModel::Validations
  attr_accessor :email
  validates_presence_of :email
end

>> p = Person.new
=> #<Person:0x000001021b7198>
=> p.valid?
=> false
=> p.errors
=> {:email=>["can't be blank"]}
```

## Shortcuts

```
validates :terms, :acceptance => true
validates :password, :confirmation => true
validates :username, :exclusion => { :in => %w(admin) }
validates :email, :format => {
  :with => /\A([@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\Z/i,
  :on => :create }
validates :age, :inclusion => { :in => 0..9 }
validates :first_name, :length => { :maximum => 30 }
validates :age, :numericality => true
validates :username, :presence => true
validates :username, :uniqueness => true
```

## SERIALIZATION

```
class Person
  include ActiveModel::Serializers::JSON
  attr_accessor :name
  def attributes
    { :name => name }
  end
end
```

```
>> p = Person.new
=> #<Person:0x00000102186d68>
=> p.name = "Gregg"
=> "Gregg"
=> p.to_json
=> "{\"name\":\"Gregg\"}"
```

## CALLBACKS

```
class Person
  extend ActiveModel::Callbacks
  define_model_callbacks :save
  before_save :action_before_save

  def save
    _run_save_callbacks do
      # Your save action methods here
    end
  end
  private

  def action_before_save
    # Your code here
  end
end
```