Ruby on Rails 2.2 Release Notes

Rails 2.2 delivers several new and improved features. This list covers the major upgrades but doesn't include every little bug fix and change. If you want to see everything, check out the list of commits in the main Rails repository on GitHub.

Along with Rails, 2.2 marks the launch of the Ruby on Rails Guides, the first results of the ongoing Rails Guides hackfest. This site will deliver high-quality documentation of the major features of Rails.

Infrastructure

Rails 2.2 is a significant release for the infrastructure that keeps Rails humming along and connected to the rest of the world.

Internationalization

Rails 2.2 supplies an easy system for internationalization (or i18n, for those of you tired of typing).

- Lead Contributors: Rails i18 Team
- More information :
  - Official Rails i18 website
  - Finally. Ruby on Rails gets internationalized
  - Localizing Rails : Demo application

Compatibility with Ruby 1.9 and JRuby

Along with thread safety, a lot of work has been done to make Rails work well with JRuby and the upcoming Ruby 1.9. With Ruby 1.9 being a moving target, running edge Rails on edge Ruby is still a hit-or-miss proposition, but Rails is ready to make the transition to Ruby 1.9 when the latter is released.

Documentation

The internal documentation of Rails, in the form of code comments, has been improved in numerous places. In addition, the Ruby on Rails Guides project is the definitive source for information on major Rails components. In its first official release, the Guides page includes:

- Getting Started with Rails
- Rails Database Migrations

- Active Record Associations
- Active Record Query Interface
- Layouts and Rendering in Rails
- Action View Form Helpers
- Rails Routing from the Outside In
- Action Controller Overview
- Rails Caching
- A Guide to Testing Rails Applications
- Securing Rails Applications
- Debugging Rails Applications
- The Basics of Creating Rails Plugins

All told, the Guides provide tens of thousands of words of guidance for beginning and intermediate Rails developers.

If you want to generate these guides locally, inside your application:

```
$ rake doc:guides
```

This will put the guides inside `Rails.root/doc/guides` and you may start surfing straight away by opening `Rails.root/doc/guides/index.html` in your favorite browser.

- Major contributions from Xavier Noria and Hongli Lai.
- More information:
  - Rails Guides hackfest
  - Help improve Rails documentation on Git branch

Better integration with HTTP : Out of the box ETag support

Supporting the ETag and last modified timestamp in HTTP headers means that Rails can now send back an empty response if it gets a request for a resource that hasn't been modified lately. This allows you to check whether a response needs to be sent at all.

```
class ArticlesController < ApplicationController
  def show_with_respond_to_block
    @article = Article.find(params[:id])

    # If the request sends headers that differs from the options provided to stale?, then
    # the request is indeed stale and the respond_to block is triggered (and the options
    # to the stale? call is set on the response).
    #
    # If the request headers match, then the request is fresh and the respond_to block is
    # not triggered. Instead, the default render will occur, which will check the last-modi
    # and etag headers and conclude that it only needs to send a "304 Not Modified" instead
    # of rendering the template.
    if stale?(:last_modified => @article.published_at.utc, :etag => @article)
      respond_to do |wants|
        # normal response processing
      end
    end
  end

  def show_with_implied_render
    @article = Article.find(params[:id])

    # Sets the response headers and checks them against the request, if the request is stal
    # (i.e. no match of either etag or last-modified), then the default render of the templ
    # If the request is fresh, then the default render will return a "304 Not Modified"
    # instead of rendering the template.
    fresh_when(:last_modified => @article.published_at.utc, :etag => @article)
  end
end
```

Thread Safety

The work done to make Rails thread-safe is rolling out in Rails 2.2. Depending on your web server infrastructure, this means you can handle more requests with fewer copies of Rails in memory, leading to better server performance and higher utilization of multiple cores.

To enable multithreaded dispatching in production mode of your application, add the following line in your `config/environments/production.rb`:

```
config.threadsafe!
```

- More information :
  - Thread safety for your Rails
  - Thread safety project announcement
  - Q/A: What Thread-safe Rails Means

Active Record

There are two big additions to talk about here: transactional migrations and pooled database transactions. There's also a new (and cleaner) syntax for join table conditions, as well as a number of smaller improvements.

Transactional Migrations

Historically, multiple-step Rails migrations have been a source of trouble. If something went wrong during a migration, everything before the error changed the database and everything after the error wasn't applied. Also, the migration version was stored as having been executed, which means that it couldn't be simply rerun by `rake db:migrate:redo` after you fix the problem. Transactional migrations change this by wrapping migration steps in a DDL transaction, so that if any of them fail, the entire migration is undone. In Rails 2.2, transactional migrations are supported on PostgreSQL out of the box. The code is extensible to other database types in the future - and IBM has already extended it to support the DB2 adapter.

- Lead Contributor: Adam Wiggins
- More information:
    - DDL Transactions
    - A major milestone for DB2 on Rails

Connection Pooling

Connection pooling lets Rails distribute database requests across a pool of database connections that will grow to a maximum size (by default 5, but you can add a `pool` key to your `database.yml` to adjust this). This helps remove bottlenecks in applications that support many concurrent users. There's also a `wait_timeout` that defaults to 5 seconds before giving up. `ActiveRecord::Base.connection_pool` gives you direct access to the pool if you need it.

```
development:
  adapter: mysql
  username: root
  database: sample_development
  pool: 10
  wait_timeout: 10
```

- Lead Contributor: Nick Sieger
- More information:
    - What's New in Edge Rails: Connection Pools

Hashes for Join Table Conditions

You can now specify conditions on join tables using a hash. This is a big help if you need to query across complex joins.

```
class Photo < ActiveRecord::Base
  belongs_to :product
end

class Product < ActiveRecord::Base
  has_many :photos
end

# Get all products with copyright-free photos:
Product.all(:joins => :photos, :conditions => { :photos => { :copyright => false }})
```

- More information:
    - What's New in Edge Rails: Easy Join Table Conditions

New Dynamic Finders

Two new sets of methods have been added to Active Record's dynamic finders family.

`find_last_by_attribute` The `find_last_by_attribute` method is equivalent to `Model.last(:conditions => {:attribute => value})`

```ruby
# Get the last user who signed up from London
User.find_last_by_city('London')
```

- Lead Contributor: Emilio Tagua

`find_by_attribute!` The new bang! version of `find_by_attribute!` is equivalent to `Model.first(:conditions => {:attribute => value}) || raise ActiveRecord::RecordNotFound` Instead of returning `nil` if it can't find a matching record, this method will raise an exception if it cannot find a match.

```ruby
# Raise ActiveRecord::RecordNotFound exception if 'Moby' hasn't signed up yet!
User.find_by_name!('Moby')
```

- Lead Contributor: Josh Susser

Associations Respect Private/Protected Scope

Active Record association proxies now respect the scope of methods on the proxied object. Previously (given User has_one :account) `@user.account.private_method` would call the private method on the associated Account object. That fails in Rails 2.2; if you need this functionality, you should use `@user.account.send(:private_method)` (or make the method public instead of private or protected). Please note that if you're overriding `method_missing`, you should also override `respond_to` to match the behavior in order for associations to function normally.

- Lead Contributor: Adam Milligan
- More information:
  - Rails 2.2 Change: Private Methods on Association Proxies are Private

Other Active Record Changes

- `rake db:migrate:redo` now accepts an optional VERSION to target that specific migration to redo
- Set `config.active_record.timestamped_migrations = false` to have migrations with numeric prefix instead of UTC timestamp.
- Counter cache columns (for associations declared with `:counter_cache => true`) do not need to be initialized to zero any longer.
- `ActiveRecord::Base.human_name` for an internationalization-aware humane translation of model names

Action Controller

On the controller side, there are several changes that will help tidy up your routes. There are also some internal changes in the routing engine to lower memory usage on complex applications.

Shallow Route Nesting

Shallow route nesting provides a solution to the well-known difficulty of using deeply-nested resources. With shallow nesting, you need only supply enough information to uniquely identify the resource that you want to work with.

```
map.resources :publishers, :shallow => true do |publisher|
  publisher.resources :magazines do |magazine|
    magazine.resources :photos
  end
end
```

This will enable recognition of (among others) these routes:

```
/publishers/1           ==> publisher_path(1)
/publishers/1/magazines ==> publisher_magazines_path(1)
/magazines/2            ==> magazine_path(2)
/magazines/2/photos     ==> magazines_photos_path(2)
/photos/3               ==> photo_path(3)
```

- Lead Contributor: S. Brent Faulkner
- More information:
  - Rails Routing from the Outside In
  - What's New in Edge Rails: Shallow Routes

Method Arrays for Member or Collection Routes

You can now supply an array of methods for new member or collection routes. This removes the annoyance of having to define a route as accepting any verb as soon as you need it to handle more than one. With Rails 2.2, this is a legitimate route declaration:

```
map.resources :photos, :collection => { :search => [:get, :post] }
```

- Lead Contributor: Brennan Dunn

Resources With Specific Actions

By default, when you use `map.resources` to create a route, Rails generates routes for seven default actions (index, show, create, new, edit, update, and destroy). But each of these routes takes up memory in your application, and causes Rails to generate additional routing logic. Now you can use the `:only` and `:except` options to fine-tune the routes that Rails will generate for resources.

You can supply a single action, an array of actions, or the special `:all` or `:none` options. These options are inherited by nested resources.

```
map.resources :photos, :only => [:index, :show]
map.resources :products, :except => :destroy
```

- Lead Contributor: Tom Stuart

Other Action Controller Changes

- You can now easily show a custom error page for exceptions raised while routing a request.
- The HTTP Accept header is disabled by default now. You should prefer the use of formatted URLs (such as `/customers/1.xml`) to indicate the format that you want. If you need the Accept headers, you can turn them back on with `config.action_controller.use_accept_header = true`.
- Benchmarking numbers are now reported in milliseconds rather than tiny fractions of seconds
- Rails now supports HTTP-only cookies (and uses them for sessions), which help mitigate some cross-site scripting risks in newer browsers.
- `redirect_to` now fully supports URI schemes (so, for example, you can redirect to a svn‘ssh: URI).
- `render` now supports a `:js` option to render plain vanilla JavaScript with the right mime type.
- Request forgery protection has been tightened up to apply to HTML-formatted content requests only.
- Polymorphic URLs behave more sensibly if a passed parameter is nil. For example, calling `polymorphic_path([@project, @date, @area])` with a nil date will give you `project_area_path`.

Action View

- `javascript_include_tag` and `stylesheet_link_tag` support a new `:recursive` option to be used along with `:all`, so that you can load an entire tree of files with a single line of code.
- The included Prototype JavaScript library has been upgraded to version 1.6.0.3.
- `RJS#page.reload` to reload the browser's current location via JavaScript
- The `atom_feed` helper now takes an `:instruct` option to let you insert XML processing instructions.

Action Mailer

Action Mailer now supports mailer layouts. You can make your HTML emails as pretty as your in-browser views by supplying an appropriately-named layout - for example, the `CustomerMailer` class expects to use `layouts/customer_mailer.html.erb`.

- More information:
    - What's New in Edge Rails: Mailer Layouts

Action Mailer now offers built-in support for GMail's SMTP servers, by turning on STARTTLS automatically. This requires Ruby 1.8.7 to be installed.

Active Support

Active Support now offers built-in memoization for Rails applications, the `each_with_object` method, prefix support on delegates, and various other new utility methods.

Memoization

Memoization is a pattern of initializing a method once and then stashing its value away for repeat use. You've probably used this pattern in your own applications:

```
def full_name
  @full_name ||= "#{first_name} #{last_name}"
end
```

Memoization lets you handle this task in a declarative fashion:

```
extend ActiveSupport::Memoizable

def full_name
  "#{first_name} #{last_name}"
end
memoize :full_name
```

Other features of memoization include `unmemoize`, `unmemoize_all`, and `memoize_all` to turn memoization on or off.

- Lead Contributor: Josh Peek
- More information:
    - What's New in Edge Rails: Easy Memoization
    - Memo-what? A Guide to Memoization

each_with_object

The `each_with_object` method provides an alternative to `inject`, using a method backported from Ruby 1.9. It iterates over a collection, passing the current element and the memo into the block.

```
%w(foo bar).each_with_object({}) { |str, hsh| hsh[str] = str.upcase } # => {'foo' => 'FOO',
```

Lead Contributor: Adam Keys

Delegates With Prefixes

If you delegate behavior from one class to another, you can now specify a prefix that will be used to identify the delegated methods. For example:

```ruby
class Vendor < ActiveRecord::Base
  has_one :account
  delegate :email, :password, :to => :account, :prefix => true
end
```

This will produce delegated methods `vendor#account_email` and `vendor#account_password`. You can also specify a custom prefix:

```ruby
class Vendor < ActiveRecord::Base
  has_one :account
  delegate :email, :password, :to => :account, :prefix => :owner
end
```

This will produce delegated methods `vendor#owner_email` and `vendor#owner_password`.

Lead Contributor: Daniel Schierbeck

Other Active Support Changes

- Extensive updates to `ActiveSupport::Multibyte`, including Ruby 1.9 compatibility fixes.
- The addition of `ActiveSupport::Rescuable` allows any class to mix in the `rescue_from` syntax.
- `past?`, `today?` and `future?` for `Date` and `Time` classes to facilitate date/time comparisons.
- `Array#second` through `Array#fifth` as aliases for `Array#[1]` through `Array#[4]`
- `Enumerable#many?` to encapsulate `collection.size > 1`
- `Inflector#parameterize` produces a URL-ready version of its input, for use in `to_param`.
- `Time#advance` recognizes fractional days and weeks, so you can do `1.7.weeks.ago`, `1.5.hours.since`, and so on.
- The included TzInfo library has been upgraded to version 0.3.12.
- `ActiveSupport::StringInquirer` gives you a pretty way to test for equality in strings: `ActiveSupport::StringInquirer.new("abc").abc? => true`

Railties

In Railties (the core code of Rails itself) the biggest changes are in the `config.gems` mechanism.

config.gems

To avoid deployment issues and make Rails applications more self-contained, it's possible to place copies of all of the gems that your Rails application requires in `/vendor/gems`. This capability first appeared in Rails 2.1, but it's much more flexible and robust in Rails 2.2, handling complicated dependencies between gems. Gem management in Rails includes these commands:

- `config.gem _gem_name_` in your `config/environment.rb` file
- `rake gems` to list all configured gems, as well as whether they (and their dependencies) are installed, frozen, or framework (framework gems are those loaded by Rails before the gem dependency code is executed; such gems cannot be frozen)
- `rake gems:install` to install missing gems to the computer
- `rake gems:unpack` to place a copy of the required gems into `/vendor/gems`
- `rake gems:unpack:dependencies` to get copies of the required gems and their dependencies into `/vendor/gems`
- `rake gems:build` to build any missing native extensions
- `rake gems:refresh_specs` to bring vendored gems created with Rails 2.1 into alignment with the Rails 2.2 way of storing them

You can unpack or install a single gem by specifying `GEM=_gem_name_` on the command line.

- Lead Contributor: Matt Jones
- More information:
  - What's New in Edge Rails: Gem Dependencies
  - Rails 2.1.2 and 2.2RC1: Update Your RubyGems
  - Detailed discussion on Lighthouse

Other Railties Changes

- If you're a fan of the Thin web server, you'll be happy to know that `script/server` now supports Thin directly.
- `script/plugin install &lt;plugin&gt; -r &lt;revision&gt;` now works with git-based as well as svn-based plugins.
- `script/console` now supports a `--debugger` option
- Instructions for setting up a continuous integration server to build Rails itself are included in the Rails source
- `rake notes:custom ANNOTATION=MYFLAG` lets you list out custom annotations.
- Wrapped `Rails.env` in `StringInquirer` so you can do `Rails.env.development?`
- To eliminate deprecation warnings and properly handle gem dependencies, Rails now requires rubygems 1.3.1 or higher.

Deprecated

A few pieces of older code are deprecated in this release:

- `Rails::SecretKeyGenerator` has been replaced by `ActiveSupport::SecureRandom`
- `render_component` is deprecated. There's a render_components plugin available if you need this functionality.
- Implicit local assignments when rendering partials has been deprecated.

  ```
  def partial_with_implicit_local_assignment
    @customer = Customer.new("Marcel")
    render :partial => "customer"
  end
  ```

  Previously the above code made available a local variable called `customer` inside the partial 'customer'. You should explicitly pass all the variables via :locals hash now.
- `country_select` has been removed. See the deprecation page for more information and a plugin replacement.
- `ActiveRecord::Base.allow_concurrency` no longer has any effect.
- `ActiveRecord::Errors.default_error_messages` has been deprecated in favor of `I18n.translate('activerecord.errors.messages')`
- The `%s` and `%d` interpolation syntax for internationalization is deprecated.
- `String#chars` has been deprecated in favor of `String#mb_chars`.
- Durations of fractional months or fractional years are deprecated. Use Ruby's core `Date` and `Time` class arithmetic instead.
- `Request#relative_url_root` is deprecated. Use `ActionController::Base.relative_url_root` instead.

Credits

Release notes compiled by Mike Gunderloy