DO NOT READ THIS FILE ON GITHUB, GUIDES ARE PUBLISHED ON https://guides.rubyonrails.org.

Ruby on Rails 2.3 Release Notes

Rails 2.3 delivers a variety of new and improved features, including pervasive Rack integration, refreshed support for Rails Engines, nested transactions for Active Record, dynamic and default scopes, unified rendering, more efficient routing, application templates, and quiet backtraces. 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 or review the `CHANGELOG` files for the individual Rails components.

Application Architecture

There are two major changes in the architecture of Rails applications: complete integration of the Rack modular web server interface, and renewed support for Rails Engines.

Rack Integration

Rails has now broken with its CGI past, and uses Rack everywhere. This required and resulted in a tremendous number of internal changes (but if you use CGI, don't worry; Rails now supports CGI through a proxy interface). Still, this is a major change to Rails internals. After upgrading to 2.3, you should test on your local environment and your production environment. Some things to test:

- Sessions
- Cookies
- File uploads
- JSON/XML APIs

Here's a summary of the rack-related changes:

- `script/server` has been switched to use Rack, which means it supports any Rack compatible server. `script/server` will also pick up a rackup configuration file if one exists. By default, it will look for a `config.ru` file, but you can override this with the `-c` switch.
- The FCGI handler goes through Rack.
- `ActionController::Dispatcher` maintains its own default middleware stack. Middlewares can be injected in, reordered, and removed. The stack is compiled into a chain on boot. You can configure the middleware stack in `environment.rb`.
- The `rake middleware` task has been added to inspect the middleware stack. This is useful for debugging the order of the middleware stack.
- The integration test runner has been modified to execute the entire middleware and application stack. This makes integration tests perfect for testing Rack middleware.
- `ActionController::CGIHandler` is a backwards compatible CGI wrapper around Rack. The `CGIHandler` is meant to take an old CGI object and convert its environment information into a Rack compatible form.
- `CgiRequest` and `CgiResponse` have been removed.
- Session stores are now lazy loaded. If you never access the session object during a request, it will never attempt to load the session data (parse the cookie, load the data from memcache, or lookup an Active Record object).
- You no longer need to use `CGI::Cookie.new` in your tests for setting a cookie value. Assigning a `String` value to request.cookies["foo"] now sets the cookie as expected.

- `CGI::Session::CookieStore` has been replaced by `ActionController::Session::CookieStore`.
- `CGI::Session::MemCacheStore` has been replaced by `ActionController::Session::MemCacheStore`.
- `CGI::Session::ActiveRecordStore` has been replaced by `ActiveRecord::SessionStore`.
- You can still change your session store with `ActionController::Base.session_store = :active_record_store`.
- Default sessions options are still set with `ActionController::Base.session = { :key => "..." }`. However, the `:session_domain` option has been renamed to `:domain`.
- The mutex that normally wraps your entire request has been moved into middleware, `ActionController::Lock`.
- `ActionController::AbstractRequest` and `ActionController::Request` have been unified. The new `ActionController::Request` inherits from `Rack::Request`. This affects access to `response.headers['type']` in test requests. Use `response.content_type` instead.
- `ActiveRecord::QueryCache` middleware is automatically inserted onto the middleware stack if `ActiveRecord` has been loaded. This middleware sets up and flushes the per-request Active Record query cache.
- The Rails router and controller classes follow the Rack spec. You can call a controller directly with `SomeController.call(env)`. The router stores the routing parameters in `rack.routing_args`.
- `ActionController::Request` inherits from `Rack::Request`.
- Instead of `config.action_controller.session = { :session_key => 'foo', ...` use `config.action_controller.session = { :key => 'foo', ...`.
- Using the `ParamsParser` middleware preprocesses any XML, JSON, or YAML requests so they can be read normally with any `Rack::Request` object after it.

Renewed Support for Rails Engines

After some versions without an upgrade, Rails 2.3 offers some new features for Rails Engines (Rails applications that can be embedded within other applications). First, routing files in engines are automatically loaded and reloaded now, just like your `routes.rb` file (this also applies to routing files in other plugins). Second, if your plugin has an app folder, then app/[models|controllers|helpers] will automatically be added to the Rails load path. Engines also support adding view paths now, and Action Mailer as well as Action View will use views from engines and other plugins.

Documentation

The Ruby on Rails guides project has published several additional guides for Rails 2.3. In addition, a separate site maintains updated copies of the Guides for Edge Rails. Other documentation efforts include a relaunch of the Rails wiki and early planning for a Rails Book.

- More Information: Rails Documentation Projects

Ruby 1.9.1 Support

Rails 2.3 should pass all of its own tests whether you are running on Ruby 1.8 or the now-released Ruby 1.9.1. You should be aware, though, that moving to 1.9.1 entails checking all of the data adapters, plugins, and other code that you depend on for Ruby 1.9.1 compatibility, as well as Rails core.

Active Record

Active Record gets quite a number of new features and bug fixes in Rails 2.3. The highlights include nested attributes, nested transactions, dynamic and default scopes, and batch processing.

Nested Attributes

Active Record can now update the attributes on nested models directly, provided you tell it to do so:

```
class Book < ActiveRecord::Base
  has_one :author
  has_many :pages

  accepts_nested_attributes_for :author, :pages
end
```

Turning on nested attributes enables a number of things: automatic (and atomic) saving of a record together with its associated children, child-aware validations, and support for nested forms (discussed later).

You can also specify requirements for any new records that are added via nested attributes using the `:reject_if` option:

```
accepts_nested_attributes_for :author,
  :reject_if => proc { |attributes| attributes['name'].blank? }
```

- Lead Contributor: Eloy Duran
- More Information: Nested Model Forms

Nested Transactions

Active Record now supports nested transactions, a much-requested feature. Now you can write code like this:

```
User.transaction do
  User.create(:username => 'Admin')
  User.transaction(:requires_new => true) do
    User.create(:username => 'Regular')
    raise ActiveRecord::Rollback
  end
end

User.find(:all)  # => Returns only Admin
```

Nested transactions let you roll back an inner transaction without affecting the state of the outer transaction. If you want a transaction to be nested, you must explicitly add the `:requires_new` option; otherwise, a nested transaction simply becomes part of the parent transaction (as it does currently on Rails 2.2). Under the covers, nested transactions are using savepoints so they're supported even on databases that don't have true nested transactions. There is also a bit of magic going on to make these transactions play well with transactional fixtures during testing.

- Lead Contributors: Jonathan Viney and Hongli Lai

Dynamic Scopes

You know about dynamic finders in Rails (which allow you to concoct methods like `find_by_color_and_flavor` on the fly) and named scopes (which allow you to encapsulate reusable query conditions into friendly names like

`currently_active`). Well, now you can have dynamic scope methods. The idea is to put together syntax that allows filtering on the fly *and* method chaining. For example:

```
Order.scoped_by_customer_id(12)
Order.scoped_by_customer_id(12).find(:all,
  :conditions => "status = 'open'")
Order.scoped_by_customer_id(12).scoped_by_status("open")
```

There's nothing to define to use dynamic scopes: they just work.

- Lead Contributor: Yaroslav Markin
- More Information: What's New in Edge Rails: Dynamic Scope Methods

Default Scopes

Rails 2.3 will introduce the notion of *default scopes* similar to named scopes, but applying to all named scopes or find methods within the model. For example, you can write `default_scope :order => 'name ASC'` and any time you retrieve records from that model they'll come out sorted by name (unless you override the option, of course).

- Lead Contributor: Paweł Kondzior
- More Information: What's New in Edge Rails: Default Scoping

Batch Processing

You can now process large numbers of records from an Active Record model with less pressure on memory by using `find_in_batches`:

```
Customer.find_in_batches(:conditions => {:active => true}) do |customer_group|
  customer_group.each { |customer| customer.update_account_balance! }
end
```

You can pass most of the `find` options into `find_in_batches`. However, you cannot specify the order that records will be returned in (they will always be returned in ascending order of primary key, which must be an integer), or use the `:limit` option. Instead, use the `:batch_size` option, which defaults to 1000, to set the number of records that will be returned in each batch.

The new `find_each` method provides a wrapper around `find_in_batches` that returns individual records, with the find itself being done in batches (of 1000 by default):

```
Customer.find_each do |customer|
  customer.update_account_balance!
end
```

Note that you should only use this method for batch processing: for small numbers of records (less than 1000), you should just use the regular find methods with your own loop.

- More Information (at that point the convenience method was called just `each`):
  - Rails 2.3: Batch Finding
  - What's New in Edge Rails: Batched Find

Multiple Conditions for Callbacks

When using Active Record callbacks, you can now combine `:if` and `:unless` options on the same callback, and supply multiple conditions as an array:

```
before_save :update_credit_rating, :if => :active,
  :unless => [:admin, :cash_only]
```

- Lead Contributor: L. Caviola

Find with having

Rails now has a `:having` option on find (as well as on `has_many` and `has_and_belongs_to_many` associations) for filtering records in grouped finds. As those with heavy SQL backgrounds know, this allows filtering based on grouped results:

```
developers = Developer.find(:all, :group => "salary",
  :having => "sum(salary) > 10000", :select => "salary")
```

- Lead Contributor: Emilio Tagua

Reconnecting MySQL Connections

MySQL supports a reconnect flag in its connections - if set to true, then the client will try reconnecting to the server before giving up in case of a lost connection. You can now set `reconnect = true` for your MySQL connections in `database.yml` to get this behavior from a Rails application. The default is `false`, so the behavior of existing applications doesn't change.

- Lead Contributor: Dov Murik
- More information:
  - Controlling Automatic Reconnection Behavior
  - MySQL auto-reconnect revisited

Other Active Record Changes

- An extra `AS` was removed from the generated SQL for `has_and_belongs_to_many` preloading, making it work better for some databases.
- `ActiveRecord::Base#new_record?` now returns `false` rather than `nil` when confronted with an existing record.
- A bug in quoting table names in some `has_many :through` associations was fixed.
- You can now specify a particular timestamp for `updated_at` timestamps: `cust = Customer.create(:name => "ABC Industries", :updated_at => 1.day.ago)`
- Better error messages on failed `find_by_attribute!` calls.
- Active Record's `to_xml` support gets just a little bit more flexible with the addition of a `:camelize` option.
- A bug in canceling callbacks from `before_update` or `before_create` was fixed.
- Rake tasks for testing databases via JDBC have been added.
- `validates_length_of` will use a custom error message with the `:in` or `:within` options (if one is supplied).
- Counts on scoped selects now work properly, so you can do things like `Account.scoped(:select => "DISTINCT credit_limit").count`.
- `ActiveRecord::Base#invalid?` now works as the opposite of `ActiveRecord::Base#valid?`.

Action Controller

Action Controller rolls out some significant changes to rendering, as well as improvements in routing and other areas, in this release.

Unified Rendering

`ActionController::Base#render` is a lot smarter about deciding what to render. Now you can just tell it what to render and expect to get the right results. In older versions of Rails, you often need to supply explicit information to render:

```
render :file => '/tmp/random_file.erb'
render :template => 'other_controller/action'
render :action => 'show'
```

Now in Rails 2.3, you can just supply what you want to render:

```
render '/tmp/random_file.erb'
render 'other_controller/action'
render 'show'
render :show
```

Rails chooses between file, template, and action depending on whether there is a leading slash, an embedded slash, or no slash at all in what's to be rendered. Note that you can also use a symbol instead of a string when rendering an action. Other rendering styles (`:inline` , `:text` , `:update` , `:nothing` , `:json` , `:xml` , `:js`) still require an explicit option.

Application Controller Renamed

If you're one of the people who has always been bothered by the special-case naming of `application.rb` , rejoice! It's been reworked to be `application_controller.rb` in Rails 2.3. In addition, there's a new rake task, `rake rails:update:application_controller` to do this automatically for you - and it will be run as part of the normal `rake rails:update` process.

- More Information:
  - The Death of Application.rb
  - What's New in Edge Rails: Application.rb Duality is no More

HTTP Digest Authentication Support

Rails now has built-in support for HTTP digest authentication. To use it, you call `authenticate_or_request_with_http_digest` with a block that returns the user's password (which is then hashed and compared against the transmitted credentials):

```
class PostsController < ApplicationController
  Users = {"dhh" => "secret"}
  before_filter :authenticate

  def secret
    render :text => "Password Required!"
```

```
  end

  private
  def authenticate
    realm = "Application"
    authenticate_or_request_with_http_digest(realm) do |name|
      Users[name]
    end
  end
end
```

- Lead Contributor: Gregg Kellogg
- More Information: What's New in Edge Rails: HTTP Digest Authentication

More Efficient Routing

There are a couple of significant routing changes in Rails 2.3. The `formatted_` route helpers are gone, in favor just passing in `:format` as an option. This cuts down the route generation process by 50% for any resource - and can save a substantial amount of memory (up to 100MB on large applications). If your code uses the `formatted_` helpers, it will still work for the time being - but that behavior is deprecated and your application will be more efficient if you rewrite those routes using the new standard. Another big change is that Rails now supports multiple routing files, not just `routes.rb` . You can use `RouteSet#add_configuration_file` to bring in more routes at any time - without clearing the currently loaded routes. While this change is most useful for Engines, you can use it in any application that needs to load routes in batches.

- Lead Contributors: Aaron Batalion

Rack-based Lazy-loaded Sessions

A big change pushed the underpinnings of Action Controller session storage down to the Rack level. This involved a good deal of work in the code, though it should be completely transparent to your Rails applications (as a bonus, some icky patches around the old CGI session handler got removed). It's still significant, though, for one simple reason: non-Rails Rack applications have access to the same session storage handlers (and therefore the same session) as your Rails applications. In addition, sessions are now lazy-loaded (in line with the loading improvements to the rest of the framework). This means that you no longer need to explicitly disable sessions if you don't want them; just don't refer to them and they won't load.

MIME Type Handling Changes

There are a couple of changes to the code for handling MIME types in Rails. First, `MIME::Type` now implements the `=~` operator, making things much cleaner when you need to check for the presence of a type that has synonyms:

```
if content_type && Mime::JS =~ content_type
  # do something cool
end

Mime::JS =~ "text/javascript"        => true
Mime::JS =~ "application/javascript" => true
```

The other change is that the framework now uses the `Mime::JS` when checking for JavaScript in various spots, making it handle those alternatives cleanly.

- Lead Contributor: Seth Fitzsimmons

Optimization of `respond_to`

In some of the first fruits of the Rails-Merb team merger, Rails 2.3 includes some optimizations for the `respond_to` method, which is of course heavily used in many Rails applications to allow your controller to format results differently based on the MIME type of the incoming request. After eliminating a call to `method_missing` and some profiling and tweaking, we're seeing an 8% improvement in the number of requests per second served with a simple `respond_to` that switches between three formats. The best part? No change at all required to the code of your application to take advantage of this speedup.

Improved Caching Performance

Rails now keeps a per-request local cache of read from the remote cache stores, cutting down on unnecessary reads and leading to better site performance. While this work was originally limited to `MemCacheStore`, it is available to any remote store than implements the required methods.

- Lead Contributor: Nahum Wild

Localized Views

Rails can now provide localized views, depending on the locale that you have set. For example, suppose you have a `Posts` controller with a `show` action. By default, this will render `app/views/posts/show.html.erb`. But if you set `I18n.locale = :da`, it will render `app/views/posts/show.da.html.erb`. If the localized template isn't present, the undecorated version will be used. Rails also includes `I18n#available_locales` and `I18n::SimpleBackend#available_locales`, which return an array of the translations that are available in the current Rails project.

In addition, you can use the same scheme to localize the rescue files in the public directory: `public/500.da.html` or `public/404.en.html` work, for example.

Partial Scoping for Translations

A change to the translation API makes things easier and less repetitive to write key translations within partials. If you call `translate(".foo")` from the `people/index.html.erb` template, you'll actually be calling `I18n.translate("people.index.foo")` If you don't prepend the key with a period, then the API doesn't scope, just as before.

Other Action Controller Changes

- ETag handling has been cleaned up a bit: Rails will now skip sending an ETag header when there's no body to the response or when sending files with `send_file`.
- The fact that Rails checks for IP spoofing can be a nuisance for sites that do heavy traffic with cell phones, because their proxies don't generally set things up right. If that's you, you can now set `ActionController::Base.ip_spoofing_check = false` to disable the check entirely.
- `ActionController::Dispatcher` now implements its own middleware stack, which you can see by running `rake middleware`.
- Cookie sessions now have persistent session identifiers, with API compatibility with the server-side stores.
- You can now use symbols for the `:type` option of `send_file` and `send_data`, like this: `send_file("fabulous.png", :type => :png)`.
- The `:only` and `:except` options for `map.resources` are no longer inherited by nested resources.
- The bundled memcached client has been updated to version 1.6.4.99.

- The `expires_in`, `stale?`, and `fresh_when` methods now accept a `:public` option to make them work well with proxy caching.
- The `:requirements` option now works properly with additional RESTful member routes.
- Shallow routes now properly respect namespaces.
- `polymorphic_url` does a better job of handling objects with irregular plural names.

Action View

Action View in Rails 2.3 picks up nested model forms, improvements to `render`, more flexible prompts for the date select helpers, and a speedup in asset caching, among other things.

Nested Object Forms

Provided the parent model accepts nested attributes for the child objects (as discussed in the Active Record section), you can create nested forms using `form_for` and `field_for`. These forms can be nested arbitrarily deep, allowing you to edit complex object hierarchies on a single view without excessive code. For example, given this model:

```
class Customer < ActiveRecord::Base
  has_many :orders

  accepts_nested_attributes_for :orders, :allow_destroy => true
end
```

You can write this view in Rails 2.3:

```
<% form_for @customer do |customer_form| %>
  <div>
    <%= customer_form.label :name, 'Customer Name:' %>
    <%= customer_form.text_field :name %>
  </div>

  <!-- Here we call fields_for on the customer_form builder instance.
   The block is called for each member of the orders collection. -->
  <% customer_form.fields_for :orders do |order_form| %>
    <p>
      <div>
        <%= order_form.label :number, 'Order Number:' %>
        <%= order_form.text_field :number %>
      </div>

  <!-- The allow_destroy option in the model enables deletion of
   child records. -->
      <% unless order_form.object.new_record? %>
        <div>
          <%= order_form.label :_delete, 'Remove:' %>
          <%= order_form.check_box :_delete %>
        </div>
      <% end %>
    </p>
  <% end %>
```

```
  <%= customer_form.submit %>
<% end %>
```

- Lead Contributor: Eloy Duran
- More Information:
  - Nested Model Forms
  - complex-form-examples
  - What's New in Edge Rails: Nested Object Forms

Smart Rendering of Partials

The render method has been getting smarter over the years, and it's even smarter now. If you have an object or a collection and an appropriate partial, and the naming matches up, you can now just render the object and things will work. For example, in Rails 2.3, these render calls will work in your view (assuming sensible naming):

```
# Equivalent of render :partial => 'articles/_article',
# :object => @article
render @article

# Equivalent of render :partial => 'articles/_article',
# :collection => @articles
render @articles
```

- More Information: What's New in Edge Rails: render Stops Being High-Maintenance

Prompts for Date Select Helpers

In Rails 2.3, you can supply custom prompts for the various date select helpers (`date_select` , `time_select` , and `datetime_select`), the same way you can with collection select helpers. You can supply a prompt string or a hash of individual prompt strings for the various components. You can also just set `:prompt` to `true` to use the custom generic prompt:

```
select_datetime(DateTime.now, :prompt => true)

select_datetime(DateTime.now, :prompt => "Choose date and time")

select_datetime(DateTime.now, :prompt =>
  {:day => 'Choose day', :month => 'Choose month',
   :year => 'Choose year', :hour => 'Choose hour',
   :minute => 'Choose minute'})
```

- Lead Contributor: Sam Oliver

AssetTag Timestamp Caching

You're likely familiar with Rails' practice of adding timestamps to static asset paths as a "cache buster". This helps ensure that stale copies of things like images and stylesheets don't get served out of the user's browser cache when you change them on the server. You can now modify this behavior with the `cache_asset_timestamps` configuration option for Action View. If you enable the cache, then Rails will calculate the timestamp once when it first serves an asset, and save that value. This means fewer (expensive) file system calls to serve static assets - but it

also means that you can't modify any of the assets while the server is running and expect the changes to get picked up by clients.

Asset Hosts as Objects

Asset hosts get more flexible in edge Rails with the ability to declare an asset host as a specific object that responds to a call. This allows you to implement any complex logic you need in your asset hosting.

- More Information: asset-hosting-with-minimum-ssl

grouped_options_for_select Helper Method

Action View already had a bunch of helpers to aid in generating select controls, but now there's one more: `grouped_options_for_select`. This one accepts an array or hash of strings, and converts them into a string of `option` tags wrapped with `optgroup` tags. For example:

```
grouped_options_for_select([["Hats", ["Baseball Cap","Cowboy Hat"]]],
  "Cowboy Hat", "Choose a product...")
```

returns

```
<option value="">Choose a product...</option>
<optgroup label="Hats">
  <option value="Baseball Cap">Baseball Cap</option>
  <option selected="selected" value="Cowboy Hat">Cowboy Hat</option>
</optgroup>
```

Disabled Option Tags for Form Select Helpers

The form select helpers (such as `select` and `options_for_select`) now support a `:disabled` option, which can take a single value or an array of values to be disabled in the resulting tags:

```
select(:post, :category, Post::CATEGORIES, :disabled => 'private')
```

returns

```
<select name="post[category]">
<option>story</option>
<option>joke</option>
<option>poem</option>
<option disabled="disabled">private</option>
</select>
```

You can also use an anonymous function to determine at runtime which options from collections will be selected and/or disabled:

```
options_from_collection_for_select(@product.sizes, :name, :id, :disabled =>
lambda{|size| size.out_of_stock?})
```

- Lead Contributor: Tekin Suleyman

- More Information: New in rails 2.3 - disabled option tags and lambdas for selecting and disabling options from collections

A Note About Template Loading

Rails 2.3 includes the ability to enable or disable cached templates for any particular environment. Cached templates give you a speed boost because they don't check for a new template file when they're rendered - but they also mean that you can't replace a template "on the fly" without restarting the server.

In most cases, you'll want template caching to be turned on in production, which you can do by making a setting in your `production.rb` file:

```
config.action_view.cache_template_loading = true
```

This line will be generated for you by default in a new Rails 2.3 application. If you've upgraded from an older version of Rails, Rails will default to caching templates in production and test but not in development.

Other Action View Changes

- Token generation for CSRF protection has been simplified; now Rails uses a simple random string generated by `ActiveSupport::SecureRandom` rather than mucking around with session IDs.
- `auto_link` now properly applies options (such as `:target` and `:class`) to generated e-mail links.
- The `autolink` helper has been refactored to make it a bit less messy and more intuitive.
- `current_page?` now works properly even when there are multiple query parameters in the URL.

Active Support

Active Support has a few interesting changes, including the introduction of `Object#try`.

Object#try

A lot of folks have adopted the notion of using try() to attempt operations on objects. It's especially helpful in views where you can avoid nil-checking by writing code like `<%= @person.try(:name) %>`. Well, now it's baked right into Rails. As implemented in Rails, it raises `NoMethodError` on private methods and always returns `nil` if the object is nil.

- More Information: try()

Object#tap Backport

`Object#tap` is an addition to Ruby 1.9 and 1.8.7 that is similar to the `returning` method that Rails has had for a while: it yields to a block, and then returns the object that was yielded. Rails now includes code to make this available under older versions of Ruby as well.

Swappable Parsers for XMLmini

The support for XML parsing in Active Support has been made more flexible by allowing you to swap in different parsers. By default, it uses the standard REXML implementation, but you can easily specify the faster LibXML or Nokogiri implementations for your own applications, provided you have the appropriate gems installed:

```
XmlMini.backend = 'LibXML'
```

- Lead Contributor: Bart ten Brinke

- Lead Contributor: Aaron Patterson

Fractional seconds for TimeWithZone

The `Time` and `TimeWithZone` classes include an `xmlschema` method to return the time in an XML-friendly string. As of Rails 2.3, `TimeWithZone` supports the same argument for specifying the number of digits in the fractional second part of the returned string that `Time` does:

```
Time.zone.now.xmlschema(6) # => "2009-01-16T13:00:06.13653Z"
```

- Lead Contributor: Nicholas Dainty

JSON Key Quoting

If you look up the spec on the "json.org" site, you'll discover that all keys in a JSON structure must be strings, and they must be quoted with double quotes. Starting with Rails 2.3, we do the right thing here, even with numeric keys.

Other Active Support Changes

- You can use `Enumerable#none?` to check that none of the elements match the supplied block.
- If you're using Active Support delegates the new `:allow_nil` option lets you return `nil` instead of raising an exception when the target object is nil.
- `ActiveSupport::OrderedHash` : now implements `each_key` and `each_value` .
- `ActiveSupport::MessageEncryptor` provides a simple way to encrypt information for storage in an untrusted location (like cookies).
- Active Support's `from_xml` no longer depends on XmlSimple. Instead, Rails now includes its own XmlMini implementation, with just the functionality that it requires. This lets Rails dispense with the bundled copy of XmlSimple that it's been carting around.
- If you memoize a private method, the result will now be private.
- `String#parameterize` accepts an optional separator: `"Quick Brown Fox".parameterize('_') => "quick_brown_fox"` .
- `number_to_phone` accepts 7-digit phone numbers now.
- `ActiveSupport::Json.decode` now handles `\u0000` style escape sequences.

Railties

In addition to the Rack changes covered above, Railties (the core code of Rails itself) sports a number of significant changes, including Rails Metal, application templates, and quiet backtraces.

Rails Metal

Rails Metal is a new mechanism that provides superfast endpoints inside of your Rails applications. Metal classes bypass routing and Action Controller to give you raw speed (at the cost of all the things in Action Controller, of course). This builds on all of the recent foundation work to make Rails a Rack application with an exposed middleware stack. Metal endpoints can be loaded from your application or from plugins.

- More Information:
  - Introducing Rails Metal
  - Rails Metal: a micro-framework with the power of Rails
  - Metal: Super-fast Endpoints within your Rails Apps
  - What's New in Edge Rails: Rails Metal

Application Templates

Rails 2.3 incorporates Jeremy McAnally's rg application generator. What this means is that we now have template-based application generation built right into Rails; if you have a set of plugins you include in every application (among many other use cases), you can just set up a template once and use it over and over again when you run the `rails` command. There's also a rake task to apply a template to an existing application:

```
$ rake rails:template LOCATION=~/template.rb
```

This will layer the changes from the template on top of whatever code the project already contains.

- Lead Contributor: Jeremy McAnally
- More Info:Rails templates

Quieter Backtraces

Building on thoughtbot's Quiet Backtrace plugin, which allows you to selectively remove lines from `Test::Unit` backtraces, Rails 2.3 implements `ActiveSupport::BacktraceCleaner` and `Rails::BacktraceCleaner` in core. This supports both filters (to perform regex-based substitutions on backtrace lines) and silencers (to remove backtrace lines entirely). Rails automatically adds silencers to get rid of the most common noise in a new application, and builds a `config/backtrace_silencers.rb` file to hold your own additions. This feature also enables prettier printing from any gem in the backtrace.

Faster Boot Time in Development Mode with Lazy Loading/Autoload

Quite a bit of work was done to make sure that bits of Rails (and its dependencies) are only brought into memory when they're actually needed. The core frameworks - Active Support, Active Record, Action Controller, Action Mailer, and Action View - are now using `autoload` to lazy-load their individual classes. This work should help keep the memory footprint down and improve overall Rails performance.

You can also specify (by using the new `preload_frameworks` option) whether the core libraries should be autoloaded at startup. This defaults to `false` so that Rails autoloads itself piece-by-piece, but there are some circumstances where you still need to bring in everything at once - Passenger and JRuby both want to see all of Rails loaded together.

rake gem Task Rewrite

The internals of the various `rake gem` tasks have been substantially revised, to make the system work better for a variety of cases. The gem system now knows the difference between development and runtime dependencies, has a more robust unpacking system, gives better information when querying for the status of gems, and is less prone to "chicken and egg" dependency issues when you're bringing things up from scratch. There are also fixes for using gem commands under JRuby and for dependencies that try to bring in external copies of gems that are already vendored.

- Lead Contributor: David Dollar

Other Railties Changes

- The instructions for updating a CI server to build Rails have been updated and expanded.
- Internal Rails testing has been switched from `Test::Unit::TestCase` to `ActiveSupport::TestCase`, and the Rails core requires Mocha to test.
- The default `environment.rb` file has been decluttered.
- The dbconsole script now lets you use an all-numeric password without crashing.

- `Rails.root` now returns a `Pathname` object, which means you can use it directly with the `join` method to clean up existing code that uses `File.join`.
- Various files in /public that deal with CGI and FCGI dispatching are no longer generated in every Rails application by default (you can still get them if you need them by adding `--with-dispatchers` when you run the `rails` command, or add them later with `rake rails:update:generate_dispatchers`).
- Rails Guides have been converted from AsciiDoc to Textile markup.
- Scaffolded views and controllers have been cleaned up a bit.
- `script/server` now accepts a `--path` argument to mount a Rails application from a specific path.
- If any configured gems are missing, the gem rake tasks will skip loading much of the environment. This should solve many of the "chicken-and-egg" problems where rake gems:install couldn't run because gems were missing.
- Gems are now unpacked exactly once. This fixes issues with gems (hoe, for instance) which are packed with read-only permissions on the files.

Deprecated

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

- If you're one of the (fairly rare) Rails developers who deploys in a fashion that depends on the inspector, reaper, and spawner scripts, you'll need to know that those scripts are no longer included in core Rails. If you need them, you'll be able to pick up copies via the irs_process_scripts plugin.
- `render_component` goes from "deprecated" to "nonexistent" in Rails 2.3. If you still need it, you can install the render_component plugin.
- Support for Rails components has been removed.
- If you were one of the people who got used to running `script/performance/request` to look at performance based on integration tests, you need to learn a new trick: that script has been removed from core Rails now. There's a new request_profiler plugin that you can install to get the exact same functionality back.
- `ActionController::Base#session_enabled?` is deprecated because sessions are lazy-loaded now.
- The `:digest` and `:secret` options to `protect_from_forgery` are deprecated and have no effect.
- Some integration test helpers have been removed. `response.headers["Status"]` and `headers["Status"]` will no longer return anything. Rack does not allow "Status" in its return headers. However you can still use the `status` and `status_message` helpers. `response.headers["cookie"]` and `headers["cookie"]` will no longer return any CGI cookies. You can inspect `headers["Set-Cookie"]` to see the raw cookie header or use the `cookies` helper to get a hash of the cookies sent to the client.
- `formatted_polymorphic_url` is deprecated. Use `polymorphic_url` with `:format` instead.
- The `:http_only` option in `ActionController::Response#set_cookie` has been renamed to `:httponly`.
- The `:connector` and `:skip_last_comma` options of `to_sentence` have been replaced by `:words_connector`, `:two_words_connector`, and `:last_word_connector` options.
- Posting a multipart form with an empty `file_field` control used to submit an empty string to the controller. Now it submits a nil, due to differences between Rack's multipart parser and the old Rails one.

Credits

Release notes compiled by Mike Gunderloy. This version of the Rails 2.3 release notes was compiled based on RC2 of Rails 2.3.