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

Ruby on Rails 3.1 Release Notes

Highlights in Rails 3.1:

- Streaming
- Reversible Migrations
- Assets Pipeline
- jQuery as the default JavaScript library

These release notes cover only the major changes. To learn about various bug fixes and changes, please refer to the changelogs or check out the list of commits in the main Rails repository on GitHub.

Upgrading to Rails 3.1

If you're upgrading an existing application, it's a great idea to have good test coverage before going in. You should also first upgrade to Rails 3 in case you haven't and make sure your application still runs as expected before attempting to update to Rails 3.1. Then take heed of the following changes:

Rails 3.1 requires at least Ruby 1.8.7

Rails 3.1 requires Ruby 1.8.7 or higher. Support for all of the previous Ruby versions has been dropped officially and you should upgrade as early as possible. Rails 3.1 is also compatible with Ruby 1.9.2.

TIP: Note that Ruby 1.8.7 p248 and p249 have marshalling bugs that crash Rails. Ruby Enterprise Edition have these fixed since release 1.8.7-2010.02 though. On the 1.9 front, Ruby 1.9.1 is not usable because it outright segfaults, so if you want to use 1.9.x jump on 1.9.2 for smooth sailing.

What to update in your apps

The following changes are meant for upgrading your application to Rails 3.1.3, the latest 3.1.x version of Rails.

Gemfile Make the following changes to your `Gemfile`.

```
gem 'rails', '= 3.1.3'
gem 'mysql2'

# Needed for the new asset pipeline
group :assets do
```

```
  gem 'sass-rails',   "~> 3.1.5"
  gem 'coffee-rails', "~> 3.1.1"
  gem 'uglifier',     ">= 1.0.3"
end

# jQuery is the default JavaScript library in Rails 3.1
gem 'jquery-rails'
```

config/application.rb

- The asset pipeline requires the following additions:

  ```
  config.assets.enabled = true
  config.assets.version = '1.0'
  ```

- If your application is using the "/assets" route for a resource you may want change the prefix used for assets to avoid conflicts:

  ```
  # Defaults to '/assets'
  config.assets.prefix = '/asset-files'
  ```

config/environments/development.rb

- Remove the RJS setting `config.action_view.debug_rjs = true`.
- Add the following, if you enable the asset pipeline.

  ```
  # Do not compress assets
  config.assets.compress = false

  # Expands the lines which load the assets
  config.assets.debug = true
  ```

config/environments/production.rb

- Again, most of the changes below are for the asset pipeline. You can read more about these in the Asset Pipeline guide.

  ```
  # Compress JavaScripts and CSS
  config.assets.compress = true

  # Don't fallback to assets pipeline if a precompiled asset is missed
  config.assets.compile = false

  # Generate digests for assets URLs
  config.assets.digest = true

  # Defaults to Rails.root.join("public/assets")
  # config.assets.manifest = YOUR_PATH
  ```

```
# Precompile additional assets (application.js, application.css, and all non-JS/CSS are
# config.assets.precompile `= %w( admin.js admin.css )


# Force all access to the app over SSL, use Strict-Transport-Security, and use secure c
# config.force_ssl = true
```

config/environments/test.rb

```
# Configure static asset server for tests with Cache-Control for performance
config.serve_static_assets = true
config.static_cache_control = "public, max-age=3600"
```

config/initializers/wrap_parameters.rb

- Add this file with the following contents, if you wish to wrap parameters into a nested hash. This is on by default in new applications.

```
# Be sure to restart your server when you modify this file.
# This file contains settings for ActionController::ParamsWrapper which
# is enabled by default.

# Enable parameter wrapping for JSON. You can disable this by setting :format to an emp
ActiveSupport.on_load(:action_controller) do
  wrap_parameters :format => [:json]
end

# Disable root element in JSON by default.
ActiveSupport.on_load(:active_record) do
  self.include_root_in_json = false
end
```

Remove :cache and :concat options in asset helpers references in views

- With the Asset Pipeline the :cache and :concat options aren't used anymore, delete these options from your views.

Creating a Rails 3.1 application

```
# You should have the 'rails' RubyGem installed
$ rails new myapp
$ cd myapp
```

Vendoring Gems

Rails now uses a `Gemfile` in the application root to determine the gems you require for your application to start. This `Gemfile` is processed by the Bundler gem, which then installs all your dependencies. It can even install all the

dependencies locally to your application so that it doesn't depend on the system gems.

More information: - bundler homepage

Living on the Edge

`Bundler` and `Gemfile` makes freezing your Rails application easy as pie with the new dedicated `bundle` command. If you want to bundle straight from the Git repository, you can pass the `--edge` flag:

```
$ rails new myapp --edge
```

If you have a local checkout of the Rails repository and want to generate an application using that, you can pass the `--dev` flag:

```
$ ruby /path/to/rails/railties/bin/rails new myapp --dev
```

Rails Architectural Changes

Assets Pipeline

The major change in Rails 3.1 is the Assets Pipeline. It makes CSS and JavaScript first-class code citizens and enables proper organization, including use in plugins and engines.

The assets pipeline is powered by Sprockets and is covered in the Asset Pipeline guide.

HTTP Streaming

HTTP Streaming is another change that is new in Rails 3.1. This lets the browser download your stylesheets and JavaScript files while the server is still generating the response. This requires Ruby 1.9.2, is opt-in and requires support from the web server as well, but the popular combo of NGINX and Unicorn is ready to take advantage of it.

Default JS library is now jQuery

jQuery is the default JavaScript library that ships with Rails 3.1. But if you use Prototype, it's simple to switch.

```
$ rails new myapp -j prototype
```

Identity Map

Active Record has an Identity Map in Rails 3.1. An identity map keeps previously instantiated records and returns the object associated with the record if accessed again. The identity map is created on a per-request basis and is flushed at request completion.

Rails 3.1 comes with the identity map turned off by default.

Railties

- jQuery is the new default JavaScript library.
- jQuery and Prototype are no longer vendored and is provided from now on by the `jquery-rails` and `prototype-rails` gems.
- The application generator accepts an option `-j` which can be an arbitrary string. If passed "foo", the gem "foo-rails" is added to the `Gemfile`, and the application JavaScript manifest requires "foo" and "foo_ujs". Currently only "prototype-rails" and "jquery-rails" exist and provide those files via the asset pipeline.
- Generating an application or a plugin runs `bundle install` unless `--skip-gemfile` or `--skip-bundle` is specified.
- The controller and resource generators will now automatically produce asset stubs (this can be turned off with `--skip-assets`). These stubs will use CoffeeScript and Sass, if those libraries are available.
- Scaffold and app generators use the Ruby 1.9 style hash when running on Ruby 1.9. To generate old style hash, `--old-style-hash` can be passed.
- Scaffold controller generator creates format block for JSON instead of XML.
- Active Record logging is directed to STDOUT and shown inline in the console.
- Added `config.force_ssl` configuration which loads `Rack::SSL` middleware and force all requests to be under HTTPS protocol.
- Added `rails plugin new` command which generates a Rails plugin with gemspec, tests and a dummy application for testing.
- Added `Rack::Etag` and `Rack::ConditionalGet` to the default middleware stack.
- Added `Rack::Cache` to the default middleware stack.
- Engines received a major update - You can mount them at any path, enable assets, run generators, etc.

Action Pack

Action Controller

- A warning is given out if the CSRF token authenticity cannot be verified.

- Specify `force_ssl` in a controller to force the browser to transfer data via HTTPS protocol on that particular controller. To limit to specific actions, `:only` or `:except` can be used.
- Sensitive query string parameters specified in `config.filter_parameters` will now be filtered out from the request paths in the log.
- URL parameters which return `nil` for `to_param` are now removed from the query string.
- Added `ActionController::ParamsWrapper` to wrap parameters into a nested hash, and will be turned on for JSON request in new applications by default. This can be customized in `config/initializers/wrap_parameters.rb`.
- Added `config.action_controller.include_all_helpers`. By default `helper :all` is done in `ActionController::Base`, which includes all the helpers by default. Setting `include_all_helpers` to `false` will result in including only application_helper and the helper corresponding to controller (like foo_helper for foo_controller).
- `url_for` and named URL helpers now accept `:subdomain` and `:domain` as options.
- Added `Base.http_basic_authenticate_with` to do simple http basic authentication with a single class method call.

```
class PostsController < ApplicationController
  USER_NAME, PASSWORD = "dhh", "secret"

  before_filter :authenticate, :except => [ :index ]

  def index
    render :text => "Everyone can see me!"
  end

  def edit
    render :text => "I'm only accessible if you know the password"
  end

  private
    def authenticate
      authenticate_or_request_with_http_basic do |user_name, password|
        user_name == USER_NAME && password == PASSWORD
      end
    end
end
```

..can now be written as

```
class PostsController < ApplicationController
```

```
  http_basic_authenticate_with :name => "dhh", :password => "secret", :except => :inde

  def index
    render :text => "Everyone can see me!"
  end

  def edit
    render :text => "I'm only accessible if you know the password"
  end
end
```

- Added streaming support, you can enable it with:

```
class PostsController < ActionController::Base
  stream
end
```

  You can restrict it to some actions by using `:only` or `:except`. Please read the docs at `ActionController::Streaming` for more information.

- The redirect route method now also accepts a hash of options which will only change the parts of the URL in question, or an object which responds to call, allowing for redirects to be reused.

Action Dispatch

- `config.action_dispatch.x_sendfile_header` now defaults to `nil` and `config/environments/production.rb` doesn't set any particular value for it. This allows servers to set it through `X-Sendfile-Type`.
- `ActionDispatch::MiddlewareStack` now uses composition over inheritance and is no longer an array.
- Added `ActionDispatch::Request.ignore_accept_header` to ignore accept headers.
- Added `Rack::Cache` to the default stack.
- Moved etag responsibility from `ActionDispatch::Response` to the middleware stack.
- Rely on `Rack::Session` stores API for more compatibility across the Ruby world. This is backwards incompatible since `Rack::Session` expects `#get_session` to accept four arguments and requires `#destroy_session` instead of simply `#destroy`.
- Template lookup now searches further up in the inheritance chain.

Action View

- Added an `:authenticity_token` option to `form_tag` for custom handling or to omit the token by passing `:authenticity_token => false`.
- Created `ActionView::Renderer` and specified an API for `ActionView::Context`.
- In place `SafeBuffer` mutation is prohibited in Rails 3.1.
- Added HTML5 `button_tag` helper.
- `file_field` automatically adds `:multipart => true` to the enclosing form.
- Added a convenience idiom to generate HTML5 data-* attributes in tag helpers from a `:data` hash of options:

```
tag("div", :data => {:name => 'Stephen', :city_state => %w(Chicago IL)})
# => <div data-name="Stephen" data-city-state="[&quot;Chicago&quot;,&quot;IL&quot;]" />
```

Keys are dasherized. Values are JSON-encoded, except for strings and symbols.

- `csrf_meta_tag` is renamed to `csrf_meta_tags` and aliases `csrf_meta_tag` for backwards compatibility.
- The old template handler API is deprecated and the new API simply requires a template handler to respond to call.
- rhtml and rxml are finally removed as template handlers.
- `config.action_view.cache_template_loading` is brought back which allows to decide whether templates should be cached or not.
- The submit form helper does not generate an id "object_name_id" anymore.
- Allows `FormHelper#form_for` to specify the `:method` as a direct option instead of through the `:html` hash. `form_for(@post, remote: true, method: :delete)` instead of `form_for(@post, remote: true, html: { method: :delete })`.
- Provided `JavaScriptHelper#j()` as an alias for `JavaScriptHelper#escape_javascript()`. This supersedes the `Object#j()` method that the JSON gem adds within templates using the JavaScriptHelper.
- Allows AM/PM format in datetime selectors.
- `auto_link` has been removed from Rails and extracted into the rails_autolink gem

Active Record

- Added a class method `pluralize_table_names` to singularize/pluralize table names of individual models. Previously this could only be set globally

for all models through `ActiveRecord::Base.pluralize_table_names`.

```
class User < ActiveRecord::Base
  self.pluralize_table_names = false
end
```

- Added block setting of attributes to singular associations. The block will get called after the instance is initialized.

```
class User < ActiveRecord::Base
  has_one :account
end

user.build_account{ |a| a.credit_limit = 100.0 }
```

- Added `ActiveRecord::Base.attribute_names` to return a list of attribute names. This will return an empty array if the model is abstract or the table does not exist.
- CSV Fixtures are deprecated and support will be removed in Rails 3.2.0.
- `ActiveRecord#new`, `ActiveRecord#create` and `ActiveRecord#update_attributes` all accept a second hash as an option that allows you to specify which role to consider when assigning attributes. This is built on top of Active Model's new mass assignment capabilities:

```
class Post < ActiveRecord::Base
  attr_accessible :title
  attr_accessible :title, :published_at, :as => :admin
end

Post.new(params[:post], :as => :admin)
```

- `default_scope` can now take a block, lambda, or any other object which responds to call for lazy evaluation.
- Default scopes are now evaluated at the latest possible moment, to avoid problems where scopes would be created which would implicitly contain the default scope, which would then be impossible to get rid of via Model.unscoped.
- PostgreSQL adapter only supports PostgreSQL version 8.2 and higher.
- `ConnectionManagement` middleware is changed to clean up the connection pool after the rack body has been flushed.
- Added an `update_column` method on Active Record. This new method updates a given attribute on an object, skipping validations and callbacks. It is recommended to use `update_attributes` or `update_attribute` unless you are sure you do not want to execute any callback, including the modification of the `updated_at` column. It should not be called on new records.

- Associations with a `:through` option can now use any association as the through or source association, including other associations which have a `:through` option and `has_and_belongs_to_many` associations.
- The configuration for the current database connection is now accessible via `ActiveRecord::Base.connection_config`.
- limits and offsets are removed from COUNT queries unless both are supplied.

```
People.limit(1).count           # => 'SELECT COUNT(*) FROM people'
People.offset(1).count          # => 'SELECT COUNT(*) FROM people'
People.limit(1).offset(1).count # => 'SELECT COUNT(*) FROM people LIMIT 1 OFFSET 1'
```

- `ActiveRecord::Associations::AssociationProxy` has been split. There is now an `Association` class (and subclasses) which are responsible for operating on associations, and then a separate, thin wrapper called `CollectionProxy`, which proxies collection associations. This prevents namespace pollution, separates concerns, and will allow further refactorings.
- Singular associations (`has_one`, `belongs_to`) no longer have a proxy and simply returns the associated record or `nil`. This means that you should not use undocumented methods such as `bob.mother.create` - use `bob.create_mother` instead.
- Support the `:dependent` option on `has_many :through` associations. For historical and practical reasons, `:delete_all` is the default deletion strategy employed by `association.delete(*records)`, despite the fact that the default strategy is `:nullify` for regular has_many. Also, this only works at all if the source reflection is a belongs_to. For other situations, you should directly modify the through association.
- The behavior of `association.destroy` for `has_and_belongs_to_many` and `has_many :through` is changed. From now on, 'destroy' or 'delete' on an association will be taken to mean 'get rid of the link', not (necessarily) 'get rid of the associated records'.
- Previously, `has_and_belongs_to_many.destroy(*records)` would destroy the records themselves. It would not delete any records in the join table. Now, it deletes the records in the join table.
- Previously, `has_many_through.destroy(*records)` would destroy the records themselves, and the records in the join table. [Note: This has not always been the case; previous version of Rails only deleted the records themselves.] Now, it destroys only the records in the join table.
- Note that this change is backwards-incompatible to an extent, but there is unfortunately no way to 'deprecate' it before changing it. The change is being made in order to have consistency as to the meaning of 'destroy' or

'delete' across the different types of associations. If you wish to destroy the records themselves, you can do `records.association.each(&:destroy)`.

- Add `:bulk => true` option to `change_table` to make all the schema changes defined in a block using a single ALTER statement.

```
change_table(:users, :bulk => true) do |t|
  t.string :company_name
  t.change :birthdate, :datetime
end
```

- Removed support for accessing attributes on a `has_and_belongs_to_many` join table. `has_many :through` needs to be used.
- Added a `create_association!` method for `has_one` and `belongs_to` associations.
- Migrations are now reversible, meaning that Rails will figure out how to reverse your migrations. To use reversible migrations, just define the `change` method.

```
class MyMigration < ActiveRecord::Migration
  def change
    create_table(:horses) do |t|
      t.column :content, :text
      t.column :remind_at, :datetime
    end
  end
end
```

- Some things cannot be automatically reversed for you. If you know how to reverse those things, you should define `up` and `down` in your migration. If you define something in change that cannot be reversed, an `IrreversibleMigration` exception will be raised when going down.
- Migrations now use instance methods rather than class methods:

```
class FooMigration < ActiveRecord::Migration
  def up # Not self.up
    # ...
  end
end
```

- Migration files generated from model and constructive migration generators (for example, add_name_to_users) use the reversible migration's `change` method instead of the ordinary `up` and `down` methods.
- Removed support for interpolating string SQL conditions on associations. Instead, a proc should be used.

```
has_many :things, :conditions => 'foo = #{bar}'          # before
has_many :things, :conditions => proc { "foo = #{bar}" } # after
```

Inside the proc, `self` is the object which is the owner of the association, unless you are eager loading the association, in which case `self` is the class which the association is within.

You can have any "normal" conditions inside the proc, so the following will work too:

```
has_many :things, :conditions => proc { ["foo = ?", bar] }
```

- Previously `:insert_sql` and `:delete_sql` on `has_and_belongs_to_many` association allowed you to call 'record' to get the record being inserted or deleted. This is now passed as an argument to the proc.
- Added `ActiveRecord::Base#has_secure_password` (via `ActiveModel::SecurePassword`) to encapsulate dead-simple password usage with BCrypt encryption and salting.

```
# Schema: User(name:string, password_digest:string, password_salt:string)
class User < ActiveRecord::Base
  has_secure_password
end
```

- When a model is generated `add_index` is added by default for `belongs_to` or `references` columns.
- Setting the id of a `belongs_to` object will update the reference to the object.
- `ActiveRecord::Base#dup` and `ActiveRecord::Base#clone` semantics have changed to closer match normal Ruby dup and clone semantics.
- Calling `ActiveRecord::Base#clone` will result in a shallow copy of the record, including copying the frozen state. No callbacks will be called.
- Calling `ActiveRecord::Base#dup` will duplicate the record, including calling after initialize hooks. Frozen state will not be copied, and all associations will be cleared. A duped record will return `true` for `new_record?`, have a `nil` id field, and is saveable.
- The query cache now works with prepared statements. No changes in the applications are required.

Active Model

- `attr_accessible` accepts an option `:as` to specify a role.
- `InclusionValidator`, `ExclusionValidator`, and `FormatValidator` now accepts an option which can be a proc, a lambda, or anything that respond to `call`. This option will be called with the current record as an argument and returns an object which respond to `include?` for `InclusionValidator` and `ExclusionValidator`, and returns a regular expression object for `FormatValidator`.

- Added `ActiveModel::SecurePassword` to encapsulate dead-simple password usage with BCrypt encryption and salting.
- `ActiveModel::AttributeMethods` allows attributes to be defined on demand.
- Added support for selectively enabling and disabling observers.
- Alternate `I18n` namespace lookup is no longer supported.

Active Resource

- The default format has been changed to JSON for all requests. If you want to continue to use XML you will need to set `self.format = :xml` in the class. For example,

```
class User < ActiveResource::Base
  self.format = :xml
end
```

Active Support

- `ActiveSupport::Dependencies` now raises `NameError` if it finds an existing constant in `load_missing_constant`.
- Added a new reporting method `Kernel#quietly` which silences both `STDOUT` and `STDERR`.
- Added `String#inquiry` as a convenience method for turning a String into a `StringInquirer` object.
- Added `Object#in?` to test if an object is included in another object.
- `LocalCache` strategy is now a real middleware class and no longer an anonymous class.
- `ActiveSupport::Dependencies::ClassCache` class has been introduced for holding references to reloadable classes.
- `ActiveSupport::Dependencies::Reference` has been refactored to take direct advantage of the new `ClassCache`.
- Backports `Range#cover?` as an alias for `Range#include?` in Ruby 1.8.
- Added `weeks_ago` and `prev_week` to Date/DateTime/Time.
- Added `before_remove_const` callback to `ActiveSupport::Dependencies.remove_unloadable_constar`

Deprecations:

- `ActiveSupport::SecureRandom` is deprecated in favor of `SecureRandom` from the Ruby standard library.

Credits

See the full list of contributors to Rails for the many people who spent many hours making Rails, the stable and robust framework it is. Kudos to all of them.

Rails 3.1 Release Notes were compiled by Vijay Dev