RUBY ON RAILS 2.1

¿QUE HAY DE NUEVO?

CARLOS BRANDO

revisión: MARCOS TAPAJÓS - tapa: DANIEL LOPES - traducción a cargo de: GASTÓN RAMOS y LUCAS FLORIO

Ruby on Rails 2.1

¿QUÉ HAY DE NUEVO?

Primer Edición

Ruby on Rails 2.1

¿QUÉ HAY DE NUEVO?

Primer Edición

Carlos Brando
Marcos Tapajós

Primer Edición: Junio 2008

Carlos Brando
Website: www.nomedojogo.com

Marcos Tapajós
Website: www.improveit.com.br/en/company/tapajos

Chapter 1

Introducción

Alrededor de julio de 2004 David Heinemeier Hansson lanzó la versión pública del framework Ruby on Rails, el cual surgió como consecuencia del desarrollo de una aplicación llamada Basecamp. Después de tres años, el 7 de Diciembre de 2007 se publicó Ruby on Rails 2.0 con una cantidad importante de cambios.

Seis meses después, y durante todo este tiempo han contribuido más de **1400 desarrolladores** alrededor de todo el mundo con **1600 parches** para el framework. Hoy, 1ero. Junio de 2008, se ha publicado la versión 2.1 del framework Ruby on Rails.

Según David las principales novedades de esta versión son:

- Timezones
- Dirty tracking
- Gem Dependencies
- Named scope

- UTC-based migrations
- Better caching

Como siempre, para actualizar o instalar la nueva versión:

```
gem install rails
```

AGRADECIMIENTOS

A Marcos Tapajós, co-autor de este libro. De no ser por él, no podrías estar leyendo este libro.

A Daniel Lopes quien hizo la hermosa tapa de esta edición.

A toda la comunidad Brazilera de Ruby on Rails que nos ayudó directa o indirectamente con este libro, comentando los artículos del blog o aconsejándonos. Como me gusta decir siempre, lo mejor de Rails es su comunidad! se mantiene creando, inventando, y especialmente compartiendo.

TRADUCTORES

Este libro fue orgullosamente traducido al español por estos chicos Argentinos:

Lucas Florio - http://blog.lucasefe.com.ar/

Capítulos 8-12

Gastón Ramos - http://gastonramos.wordpress.com

Introducción y capítulos 1-7, 13-14

Chapter 2

ActiveRecord

ActiveRecord es una capa de mapeo objeto-relacional (object-relational mapping layer) responsable tanto de la interoperabiliadad entre la aplicación y la base de datos como de la abstracción de los datos. (wikipedia)

EL MÉTODO SUM

Expresiones en el método sum

Ahora podemos usar expresiones en los métodos de cálculo de **ActiveRecord**, **sum**, por ejemplo:

```
Person.sum("2 * age")
```

Cambiando el valor de retorno por defecto del método sum

En las versiones previas, cuando usábamos el método **sum** de **ActiveRecord** para calcular la suma de una determinada columna para todos los registros de una tabla, si ningún registro correspondía con las condiciones expresadas en el método de invocación, entonces el valor de retorno por defecto era **nil**.

En Rails 2.1 el valor de retorno por defecto (cuando no se encuentra ningún registro) es 0. Vea el ejemplo:

```
Account.sum(:balance, :conditions => '1 = 2') #=> 0
```

HAS_ONE

Soporte para la opción through

El método **has_one** ahora tiene la opción **through**. Esta funciona como **has_many :through**, pero en este caso representa la asociación a un objeto simple **ActiveRecord**. Ejemplo:

```
class Magazine < ActiveRecord::Base
  has_many :subscriptions
end

class Subscription < ActiveRecord::Base
  belongs_to :magazine
  belongs_to :user
end

class User < ActiveRecord::Base
  has_many :subscriptions
  has_one :magazine, :through => : subscriptions,
```

```
                    :conditions => ['subscriptions.active = ?', true]
end
```

Has_one con :source_type

El método **has_one :through** que acabamos de ver, puede también tomar **:source_type** como argumento. Voy a intentar explicar esto a través de algunos ejemplos. Vamos comenzar con estas dos clases:

```
class Client < ActiveRecord::Base
  has_many :contact_cards

  has_many :contacts, :through => :contact_cards
end
```

Lo que estamos viendo hasta aquí es una clase **Client** la cual tiene muchos (**has_many**) tipos de contactos (**contacts**), ya que la clase **ContactCard** tiene una relación polimórfica.

Para seguir con nuestro ejemplo, vamos a crear dos clases que van a representar a **ContactCard**:

```
class Person < ActiveRecord::Base
  has_many :contact_cards, :as => :contact
end

class Business < ActiveRecord::Base
  has_many :contact_cards, :as => :contact
end
```

Person y **Business** están relacionadas con mi clase **Client** a través de la tabla **ContactCard**. En otras palabras, tengo dos tipos de contactos, personal y de negocios.

Por otro lado, cuando intente recuperar un contacto esto no va a funcionar, veamos:

```
>> Client.find(:first).contacts
# ArgumentError: /.../active_support/core_ext/hash/keys.rb:48:
# in `assert_valid_keys': Unknown key(s): polymorphic
```

Para que esto funcione tenemos que usar **:source_type**. Vamos cambiar nuestra clase **Client**:

```
class Client < ActiveRecord::Base
  has_many :people_contacts,
           :through => :contact_cards,
           :source => :contacts,
           :source_type => :person

  has_many :business_contacts,
           :through => :contact_cards,
           :source => :contacts,
           :source_type => :business
end
```

Note que ahora tenemos dos maneras diferentes de recuperar nuestros contactos y podemos indicar que tipo de contacto **:source_type** esperamos.

```
Client.find(:first).people_contacts
Client.find(:first).business_contacts
```

NAMED_SCOPE

La gema *has_finder* ha sido incluida en Rails pero con un nombre diferente: **named_scope**.

Para entender lo que significa este agregado en Rails vamos a ver los siguientes ejemplos:

```
class Article < ActiveRecord::Base
  named_scope :published, :conditions => {:published => true}
  named_scope :containing_the_letter_a, :conditions => "body LIKE '%a%â€™"
end

Article.published.paginate(:page => 1)
Article.published.containing_the_letter_a.count
Article.containing_the_letter_a.find(:first)
Article.containing_the_letter_a.find(:all, :conditions => {â€¦})
```

En vez de crear un método nuevo llamado **published** para retornar todos los posts publicados, estoy usando un **named_scope** para que lo haga por mí. Pero puedo ir más allá que esto. Vamos a ver otro ejemplo para ver como se puede usar esto:

```
named_scope :written_before, lambda { |time|
  { :conditions => ['written_on < ?', time] }
}

named_scope :anonymous_extension do
  def one
    1
  end
end

named_scope :named_extension, :extend => NamedExtension

named_scope :multiple_extensions,
        :extend => [MultipleExtensionTwo, MultipleExtensionOne]
```

TESTEANDO NAMED_SCOPE CON PROXY_OPTIONS

Named scopes es una nueva e interesante caraterística de Rails 2.1, pero después de usarlo durante un tiempo quizás sea díficil testear estas estructuras más complejas.

Veamos un ejemplo:

```
class Shirt < ActiveRecord::Base
  named_scope :colored, lambda { |color|
    { :conditions => { :color => color } }
  }
end
```

¿Cómo creamos un test que valide la generación de este scope ?

Para solucionar esto, se creó el método **proxy_options**. El cual nos permite examinar las opciones usadas en **named_scope**. Para testear el código de arriba podríamos escribir esto:

```
class ShirtTest < Test::Unit
  def test_colored_scope
    red_scope = { :conditions => { :colored => 'red' } }
    blue_scope = { :conditions => { :colored => 'blue' } }
    assert_equal red_scope, Shirt.colored('red').scope_options
    assert_equal blue_scope, Shirt.colored('blue').scope_options
  end
end
```

INCREMENTO Y DECREMENTO

Los métodos de **ActiveRecord increment**, **increment!**, **decrement** y **decrement!** ahora toman un nuevo parámetro opcional. En las versiones previas de Rails se podían usar estos métodos para restar 1 a una columna dada. En rails 2.1 podemos decir que valor restar o sumar. Así:

```
player1.increment!(:points, 5)
player2.decrement!(:points, 2)
```

En el ejemplo estoy sumando 5 puntos al jugador 1 y restando 2 puntos al jugador 2. Cómo este es un parámetro opcional, el código viejo no se ve afectado.

FIND

Conditions

A partir de ahora, se va a poder pasar un objeto como parámetro al método **find** de **ActiveRecord**, Vea este caso como ejemplo:

```
class Account < ActiveRecord::Base
  composed_of :balance, :class_name => "Money", :mapping => %w(balance amount)
end
```

En este caso, ud. puede pasar una instancia de **Money** cómo parámetro al método **find** de la clase **Account**, de la siguiente manera:

```
amount = 500
currency = "USD"
Account.find(:all, :conditions => { :balance => Money.new(amount, currency) })
```

Last

Hasta ahora sólo podíamos usar tres operadores para buscar datos usando el método **find** de **ActiveRecord**: **:first**, **:all** o el propio id del objeto (en este caso no pasamos ningún operador específico más que el id en sí mismo)

En Rails 2.1 hay un cuarto operador llamado **:last**. Vea algunos ejemplos:

```
Person.find(:last)
Person.find(:last, :conditions => [ "user_name = ?", user_name])
Person.find(:last, :order => "created_on DESC", :offset => 5)
```

Para entender como funciona este nuevo operador, veamos el siguiente test:

```
def test_find_last
  last  = Developer.find :last
  assert_equal last, Developer.find(:first, :order => 'id desc')
end
```

All

El método estático **all** es un alias al método estático **find(:all)**. Por ejemplo:

```
Topic.all es lo mismo que Topic.find(:all)
```

First

El método estático **first** es un alias al método estático **find(:first)**. Por ejemplo:

```
Topic.first es lo mismo que Topic.find(:first)
```

Last

El método estático **last** es un alias al método estático **find(:last)**. Por ejemplo:

```
Topic.last es lo mismo que Topic.find(:last)
```

USANDO FIRST Y LAST CON NAMED_SCOPE

Todo los métodos mencionados anteriormente funcionan con **named_scope**. Supongamos que creamos un **named_scope** llamado **recent**. Entonces podríamos hacer esto:

```
post.comments.recent.last
```

EAGER LOADING

Para explicar esta nueva funcionalidad, veamos el siguiente código:

```
Author.find(:all, :include => [:posts, :comments])
```

Estoy haciendo una búsqueda en la tabla **authors** y también incluyendo las tablas **posts** y **comments** en mi consulta a través de la columna **author_id**, el cual es el nombre de columna por defecto de acuerdo a la conveción que rails utiliza para las claves foráneas. Esta búsqueda genera una consulta como la siguiente:

```
SELECT
  authors."id"          AS t0_r0,
  authors."created_at"  AS t0_r1,
  authors."updated_at"  AS t0_r2,
  posts."id"            AS t1_r0,
  posts."author_id"     AS t1_r1,
  posts."created_at"    AS t1_r2,
  posts."updated_at"    AS t1_r3,
  comments."id"         AS t2_r0,
  comments."author_id"  AS t2_r1,
  comments."created_at" AS t2_r2,
  comments."updated_at" AS t2_r3
FROM
  authors
  LEFT OUTER JOIN posts ON posts.author_id = authors.id
  LEFT OUTER JOIN comments ON comments.author_id = authors.id
```

Una única consulta SQL con **joins** entre las tablas **authors**, **posts** y **comments**. A esto lo llamamos **producto cartesiano**.

Este tipo de consultas no siempre tiene una buena performance, entonces esto se cambió en Rails 2.1. La misma consulta para la clase **Author** ahora se hace de una forma diferente para traer la información de las tres tablas. En vez de usar una única consulta SQL con las tres tablas, Rails ahora usa tres consultas diferentes - una por cada tabla - las cuales son más cortas que la anterior. El resultado se puede ver en los logs despues de ejecutar el codigo ruby on rails previo:

```
SELECT * FROM "authors"
SELECT posts.* FROM "posts" WHERE (posts.author_id IN (1))
SELECT comments.* FROM "comments" WHERE (comments.author_id IN (1))
```

En la **mayoría de los casos** estas tres simples consultas se ejecutan más rápido que la consulta larga y compleja.

BELONGS_TO

El método **belongs_to** se cambió en pro de permitir el uso de **:dependent => :destroy** y **:delete** en las asociaciones.

Por ejemplo:

```
belongs_to :author_address
belongs_to :author_address, :dependent => :destroy
belongs_to :author_address_extra, :dependent => :delete,
        :class_name => "AuthorAddress"
```

POLYMORPHIC URL

Los métodos helper para las URL polimórficas son usados para resolver de una forma más elegante una ruta nombrada cuando cuando tenemos un modelo de **ActiveRecord**.

Estos métodos son útiles cuando queremos generar la URL para un recurso **RESTful** sin especificar el tipo de registro en cuestión.

Es muy fácil de usar. Veamos algunos ejemplos (los comentarios explican las llamdas equivalentes para las versiones anteriores a Rails 2.1)

```
record = Article.find(:first)
polymorphic_url(record) #-> article_url(record)

record = Comment.find(:first)
polymorphic_url(record)  #->  comment_url(record)

# it can also identify recently created elements
record = Comment.new
polymorphic_url(record)  #->  comments_url()
```

Note como el método **polymorphic_url** es capáz de indetificar el tipo del objeto dado y generar la ruta correcta. También soporta **Recursos anidados** y **espacios de nombres**:

```
polymorphic_url([:admin, @article, @comment])
#-> this will return:
admin_article_comment_url(@article, @comment)
```

También se puede usar **new**, **edit** y **formatted** como prefijos:

```
edit_polymorphic_path(@post)
#=> /posts/1/edit

formatted_polymorphic_path([@post, :pdf])
#=> /posts/1.pdf
```

RELACIONES DE SÓLO LECTURA

Se agregó una nueva caraterística a las relaciones entre modelos. Con el fin de impedir alterar los modelos ahora podemos usar **:readonly** para asociar modelos. Veamos algunos ejemplos:

```
has_many :reports, :readonly => true

has_one :boss, :readonly => :true

belongs_to :project, :readonly => true

has_and_belongs_to_many :categories, :readonly => true
```

De esta manera los registros protegidos no podrán ser alterados desde este modelo. Si intentamos editar algunos de los registros de este modelos entonces obtendríamos la siguiente excepción: **ActiveRecord::ReadOnlyRecord**

MÉTODOS ADD_TIMESTAMPS Y REMOVE_TIMESTAMPS

Ahora tenemos 2 métodos nuevos: **add_timestamps** y **remove_timestamps**. Los cuales agregan y eliminan, respectivamente, las columnas de **timestamp**. Veamos un ejemplo:

```
def self.up
  add_timestamps :feeds
  add_timestamps :urls
end

def self.down
  remove_timestamps :urls
```

```
  remove_timestamps :feeds
end
```

CÁLCULOS

ActiveRecord::Calculations ha cambiado un poco para aceptar no sólo el nombre de la columna sino también el nombre de la tabla. Esto es muy útil cuando tenemos una relación entre dos tablas que tienen una columna con el mismo nombre. Los métodos afectados son: **sum** o **maximum** de **ActiveRecord** (entre otros). Para resumir ahora podemos hacer esto de estas dos formas:

```
authors.categories.maximum(:id)
authors.categories.maximum("categories.id")
```

ACTIVERECORD::BASE.CREATE ACEPTA BLOQUES

Estamos acostumbrados al método **ActiveRecord::Base.new** que acepta el uso de bloques. Ahora podemos hacer lo mismo con el método **create**:

```
# Creación de un nuevo objeto pasándole un bloque con la descripción de sus atributos
User.create(:first_name => 'Jamie') do |u|
  u.is_admin = false
end
```

Podemos usar este mismo método para crear muchos objetos de una sola vez:

```
# Creación de un array de objetos nuevos usando un bloque.
# El bloque se ejecuta una vez por cada nuevo objeto creado.
User.create([{:first_name => 'Jamie'}, {:first_name => 'Jeremy'}]) do |u|
```

```ruby
  u.is_admin = false
end
```

Esto también funciona para las asociaciones:

```ruby
author.posts.create!(:title => "New on Edge") {|p| p.body = "More cool stuff!"}

# o

author.posts.create!(:title => "New on Edge") do |p|
  p.body = "More cool stuff!"
end
```

CHANGE_TABLE

La creación de migraciones (**migrations**) en Rails 2.0 se hizo mucho más sexy que en las versiones previas, pero alterar una tabla usando **migrations** continuaba siendo como antes, nada sexy.

En Rails 2.1, ahora también es sexy alterar una tabla, con el nuevo método **change_table**. Veamos el ejemplo:

```ruby
change_table :videos do |t|
  t.timestamps # esto agrega las columnas created_at y updated_at
  t.belongs_to :goat # esto agrega la columna goat_id (integer)
  t.string :name, :email, :limit => 20 # esto agrega las columnas name y email
  t.remove :name, :email # esto elimina las columnas name y email
end
```

El nuevo método **change_table** funciona como **create_table** pero en vez de crear un tabla modifica una ya existente agregando o eliminando columnas e índices.

```ruby
change_table :table do |t|
  t.column # agrega una columna normal. Ej: t.column(:name, :string)
  t.index # agrega un nuevo índice.
  t.timestamps
  t.change # cambia la definición de una columna. Ej: t.change(:name, :string, :limit => 80)
  t.change_default # cambia el valor por defecto de una columna.
  t.rename # cambia el nombre de una columna.
  t.references
  t.belongs_to
  t.string
  t.text
  t.integer
  t.float
  t.decimal
  t.datetime
  t.timestamp
  t.time
  t.date
  t.binary
  t.boolean
  t.remove
  t.remove_references
  t.remove_belongs_to
  t.remove_index
  t.remove_timestamps
end
```

DIRTY OBJECTS

Ahora en Rails podemos rastrear los cambios hechos por **ActiveRecord**. Se puede saber si un objeto ha sido modificado o no. En caso de que haya cambiado, podemos ver que fue lo que cambió exactamente como así también comparar su estado anterior con el actual. Veamos algunos ejemplos:

```
article = Article.find(:first)
article.changed?  #=> false

article.title  #=> "Title"
article.title = "New Title"
article.title_changed? #=> true

# muestra title antes del cambio
article.title_was  #=> "Title"

# antes y despues del cambio
article.title_change  #=> ["Title", "New Title"]
```

Como ud. puede ver es muy simple. Puede listar todos los cambios hechos al objeto de una esta dos formas:

```
# retorna una lista con todos los atributos que cambiaron
article.changed  #=> ['title']

# retorna un hash con los atributos que cambiaron
# con sus valores antes y después
article.changes  #=> { 'title' => ["Title", "New Title"] }
```

Note que cuando el objeto se guarda su estado de alterado cambia:

```
article.changed?  #=> true
article.save  #=> true
article.changed?  #=> false
```

En caso de ud. quiera cambiar el estado de un objeto sin usar **attr=**, va a necesitar informar explícitamente que se ha modificado el atributo mediante el método **attr_name_will_change!** (reemplace **attr** con el nombre real del atributo del objeto). Veamos un ejemplo más:

```
article = Article.find(:first)
article.title_will_change!
article.title.upcase!
article.title_change  #=> ['Title', 'TITLE']
```

PARTIAL UPDATES

La implementación de **Dirty Objects** dió punto de partida a otra característica muy interesante.

Dado que ahora podemos rastrear los cambios de los atributos de un objeto, ¿por qué no usar esto para evitar actualizaciones innecesarias en la base de datos?

En las versiones previas de Rails cuando ejecutábamos el método **save** de un objeto **ActiveRecord**, se actualizaban todos los campos en la base de datos. Por más que los mismos no hayan sufrido ningún cambio.

Con **Dirty Objects** esta acción podría mejorarse, y esto es exactamente lo que pasó. Miremos un poco la consulta SQL generada en Rails 2.1 cuando intentamos guardar un objeto que a sufrido un pequeño cambio:

```
article = Article.find(:first)
article.title  #=> "Title"
article.subject  #=> "Edge Rails"
```

```
# Cambiamos el título (title)
article.title = "New Title"

# esto crea el siguiente SQL para persistir el objeto
article.save
#=>  "UPDATE articles SET title = 'New Title' WHERE id = 1"
```

Note como sólo se actualizaron en la base de datos los campos que fueron cambiados en la aplicación. Si ningún campo ha sido actualizado en la aplicación, entonces **ActiveRecord** no ejecutará ninguna actualización.

Para habilitar/deshabilitar esta nueva característica debe cambiar la propiedad **partial_updates** de sus modelos.

```
# Para habilitarla
MyClass.partial_updates = true
```

Si desea habilitar/deshabilitar esta característica para todos sus modelos debe editar el archivo *config/initializers/new_rails_defaults.rb*:

```
# Habilitando para todos los modelos
ActiveRecord::Base.partial_updates = true
```

No olvide informarle a Rails a través *config/initializers/new_rails_defaults.rb* que va a actualizar un atributo sin usar el método **attr=**, así:

```
# Si usa **attr=**,
# entocnes está bien no informar nada.
person.name = 'bobby'
person.name_change    # => ['bob', 'bobby']


# Pero debe informar que el campo se va actualizar
```

```
# si no va a usar **attr=**
person.name_will_change!
person.name << 'by'
person.name_change    # => ['bob', 'bobby']
```

Si ud. no informa cambios como estos, no se podrán rastrear los cambios de los atributos del modelo y la tabla no se actualizará correctamente.

¿SMALLINT, INT O BIGINT EN MYSQL?

El adapatador para **MySQL** de **ActiveRecord** es un poco más inteligente en el uso tipos de datos enteros a la hora de crear o alterar columnas en la base de datos. Con la opción **:limit**, podemos definir si la columna va a ser **smallint**, **int** or **bigint**. Veamos el código que hace este trabajo:

```
case limit
when 0..3
  "smallint(#{limit})"
when 4..8
  "int(#{limit})"
when 9..20
  "bigint(#{limit})"
else
  'int(11)'
end
```

Para dejar esto más claro vamos a mapear esto en un archivo de migración y ver que tipo de dato se usa para cada columna:

```
create_table :table_name, :force => true do |t|
```

```
      # 0 - 3: smallint
      t.integer :column_one, :limit => 2 # smallint(2)

      # 4 - 8: int
      t.integer :column_two, :limit => 6 # int(6)

      # 9 - 20: bigint
      t.integer :column_three, :limit => 15 # bigint(15)

      # si no usamos :limit int(11)
      t.integer :column_four # int(11)
    end
```

El adaptador de **PostgreSQL** tenía esta caraterística y **MySQL** sólo siguió esta tendencia.

OPCIONAL :SELECT EN HAS_ONE Y BELONGS_TO

Los ya conocidos métodos **has_one** y **belongs_to** ahora tienen la opción: **:select**.

El valor por defecto es "*" *(como en "SELECT* FROM table"), pero se puede editar para recuperar sólo las columnas que vamos a usar.

No olvide incluir las **claves primarias** y las **claves foráneas**, de lo contrario obtendrá un error.

El método **belongs_to** ya no tiene la opción **:order**. Pero no se preocupe por que realmente no era muy útil.

ALMACENANDO EL NOMBRE COMPLETO DE UNA CLASE CUANDO USAMOS STI

Siempre que usemos **models** con **namespace** y **STI**, **ActiveRecord** sólo almacena el nombre de la clase, sin el nombre del **namespace** (*demodulized*). Esto sólo funcionará si todas las clases en el **STI** estan en el mismo **namespace**. Veamos un ejemplo:

```
class CollectionItem < ActiveRecord::Base; end
class ComicCollection::Item < CollectionItem; end

item = ComicCollection::Item.new
item.type # => 'Item'

item2 = CollectionItem.find(item.id)
# retorna un error, por que no puede encontrar
# la clase Item
```

Este cambio agrega una nueva opción que hace que **ActiveRecord** almacene el nombre completo de la clase.

Para habilitar/deshabilitar esta característica, ud. debe incluir o editar lo siguiente en su **environment.rb**.

```
ActiveRecord::Base.store_full_sti_class = true
```

El valor por defecto es true.

EL MÉTODO TABLE_EXISTS?

Se agregó un nuevo método para la clase **AbstractAdapter**: **table_exists**. Es muy simple de usar:

```
>> ActiveRecord::Base.connection.table_exists?("users")
=> true
```

TIMESTAMPED MIGRATIONS

Cuando se está estudiando Rails o cuando estamos desarrolando algo sólos, las **migraciones** parecen ser la mejor solución a todos nuestros problemas. Pero cuando estamos trabajando en un proyecto con un equipo de desarrolladores de distintas partes del mundo, ud. descurbrirá (si aún no lo hizo) que las migrations no funcionan bien. Las nuevas migraciones (timestamped migrations) de Rails 2.1 vienen al rescate.

Antes de la llegada de **timestamped migrations**, cada nueva migración tenía un número antepuesto al nombre de la migración. Si dos migraciones eran generadas por distintos desarrolladores y no se comiteaba de inmediato, entonces podían terminar teniendo el mismo número de migración pero con diferente información. En este punto su esquema_info está desactualizado y ud. tiene un conflicto.

Hay varias formas de "intentar" resolver este problema. Se crearon muchos plugins con diferentes maneras de resolver esto. Más allá de los plugins disponibles, una cosa era clara: la antigua forma simplemente no funcionaba.

Si ud. estuviese usando Git, entonces se podría complicar mucho más, su equipo probablemente tenga algunas ramas (branches) con **migraciones** desactualizadas en todas ellas. Probablemente tenga serios conflictos a la ahora de hacer un merge de esas ramas (branches).

Para resolver este gran problema, el core team de Rails cambió el funcionanmiento de las migraciones. En vez de anteponer cada archivo de migración con un número, ahora se le antepone con un string basado en la hora **UTC** con el siguiente formato YYYYMMDDHHMMSS.

Así mismo se creó una nueva tabla llamada **schema_migrations**, esta almacena las **migraciones** ejecutadas hasta el momento. De esta manera, si alguien crea una migración con un número bajo, rails hará un **rollback** de las migraciones hasta la versión previa y luego se correrán todo de nuevo hasta la versión actual.

Esto aparentemente, resuelve el problema de las **migraciones**.

Hay una opción para deshabilitar esta caraterística, agregando esta línea en el archivo **envronment.rb**:

```
config.active_record.timestamped_migrations = false
```

También hay algunas tareas de rake nuevas para "navegar" a través de las **migraciones**:

```
rake db:migrate:up
rake db:migrate:down
```

Chapter 3

ActiveSupport

Active Support es una colección de varias clases y extensiones de bibliotecas que fueron consideradas muy útiles para las aplicaciones Ruby on Rails. (wikipedia)

ACTIVESUPPORT::COREEXTENSIONS::DATE::CALCULATIONS

Time#end_of_day

Retorna la fecha actual con la hora fijada en 11:59:59 PM.

Time#end_of_week

Retorna el fin de semana (Sunday 11:59:59 PM).

Time#end_of_quarter

Retorna un objeto date representando el fin del trimestre. En otras palabras, retorna el último día de algunos de los siguientes meses: marzo, junio, septiembre o diciembre, dependiendo de la fecha actual.

Time#end_of_year

Retorna 31 de Diciembre a las 11:59:59 PM.

Time#in_time_zone

Este método es similar a **Time#localtime**, excepto por el hecho de que usa **Time.zone** en vez de basarse en el timezone del sistema operativo. Se le puede pasar como parámetro tanto un objeto **TimeZone** como un **String**. Veamos algunos ejemplos:

```
Time.zone = 'Hawaii'
Time.utc(2000).in_time_zone
# => Fri, 31 Dec 1999 14:00:00 HST -10:00

Time.utc(2000).in_time_zone('Alaska')
# => Fri, 31 Dec 1999 15:00:00 AKST -09:00
```

Time#days_in_month

Se arregló un bug en el método **days_in_month**, el cual retornaba un número incorrecto de días en febrero cuando no se le especificaba el año.

El cambio consiste en usar el año actual como valor por defecto cuando no se le especifica el año en la llamada al método. Supongamos que estamos en un año bisiesto, veamos:

```
Loading development environment (Rails 2.0.2)
>> Time.days_in_month(2)
=> 28

Loading development environment (Rails 2.1.0)
>> Time.days_in_month(2)
=> 29
```

DateTime#to_f

La clase **DateTime** tiene un nuevo método llamado **to_f** que retorna la fecha como un flotante representando el número de segundos desde el Unix epoch (época unix, número de segundos desde el 1 de Enero de 1970 a la medianoche).

Date.current

La clase **Date** tiene un nuevo método llamado **current** el cual se usa en vez de **Date.today**, por que este considera el timezone especificado en **config.time_zone**, que en caso de que estar especificado retorna **Time.zone.today**, de no ser así, entonces retorna **Date.today**.

FRAGMENT_EXIST?

Se agregaron dos nuevos métodos a **cache_store**: **fragment_exist?** y **exist?**.

El método **fragment_exist?** hace exactamente lo que ud. esperaría, verifica si un fragmento está cacheado, informado a través de una clave existe. Básicamente sustituyendo al famoso:

```
read_fragment(path).nil?
```

El método **exist?** se agregó a **cache_store**, mientras que **fragment_exist?** es un helper el cual ud. podría usar dentro de su controlador.

¿UTC O GMT?

Una pregunta simple, pero interesante. Hasta ahora Rails estuvo usando mucho la sigla UTC, pero cuando se ejecuta el método **to_s** de **TimeZone**, este muestra GMT, no UTC. Esto es debido al hecho de que el acrónimo GMT es más común entre los usuarios finales.

Si ud. mira en el panel de control de Windows, dónde se puede seleccionar el timezone, verá el que el acrónimo utilizado es GMT. Google y Yahoo también están usando GMT dentro de sus productos.

```
TimeZone['Moscow'].to_s #=> "(GMT+03:00) Moscow"
```

JSON ESCAPE

El método **json_escape** funciona como **html_escape**. Es muy útil cuando necesitamos mostrar strings **JSON** en una página **HTML** , por ejemplo, en un proceso de documentación.

```
puts json_escape("is a > 0 & a < 10?")
# => is a \u003E 0 \u0026 a \u003C 10?
```

Podemos usar el atajo **j** en ERB:

```
<%= j @person.to_json %>
```

Si ud. desea escapar todo el código **JSON** por defecto, incluya la línea siguiente en su *environment.rb*:

```
ActiveSupport.escape_html_entities_in_json = true
```

MEM_CACHE_STORE AHORA ACEPTA OPCIONES

La inclusión de **Memcache-Client** dentro de **ActiveSupport::Cache** facilitó mucho las cosas, pero también eliminó la flexibilidad permitiendo configurar nada más que el número de IP del servidor de **memcached**.

Jonathan Weiss hizo un parche que fue incluido en Rails, que permite opciones extras como estas:

```
ActiveSupport::Cache.lookup_store :mem_cache_store, "localhost"

ActiveSupport::Cache.lookup_store :mem_cache_store, "localhost", '192.168.1.1',
        :namespace => 'foo'
```

o

```
config.action_controller.fragment_cache_store = :mem_cache_store, 'localhost',
        {:compression => true, :debug => true, :namespace =>'foo'}
```

TIME.CURRENT

Un nuevo método para la clase **Time**. El método **current** retorna dependiendo de si fue especificado antes **config.time_zone**, **Time.zone.now**, de otra manera retorna **Time.now**.

```
# retorna un valor que depende de config.time_zone
Time.current
```

los métodos **since** y **ago** también cambiaron sus valores de retorno, retornando un **TimeWithZone** en caso de que se especifique **config.time_zone**.

Esto convierte al método **Time.current** en el nuevo método por defecto para obtener la hora actual, reemplazando a **Time.now** (el cual sigue existiendo, pero no tiene en cuenta el timezone especificado).

Los métodos **datetime_select**, **select_datetime** y **select_time** se actualizaron para que tengan como valor de retorno **Time.current**.

ELIMINANDO LOS ESPACIOS EN BLANCO CON EL MÉTODO SQUISH

Se agregaron dos métodos al objeto **String**, **squish** y **squish!**.

Estos métodos hacen lo mismo que el método **strip**. Elimina los espacios en blanco del principio y del final de un texto. También elimina los espacios en blanco no utilizados (espacios en blanco consecutivos) en el medio de un texto. Veamos un ejemplo:

```
“    A    text    full    of    spaces    “.strip
#=> “A    text    full    of    spaces”
```

```
“    A    text    full    of    spaces    “.squish
#=> “A text full of spaces”
```

Chapter 4

ActiveResource

ActiveResource es la capa responsable de la implementación de sistemas RESTful del lado del cliente. A través de ActiveResource es posible consumir servicios RESTful mediante el uso de objetos que actúan como un proxy para los servicios remotos.

USANDO EL EMAIL COMO NOMBRE DE USUARIO.

Algunos servicios usan el e-mail como nombre de usuario, lo cual nos obliga a usar una URL como la siguiente:

```
http://ernesto.jimenez@negonation.com:pass@tractis.com
```

Pero nos genera un problema. Por que tenemos dos (@), el intérprete se pierde cuando lee algo como esto. Por este motivo, Se exntendió **ActiveResource** un poco más, con el fin de facilitar el uso de e-emails para la autenticación. Ahora ud. puede hacer lo siguiente:

```
class Person < ActiveResource::Base
  self.site = "http://tractis.com"
  self.user = "ernesto.jimenez@negonation.com"
  self.password = "pass"
end
```

EL MÉTODO CLONE

Ahora podemos clonar un recurso existente:

```
ryan = Person.find(1)
not_ryan = ryan.clone
not_ryan.new?  # => true
```

Note que el objeto copiado no ha clonado ningún atributo de la clase, apenas los atributos del recurso.

```
ryan = Person.find(1)
ryan.address = StreetAddress.find(1, :person_id => ryan.id)
ryan.hash = {:not => "an ARes instance"}

not_ryan = ryan.clone
not_ryan.new?            # => true
not_ryan.address         # => NoMethodError
not_ryan.hash            # => {:not => "an ARes instance"}
```

TIMEOUTS

ActiveResource usa **HTTP** para acceder a la API de RESTful y por esto es suceptible a problemas con servidores lentos o servidores no alcanzables. En algunos casos, las llamadas a ActiveResource pueden expirar (timeout). Ahora ud. puede controlar el tiempo de expiración con la propiedad timeout.

```
class Person < ActiveResource::Base
  self.site = "http://api.people.com:3000/"
  self.timeout = 5 # waits 5 seconds before expire
end
```

En el ejemplo de arriba configuramos un timeout de 5 segundos. Se recomienda un valor pequeño para permitirle a su sistema una falla rápida, previeniendo que una casacada de fallas deje su servidor fuera de servicio.

Internamente, ActiveResource se basa la biblioteca Net:HTTP para hacer las peticiones HTTP. Cuando ud. define un valor por defecto para la propiedad timeout, el mismo valor es definido para la propiedad **read_timeout** de la instancia del objeto Net:HTTP que está usando en ese momento.

El valor por defecto es 60 segundos.

Chapter 5

ActionPack

Está compuesto por ActionView (generación de visualización para el usuario, por ej: HTML, XML, JavaScript, etc.) y ActionController (controla el flujo de la aplicación) (wikipedia)

TIMEZONE

Definiendo un timezone por defecto

Se agregó una nueva opción al método **time_zone_select**, ahora ud. puede poner un valor por defecto en caso de que el usuario no haya seleccionado ningún **TimeZone**, o cuando la columna en la base de datos es null. Para hacer esto, se creó la opción **:default** que se puede usar de la siguiente manera:

```
time_zone_select("user", "time_zone", nil, :include_blank => true)
```

```
time_zone_select("user", "time_zone", nil,
        :default => "Pacific Time (US & Canada)" )

time_zone_select( "user", 'time_zone', TimeZone.us_zones,
        :default => "Pacific Time (US & Canada)")
```

En los casos dónde usemos la opción **:default**, debe aparecer con el **TimeZone** especificado ya selecionado.

El método formatted_offset

Se incluyó el método **formatted_offset** en las clases **Time** y **DateTime** para retornar con el formato **+HH:MM** la desviación de la hora UTC. Por ejemplo, en nuestro timezone (Hora Brasilia) el valor de desviación retornado por el método sería un string con el valor: **"-03:00"**.

Veamos algunos ejemplos:

Obteniendo la desviación desde un DateTime:

```
datetime = DateTime.civil(2000, 1, 1, 0, 0, 0, Rational(-6, 24))
datetime.formatted_offset         # => "-06:00"
datetime.formatted_offset(false)  # => "-0600"
```

Ahora desde Time:

```
Time.local(2000).formatted_offset         # => "-06:00"
Time.local(2000).formatted_offset(false)  # => "-0600"
```

Note que este método retorna un **string**, el cual puede ser formateado o no dependiendo del valor dado como parámetro.

El método with_env_tz

El método **with_env_tz** nos permite hacer tests con diferentes timezones de una manera muy simple:

```
def test_local_offset
  with_env_tz 'US/Eastern' do
    assert_equal Rational(-5, 24), DateTime.local_offset
  end
  with_env_tz 'US/Central' do
    assert_equal Rational(-6, 24), DateTime.local_offset
  end
end
```

Este helper

This helper was supposed to call **with_timezone**, but it was renamed for **with_env_tz** to avoid confusion with the timezone informed by using **ENV['TZ']** and **Time.zone**.

Time.zone_reset!

Se eliminó por que ya no se usa más.

Time#in_current_time_zone

Se modificó para que retorne **self** cuando **Time.zone** es null.

Time#change_time_zone_to_current

Se modificó para que retorne **self** cuando **Time.zone** es null.

TimeZone#now

El método **TimeZone#now** se modificó para que retorne un objeto **ActiveSupport::TimeWithZone** represetando la hora actual en el timezone configurado en **Time.zone**. Por ejemplo:

```
Time.zone = 'Hawaii'  # => "Hawaii"
Time.zone.now         # => Wed, 23 Jan 2008 20:24:27 HST -10:00
```

Compare_with_coercion

Se creó el método **compare_with_coercion** (con un alias para <=>) en las clases **Time** y **DateTime**, posibilitando la comparación cronológica entre las clases **Time**, **DateTime** y las instancias de **ActiveSupport::TimeWithZone**. Para una mejor comprensión, veamos los siguientes ejemplos:

```
Time.utc(2000) <=> Time.utc(1999, 12, 31, 23, 59, 59, 999) # 1
Time.utc(2000) <=> Time.utc(2000, 1, 1, 0, 0, 0) # 0
Time.utc(2000) <=> Time.utc(2000, 1, 1, 0, 0, 0, 001)) # -1

Time.utc(2000) <=> DateTime.civil(1999, 12, 31, 23, 59, 59) # 1
Time.utc(2000) <=> DateTime.civil(2000, 1, 1, 0, 0, 0) # 0
Time.utc(2000) <=> DateTime.civil(2000, 1, 1, 0, 0, 1)) # -1

Time.utc(2000) <=> ActiveSupport::TimeWithZone.new(Time.utc(1999, 12, 31, 23, 59, 59) )
Time.utc(2000) <=> ActiveSupport::TimeWithZone.new(Time.utc(2000, 1, 1, 0, 0, 0) )
Time.utc(2000) <=> ActiveSupport::TimeWithZone.new(Time.utc(2000, 1, 1, 0, 0, 1) ))
```

TimeWithZone#between?

Se incluyó el método **between?** en la clase **TimeWithZone** para verificar si una instancia está entre dos fechas. Por ejemplo:

```
@twz.between?(Time.utc(1999,12,31,23,59,59),
              Time.utc(2000,1,1,0,0,1))
```

TimeZone#parse

Este método crea una nueva instancia de **ActiveSupport::TimeWithZone** a partir de un string. Por ejemplo:

```
Time.zone = "Hawaii"
# => "Hawaii"
Time.zone.parse('1999-12-31 14:00:00')
# => Fri, 31 Dec 1999 14:00:00 HST -10:00


Time.zone.now
# => Fri, 31 Dec 1999 14:00:00 HST -10:00
Time.zone.parse('22:30:00')
# => Fri, 31 Dec 1999 22:30:00 HST -10:00
```

TimeZone#at

Este método crea una nueva instancia de **ActiveSupport::TimeWithZone** a partir del número de segundos desde la fecha Unix epoch. Por ejemplo:

```
Time.zone = "Hawaii" # => "Hawaii"
Time.utc(2000).to_f  # => 946684800.0
```

```
Time.zone.at(946684800.0)
# => Fri, 31 Dec 1999 14:00:00 HST -10:00
```

Más métodos

Los métodos **to_a**, **to_f**, **to_i**, **httpdate**, **rfc2822**, **to_yaml**, **to_datetime** y **eql?** se agregaron a la clase **TImeWithZone**. Para más información acerca de estos métodos por favor lea la documentación de Rails.

Se preparó la clase TimeWithZone para Ruby 1.9

En Ruby 1.9 tendremos algunos métodos nuevos en la clase **Time**, como por ejemplo:

```
Time.now
# => Thu Nov 03 18:58:25 CET 2005

Time.now.sunday?
# => false
```

Este mismo método existe para cada día de la semana

Otras cosa interesante es que el método **to_s** del objeto **Time** tendrá un valor de retorno diferente. Hoy cuando ejecutamos **Time.new.to_s**, obtenemos lo siguiente:

```
Time.new.to_s
# => "Thu Oct 12 10:39:27 +0200 2006"
```

En Ruby 1.9 vamos a tener:

```
Time.new.to_s
# => "2006-10-12 10:39:24 +0200"
```

¿Que tiene que ver todo esto con Rails 2.1? Todo. Rails se está preparando para proveer estas modificaciones. La clase **TimeWithZone**, por ejemplo, sólo tuvo algunas mejoras para funcionar con los métodos del primer ejemplo.

AUTO LINK

El método **auto_link** recibe cualquier texto como parámetro, y si el texto contiene alguna dirección de email o website, retorna el mismo texto, pero con hyperlinks.

Por ejemplo:

```
auto_link("Go to this website now: http://www.rubyonrails.com")
# => Go to this website now: http://www.rubyonrails.com
```

Algunos sites, como Amazon, usan el símbolo "=" en sus URL. Este método no reconoce este símbolo. Veamos como se comporta este método en tal caso:

```
auto_link("http://www.amazon.com/Testing/ref=pd_bbs_sr_1")
# => http://www.amazon.com/Testing/ref
```

Note que el método corta el hyperlink exactamente antes del símbolo "=", antes de Rails 2.1 este símbolo estaba soportado.

El mismo método se actualizó para permitir el uso de URL con paréntesis.

Un ejemplo de URL coc paréntesis:

```
http://en.wikipedia.org/wiki/Sprite_(computer_graphics)
```

ETIQUETAS

Cuando creamos un nuevo formulario con **scaffold** será creado con el siguiente código:

```
<% form_for(@post) do |f| %>
  <p>
    <%= f.label :title %><br />
    <%= f.text_field :title %>
  </p>
  <p>
    <%= f.label :body %><br />
    <%= f.text_area :body %>
  </p>
  <p>
    <%= f.submit "Update" %>
  </p>
<% end %>
```

Se incluyó el método **label**. Este método retorna un *string* con el título de la columna dentro de un tag HTML **<label>**.

```
>> f.label :title
=> <label for="post_title">Title</label>

>> f.label :title, "A short title"
=> <label for="post_title">A short title</label>
```

```
>> label :title, "A short title", :class => "title_label"
=> <label for="post_title" class="title_label">A short title</label>
```

¿Se dió cuenta del parámetro **for** dentro del tag? "post_title" es el título que contiene nuetro título para el post. El tag **<label>** de hecho una etiqueta asociada al objeto **post_title**. Cuando alguien hace click sobre la etiqueta (que no es un link) el controlador HTML asociado recive el foco.

Robby Russell escribió un post interesante en su blog acerca de este tema. Ud. lo puede leer en: http://www.robbyonrails.com/articles/2007/12/02/that-checkbox-needs-a-label

También se incluyó el método **label_tag** en **FormTagHelper**. Este método funciona como label, pero de una forma más simple:

```
>> label_tag 'name'
=> <label for="name">Name</label>

>> label_tag 'name', 'Your name'
=> <label for="name">Your name</label>

>> label_tag 'name', nil, :class => 'small_label'
=> <label for="name" class="small_label">Name</label>
```

El método acepta la opción **:for**, Veamos un ejemplo:

```
label(:post, :title, nil, :for => "my_for")
```

Esto va retornar algo como lo siguiente:

```
<label for="my_for">Title</label>
```

UNA NUEVA FORMA DE USAR PARTIALS

El uso de partials es algo muy común para eliminar la duplicación de código en el desarrollo de software con Rails. Aquí hay un ejemplo:

```
<% form_for :user, :url => users_path do %>
        <%= render :partial => 'form' %>
        <%= submit_tag 'Create' %>
<% end %>
```

Un **Partial** es un fragmento de código (una plantilla). La ventaja de usar un **partial** es eliminar la duplicación innecesaria de código. Usar un **partial** es muy simple, puede comenzar con algo como esto: **render :partial => "name"**. Y debe crear un archivo con el mismo nombre que su **partial**, pero anteponiéndole un guión bajo

En el código de arriba vimos como podemos usar un patial, en Rails 2.1 ud. va a poder hacer lo mismo pero una forma un poco diferente:

```
<% form_for(@user) do |f| %>
        <%= render :partial => f %>
        <%= submit_tag 'Create' %>
<% end %>
```

En este ejemplo vamos a renderizar el partial "users/_form", el cual va a recibir una variable llamada "form" con las referencias creadas por **FormBuilder**.

La vieja forma continúa funcionando en Rails 2.1

NUEVOS ESPACIOS DE NOMBRES EN ATOM FEED

¿Conoce el método **atom_feed**? Es una de las nuevas características de Rails 2.0, que facilita la creación de Feeds de Atom. Veamos un ejemplo de como se usa:

En el archivo *index.atom.builder*:

```
atom_feed do |feed|
  feed.title("Nome do Jogo")
  feed.updated((@posts.first.created_at))

  for post in @posts
    feed.entry(post) do |entry|
      entry.title(post.title)
      entry.content(post.body, :type => 'html')

      entry.author do |author|
        author.name("Carlos Brando")
      end
    end
  end
end
```

¿Qué es un Atom feed? Atom es el nombre de estilos y metadatos en formato XML. En otras palabras es un protocolo para publicar contenidos en Internet que son actualizados con frecuencia, como un blog por ejemplo. Los Feeds se publican siempre en XML y en Atom son identificados como tipo de medios application/atom+xml.

En las primeras versiones de Rails 2.0 este método aceptaba parámetros como: **:language**, **:root_url** and **:url**, puede obtener más información acerca de estos métodos en la documentación de Rails. pero con la nueva modificación, ahora podemos incluir nuevos espacios de nombre en el elemento raíz del feed. Por ejemplo:

```
atom_feed('xmlns:app' => 'http://www.w3.org/2007/app') do |feed|
```

Va a retornar:

```
<feed xml:lang="en-US" xmlns="http://www.w3.org/2005/Atom"
        xmlns:app="http://www.w3.org/2007/app">
```

Modificando el ejemplo usado antes, podríamos hacer esto:

```
atom_feed({'xmlns:app' => 'http://www.w3.org/2007/app',
        'xmlns:openSearch' => 'http://a9.com/-/spec/opensearch/1.1/'}) do |feed|

  feed.title("Nome do Jogo")
  feed.updated((@posts.first.created_at))
  feed.tag!(openSearch:totalResults, 10)

  for post in @posts
    feed.entry(post) do |entry|
      entry.title(post.title)
      entry.content(post.body, :type => 'html')
      entry.tag!('app:edited', Time.now)

      entry.author do |author|
        author.name("Carlos Brando")
      end
    end
  end
end
```

CACHE

Ahora todos los métodos de **fragment_cache_key** retornan por defecto el namespace 'view/' como prefijo.

Se eliminaron todos los caching store de **ActionController::Caching::Fragments::** *y ahora están en* ***ActiveSupport::Cache::***. En este caso, si ud. a hecho una referencia a store, como por ejemplo: **ActionController::Caching::Fragments::MemoryStore**, va a tener que cambiarla por: **ActiveSupport::Cache::MemoryStore**.

ActionController::Base.fragment_cache_store* no va más, y ActionController::Base.cache_store** toma su lugar.

Se incluyó en **ActiveRecord::Base** el método **cache_key** que facilita el modo de almacenamiento de cache de Active Records mediante la nueva biblioteca **ActiveSupport::Cache::***. Funciona así:

```
>> Product.new.cache_key
=> "products/new"

>> Product.find(5).cache_key
=> "products/5"

>> Person.find(5).cache_key
=> "people/5-20071224150000"
```

Se incluyó **ActiveSupport::Gzip.decompress/compress** para facilitar el uso de como mapeo para **Zlib**.

Ahora ud. puede usar las opciones de ambiente (environment) **config.cache_store** para especificar el valor por defecto del tipo de almacenamiento de cache. Como hemos mencionado si existe el directorio **tmp/cache**, el valor por defecto es **FileStore**, en otro caso se usará **MemoryStore**, puede configurarlo de la siguiente manera:

```
config.cache_store = :memory_store
config.cache_store = :file_store, "/path/to/cache/directory"
config.cache_store = :drb_store, "druby://localhost:9192"
config.cache_store = :mem_cache_store, "localhost"
config.cache_store = MyOwnStore.new("parameter")
```

Para hace las cosas más fáciles aún, el comentario de abajo se incluye en *environments/production.rb*, en favor de recordarle estas opciones.

```
# Use a different cache store in production
# config.cache_store = :mem_cache_store
```

APLICANDO FORMATO DE TÍTULO EN STRINGS.

Había un bug cuando usábamos el método **String#titleize** en un string que contenía 's. El bug hacía que el método returne la 's en mayúsculas. Vea el ejemplo:

```
>> "brando's blog".titleize
=> "Brando'S Blog"
```

Vea el ejemplo que con el bug corregido:

```
>> "brando's blog".titleize
=> "Brando's Blog"
```

ACTION_NAME

Para saber cual vista fue invocada durante la ejecución de su vista, podemos usar el método **action_name**:

```
<%= action_name %>
```

El valor de retorno va a ser el mismo que usando **params[:action]**, pero de una forma más elegante.

CACHES_ACTION ACEPTA CONDICIONALES

El método **caches_action** ahora acepta la opción **:if**, permitiendo el uso de condicionales para especificar cuando una acción (**action**) puede ser cacheada (**cached**) Por ejemplo:

```
caches_action :index, :if => Proc.new { |c| !c.request.format.json? }
```

En el ejemplo de arriba, la **acción index** será cacheada únicamente si no es accedida por una petición JSON.

CONDITIONAL IN THE CACHES_PAGE METHOD

The **caches_page** method now has the option to use conditionals (**:if**). See example:

```
# The Rails 2.0 way
caches_page :index

# In Rails 2.1 you can use :if option
caches_page :index, :if => Proc.new { |c| !c.request.format.json? }
```

FLASH.NOW NOW WORKS IN TESTS

Who didn't have headaches because of this ? The problem was that during tests we could never confirm if a message was stored in flash, because it was cleared by Rails before going to your test script.

In rails 2.1 the problem was solved. Now you can include the following line in your tests:

```
assert_equal '>value_now<', flash['test_now']
```

ACCESSING HELPERS OUTSIDE VIEWS

How often have you created a **helper** and wished you could use it inside your **controller** ? To achieve this functionality you had to included the **helper** module inside the **controller**, which made your code look ugly.

In Rails 2.1 a proxy to access helpers outside views was developed. It works in a very simple way:

```
# To access simple_format method, for example
ApplicationController.helpers.simple_format(text)
```

Simple and Clean!

JSON

Rails now accepts POST's requests of JSON content. For example, you can send a POST request this way:

```
POST /posts
{"post": {"title": "Breaking News"}}
```

And everything goes to variable **params**. This works for example:

```
def create
  @post = Post.create params[:post]
  # …
end
```

Some of you many not know JSON is a "competitor" for XML, and it is widely used for JavaScript data interchange because it's represented in this language. It takes its name from: **JavaScript Object Notation**.

PATH NAMES

My blog readers (http://www.nomedojogo.com) should know about my **Custom Resource Name** plugin. I think it'll die very soon... :(

In rails you could already include the option **:as** in your routes (something I implemented in my plugin to keep compatibility). Now you will also have the **:path_names** option to change the name of your **actions**.

```
map.resource :schools, :as => 'escolas', :path_names => { :new => 'nova' }
```

Of course, my plugin will remain being useful for users of earlier Rails versions.

DEFINING THE LOCATION OF YOUR ROUTES FILE

In Rails 2.1 you can define in which file your routes are stored, including the following line in your *enviroment.rb*:

```
config.routes_configuration_file
```

This can be useful in a scenario where you have two separated front-ends that share the same modules, libraries and plugins.

For example, getsatisfaction.com and api.getsatisfaction.com share the same models, but not the controllers, helpers and views. getsatisfaction has its own routes file with optimizations to improve its SEO, while the API route's file doesn't know anything about SEO improvements.

SESSION(:ON)

Did you know it is possible to turn off sessions in rails? Here is how to do it:

```
class ApplicationController < ActionController::Base
  session :off
end
```

Note that in my example I'm turning off sessions for all controllers (**ApplicationController**), but I could also do it for a single controller.

If you want to have sessions on for a given controller in Rails 2.1 you can use the session method method passing the **:on** parameter:

```
class UsersController < ApplicationController
  session :on
end
```

TESTING HELPERS IN A SIMPLE WAY

One very boring thing to do in earlier versions of Rails is testing the **helpers**. I already suffered a lot to ensure 100% of coverage, creating tests for some **helpers**.

This became much simpler in Rails 2.1 with the **ActionView::TestCase** class. Look the example:

```
module PeopleHelper
  def title(text)
    content_tag(:h1, text)
  end

  def homepage_path
    people_path
  end
end
```

Now look how we can do the same in Rails 2.1:

```
class PeopleHelperTest < ActionView::TestCase
  def setup
    ActionController::Routing::Routes.draw do |map|
      map.people 'people', :controller => 'people', :action => 'index'
      map.connect ':controller/:action/:id'
    end
  end

  def test_title
    assert_equal "<h1>Ruby on Rails</h1>", title("Ruby on Rails")
  end

  def test_homepage_path
```

```
    assert_equal "/people", homepage_path
  end
end
```

Chapter 6

ActionController

ActionController es la capa responsable de recibir las peticiones web y tomar decisiones con respecto a qué se debe ejecutar y renderizar o si se debe redirigir la petición a otra acción Una acción es definida como método público dentro de los controladores que están automáticamente disponibles a través de las rutas.

ACTIONCONTROLLER::ROUTING

Map.root

Ahora, podemos usar un alias con **map.root** para que sea mucho más **DRY**.

En las primeras versiones de rails ud. hacia esto:

```
map.new_session :controller => 'sessions', :action => 'new'
map.root :controller => 'sessions', :action => 'new'
```

Ahora puede hacer lo mismo de la siguiente manera:

```
map.new_session :controller => 'sessions', :action => 'new'
map.root :new_session
```

Reconocimiento de Rutas

La vieja implementación del reconocimiento de rutas recorría todas la rutas una a una, lo que hacía que terminara consumiendo mucho tiempo. Se desarrolló una nueva y más inteligente implementación. Esta crea un árbol para las rutas y el reconocimiento de rutas se hace mediante un prefijo, dejando de lado las rutas similares. Esta aproximación baja el tiempo de reconocimiento en aproximadamente 2.7 veces.

Todas las nuevas implementaciones están en el archivo **recognition_optimisation.rb** y los detalles de su funcionamiento están en los comentarios. Vea la documentación dentro del código fuente para más información.

Assert_routing

Ahora es posible testear una ruta con un método HTTP. Vea el siguiente ejempo:

```
assert_routing({ :method => 'put',
                 :path => '/product/321' },
               { :controller => "product",
                 :action => "update",
                 :id => "321" })
```

Map.resources

Image que tiene un sitio escrito en español, y quiere remodelar todas las rutas para que también para que se correspondan con el idioma. En otras palabras, en vez de tener:

```
http://www.mysite.com.br/products/1234/reviews
```

quisiera tener algo así:

```
http://www.mysite.com.br/productos/1234/comentarios
```

Esto era posible, pero no en de una forma simple y sin comprometer algunas de las convenciones de rails.

Ahora tenemos la opción **:as** dentro de **map.resources** para personalizar nuestras rutas. Mire el ejemplo para obtener la URL de arriba en español:

```
map.resources :products, :as => 'productos' do |product|
  # product_reviews_path(product) ==
  # '/productos/1234/comentarios
  product.resources :product_reviews, :as => 'comentarios'
end
```

ACTIONCONTROLLER::CACHING::SWEEPING

En las primeras versiones de Rails, cuando declarábamos un **sweeper**, teníamos que informar la clase usando símbolos:

```
class ListsController < ApplicationController
  caches_action :index, :show, :public, :feed
```

```
  cache_sweeper :list_sweeper,
                :only => [ :edit, :destroy, :share ]
end
```

Ahora es posible declarar explícitamente una clase en vez de usar un símbolo. Esto es necesario si ud. quiere que el **sweeper** esté dentro de un módulo. Además de poder seguir usando símbolos para los demás casos, a partir de ahora puede hacer así:

```
class ListsController < ApplicationController
  caches_action :index, :show, :public, :feed
  cache_sweeper OpenBar::Sweeper,
                :only => [ :edit, :destroy, :share ]
end
```

Chapter 7

ActionView

ActionView es la capa responsable de la generación de interfaces visibles al usuario a través de la conversión de plantillas ERB.

ACTIONVIEW::HELPERS::FORMHELPER

fields_for form_for con la opción index.

Los métodos **#fields_for** y **form_for** recibían la opción **:index**, eliminando la necesidad de utilizar **:index => nil** en cada objeto form. Mira los ejemplos:

Así es como solía ser el código antes:

```
<% fields_for "project[task_attributes][]", task do |f| %>
  <%= f.text_field :name, :index => nil %>
  <%= f.hidden_field :id, :index => nil %>
  <%= f.hidden_field :should_destroy, :index => nil %>
<% end %>
```

Ahora luce de la siguiente forma:

```
<% fields_for "project[task_attributes][]", task,
              :index => nil do |f| %>
  <%= f.text_field :name %>
  <%= f.hidden_field :id %>
  <%= f.hidden_field :should_destroy %>
<% end %>
```

ACTIONVIEW::HELPERS::DATEHELPER

Ahora todos estos métodos de módulo para trabajar con fechas (**date_select**, **time_select**, **select_datetime**, etc.) aceptan opciones **HTML**. Mira un ejemplo usando **date_select**

```
<%= date_select 'item','happening', :order => [:day], :class => 'foobar'%>
```

date_helper

El método **date_helper** fue actualizado para que use **Date.current** en orden de definir su valor predeterminado.

ACTIONVIEW::HELPERS::ASSETTAGHELPER

register_javascript_expansion

Este método registra uno o más archivos javascript para su inclusión cuando un símbolo, definido por el programador, es indicado como parámetro del método **javascript_include_tag**. La idea es llamar a este método dentro del archivo **init.rb** de tu plugin, en orden de registrar los archivos javascript que tu plugin ubica en la carpeta **public/javascripts**. Veamos como funciona:

```
# En el archivo init.rb
ActionView::Helpers::AssetTagHelper.register_javascript_expansion
    :monkey => ["head", "body", "tail"]

# En nuestar vista:
javascript_include_tag :monkey

# Vamos a obtener:
<script type="text/javascript" src="/javascripts/head.js"></script>
<script type="text/javascript" src="/javascripts/body.js"></script>
<script type="text/javascript" src="/javascripts/tail.js"></script>
```

register_stylesheet_expansion

Este método hace exactamente lo mismo que que el método **ActionView::Helpers::AssetTagHelper#register_javascript_expansion**, pero crea un símbolo para ser usado luego cuando se realicen llamadas al método **stylesheet_link_tag**. Veamos un ejemplo:

```
# En el archivo init.rb
ActionView::Helpers::AssetTagHelper.register_stylesheet_expansion
```

```
    :monkey => ["head", "body", "tail"]

# En nuestar vista:
stylesheet_link_tag :monkey

# Vamos a obtener:
<link href="/stylesheets/head.css"  media="screen" rel="stylesheet"
    type="text/css" />
<link href="/stylesheets/body.css"  media="screen" rel="stylesheet"
    type="text/css" />
<link href="/stylesheets/tail.css"  media="screen" rel="stylesheet"
    type="text/css" />
```

ACTIONVIEW::HELPERS::FORMTAGHELPER

submit_tag

Se agregó la opción **:confirm** a los parámetros del método **#submit_tag**. Esta opción funciona de la misma forma que el método **link_to**. Veamos un ejemplo:

```
submit_tag('Save changes', :confirm => "Are you sure?")
```

ACTIONVIEW::HELPERS::NUMBERHELPER

number_to_currency

El método **number_to_currency** ahora acepta la opción **:format** como parámetro, permitiéndonos dar formato al valor retornado. En versiones anteriores, cuando teníamos que dar formato a monedas locales, debíamos incluir un espacio antes de la opción **:unit** para que el formato retornado sea el correcto. Veamos un ejemplo:

```
# R$ es el símbolo de la moneda de Brasil.
number_to_currency(9.99, :separator => ",", :delimiter => ".", :unit => "R$")
# => "R$9,99"

number_to_currency(9.99, :format => "%u %n", :separator => ",",
    :delimiter => ".", :unit => "R$")
# => "R$ 9,99"
```

Por otro lado, podemos personalizar de otras formas, por ejemplo:

```
number_to_currency(9.99, :format => "%n en reales brasileros", :separator => ",",
    :delimiter => ".", :unit => "R$")
# => "9,99 em reais"
```

Cuando creas tu cadena de formateo, se puede especificar el siguiente parámetro:

```
%u Para la moneda
%n Para el número
```

ACTIONVIEW::HELPERS::TEXTHELPER

excerpt

El método **excerpt** nos ayuda a encontrar una palabra dentro de una frase y nos devuelve la abreviatura de esa frase con el número de caracteres que se hayan especificado como parámetro antes y después de la palabra agregando "…" cuando sea necesario. Veamos un ejemplo:

```
excerpt('This is an example', 'an', 5)
# => "…s is an examp…"
```

Pero había un problema. Si cuentas, vas a notar que el método retorna 6 caracteres y no 5. Esto era un bug y se corrigió. Mira el ejemplo de la salida correcta de este método:

```
excerpt('This is an example', 'an', 5)
# => "…s is an exam…"
```

simple_format

El método **simple_format** básicamente recibe como parámetro cualquier texto y le da formato de una manera muy simple a **HTML**. Toma el texto y reemplaza los saltos de línea (\n) por el tag **HTML** "< br />". Y cuando tenemos dos saltos de línea, uno atrás del otro, separa el texto con tags "< p>".

En Rails 2.1 este método recibe un parámetro adicional. Además del texto, podemos informarle que atributo **HTML** nos gustaria que el tag "< p>" incluyera. Veamos el ejemplo:

```
simple_format("Hello Mom!", :class => 'description')
# => "<p class='description'>Hello Mom!</p>"
```

Los atributos **HTML** serán agregados a todos los tags "< p>" creados por este método.

Chapter 8

Railties

CONFIG.GEM

Ahora es posible configurar todas las gemas necesarias para obtener un proyecto listo para su ejecución usando una nueva característica llamada **config.gem**. En el archivo **environment.rb** puedes especificar de qué gemas depende tu proyecto para ejecutarse correctamente. Veamos un ejemplo:

```
config.gem "bj"

config.gem "hpricot", :version => '0.6',
                      :source => "http://code.whytheluckystiff.net"

config.gem "aws-s3", :lib => "aws/s3"
```

Para instalar todas las dependencias de una sola vez, utilizamos una tarea Rake:

```
# Instala todas las gemas especificadas
rake gems:install
```

También es posible listar cuales gemas están siendo usadas en el proyecto ejecutando:

```
# Listando todas las dependencias
rake gems
```

Si una de las gemas tiene un archivo **rails/init.rb** y vos querés llevar la gema con tu aplicación, puedes hacer:

```
# Copia la gema especificada a vendor/gems/nome_do_gem-x.x.x
rake gems:unpack GEM=gem_name
```

Entonces, la gema será copiada al directorio **vendor/gems/gem_name-x.x.x**. En caso de no especificar el nombre de la gema, Rails copiará todas las gemas al directorio **vendor/gem**

CONFIG.GEM EN PLUGINS

El comando **config.gem** está también disponible para ser utilizado con plugins.

Hasta Rails 2.0, el archivo **init.rb** de un plugin solía lucir de la siguiente forma:

```
# init.rb del plugin open_id_authentication
require 'yadis'
require 'openid'
ActionController::Base.send :include, OpenIdAuthentication
```

Pero en Rails 2.1 el archivo **init.rb** sería:

```
config.gem "ruby-openid", :lib => "openid", :version => "1.1.4"
config.gem "ruby-yadis",  :lib => "yadis",  :version => "0.3.4"

config.after_initialize do
  ActionController::Base.send :include, OpenIdAuthentication
end
```

Así que, cuando ejecute la tarea para instalar todas las gemas necesarias, éstas estarán entre ellas.

GEMS:BUILD

La tarea **gems:build** compila todas las extensiones de las gemas que fueron instaladas a través de **gems:unpack**. La sintaxis es la siguiente:

```
rake gems:build # Para todas las gemas
rake gems:build GEM=mygem # Estoy especificando una gema
```

NUEVO MENSAJE CUANDO SE INICIALIZA EL SERVIDOR

Hubo una pequeña mejora en el inicio del servidor de Rails. Ahora se muestra cual versión está siendo cargada:

```
Rails 2.1 application starting on http://0.0.0.0:3000
```

RAILS.PUBLIC_PATH

Se agregó un acceso directo para recuperar el path del directorio **"public"** de un proyecto.

```
Rails.public_path
```

RAILS.LOGGER, RAILS.ROOT, RAILS.ENV Y RAILS.CACHE

En Rails 2.1 en vez de usar las constantes: **RAILS_DEFAULT_LOGGER**, **RAILS_ROOT**, **RAILS_ENV** y **RAILS_CACHE**, usaremos:

```
# RAILS_DEFAULT_LOGGER
Rails.logger

# RAILS_ROOT
Rails.root

# RAILS_ENV
Rails.env

# RAILS_CACHE
Rails.cache
```

RAILS.VERSION

En versiones anteriores para descubrir, en tiempo de ejecución, cual es la versión de Rails en uso usabamos:

```
Rails::VERSION::STRING
```

En Rails 2.1 usamos:

```
Rails.version
```

OBTENIENDO INFORMACIÓN ACERCA DE UN PLUGIN

Esta es una de las nuevas características de Rails 2.1 que probablemente nunca use. Digo "probablemente", porque en algunos casos específicos puede ser útil, por ejemplo, para saber la versión de un plugin.

Para probarlo, necesitamos crear un archivo que se llame *about.yml* en el directorio del plugin, con algo como lo siguiente:

```
author: Carlos Brando
version: 1.2.0
description: Una descripción acerca del plugin
url: http://www.nomedojogo.com
```

Podemos obtener esta información luego de esta manera:

```
plugin = Rails::Plugin.new(plugin_directory)
plugin.about["author"] # => "Carlos Brando"
plugin.about["url"] # => "http://www.nomedojogo.com"
```

Si encuentras algún buen uso de esta carácterística y la quieres compartir conmigo, tal vez cambie mi opinión acerca de su real necesidad.

Chapter 9

Rake Tasks, Plugins y Scripts

TAREAS

rails:update

De ahora en más, cada vez que se ejecute la tarea **rake rails:freeze:edge**, esta también ejecutará **rails:update**, actualizando los archivos de configuración y los *JavaScripts*.

Base de datos en 127.0.0.1

Un cambio fue realizado en el archivo database.rake que solía buscar unicamente en localhost bases de datos locales. Ahora considerará **127.0.0.1**. Esto funciona para las tareas **create** y **drop**. El archivo database.rake también fue refactorizado para hacer el código menos repetitivo.

Congelando a una versión de Rails específica.

Hasta Rails 2.1 no era posible congelar a una versión de Rails específica dentro de un proyecto. Sólo se podía utilizar la Revisión como parámetro. En Rails 2.1, podemos congelar a una versión específica con el siguiente comando:

```
rake rails:freeze:edge RELEASE=1.2.0
```

TIMEZONE

rake time:zones:all

Devuelve todas las zonas horarias conocidas para Rails, agrupadas por uso horario. También se puede filtrar los valores retornados usando el parámetro óptico OFFSET, por ejemplo: OFFSET=-6.

rake time:zones:us

Muestra lista con todas las zonas horarias de Estados Unidos. La opción OFFSET también es válida en este contexto.

rake time:zones:local

Retorna todas las zonas horarias conocidas por Rails que están en el mismo uso horario que tu sistema operativo.

SCRIPTS

plugin

El comando script/plugin install ahora provee la opción -e/--export que emite un svn export.
Se agregó soporte para plugins alojados en repositorios GIT.

dbconsole

Este script hace lo mismo que script/console, solo que para la base de datos. En otras palabras, te conecta con el cliente por línea de comando de tu base de datos.

Mirando al código, éste funcionaría solamente con mysql, postgresql y sqlite(3). Cuando sea especificada otra base de datos en el archivo database.yml, el script mostrará: "not supported for this database type".

PLUGINS

Las Gemas ahora pueden ser plugins

Ahora, cualquier gema que tenga un archivo **rails/init.rb** puede ser instalado dentro del directorio **vendor** de tu proyecto Rails igual que un **plugin**.

Usando generadores en plugins

Es posible configurar **Rails** para buscar **plugins** en directorios diferentes a **vendor/plugins**, solo incluyendo esta línea de código en el archivo **environment.rb**.

```
config.plugin_paths = ['lib/plugins', 'vendor/plugins']
```

Rails 2.0 tenía un bug en esta configuración que mostraba cuando un plugin tenía generadores. Por ese bug, Rails solo encuentraba generadores en plugins que estuvieran dentro de **vendor/plugins**. En Rails 2.1 ese bug fue corregido.

Chapter 10

Prototype y script.aculo.us

PROTOTYPE

Rails 2.1 ahora usa Prototype versión 1.6.0.1. Este sirve como base de la versión 1.8.1 de script.aculo.us.

Chapter 11

Ruby 1.9

DETAILS

El foco principal de los cambios de Rails fue Ruby 1.9. Aspectos menores fueron analizados para incrementar la compatibilidad de Rails con la nueva versión de Ruby. Detalles como cambiar **File.exists?** por **File.exist?** no fueron dejados de lado.

Además, en Ruby 1.9, el módulo **Base64** (base64.rb) fue eliminado, por esa razón, todas las referencias fueron remplazadas por **ActiveSupport::Base64**.

NUEVOS MÉTODOS PARA LA CLASE DATETIME

Con el objetivo de mantener compatibilidad (duck-typing) con la clase **Time**, tres nuevos métodos fueron agregados a la clase **DateTime**. Los métodos son **#utc**, **#utc?** y **#utc_offset**. veamos un ejemplo de cada uno:

```
>> date = DateTime.civil(2005, 2, 21, 10, 11, 12, Rational(-6, 24))
#=> Mon, 21 Feb 2005 10:11:12 -0600

>> date.utc
#=> Mon, 21 Feb 2005 16:11:12 +0000

>> DateTime.civil(2005, 2, 21, 10, 11, 12, Rational(-6, 24)).utc?
#=> false

>> DateTime.civil(2005, 2, 21, 10, 11, 12, 0).utc?
#=> true

>> DateTime.civil(2005, 2, 21, 10, 11, 12, Rational(-6, 24)).utc_offset
#=> -21600
```

Chapter 12

Debug

RUBY-DEBUG NATIVO

Fue habilitada nuevamente la opción para utilizar **ruby-debug** en pruebas con Rails. Ahora, teniendo en cuenta que la gema debe estar instalada, sólo se necesita el usar el método **debugger**.

Chapter 13

Bugs y arreglos

AGREGAR COLUMNAS EN POSTGRESQL

Existía un bug cuando se usaba **PostgreSQL**. El bug ocurría cuando creabamos una migración para agregar una columna en una tabla existente. Por ejemplo:

Archivo: *db/migrate/002_add_cost.rb*

```
class AddCost < ActiveRecord::Migration
  def self.up
    add_column :items, :cost, :decimal, :precision => 6,
   :scale => 2
  end

  def self.down
    remove_column :items, :cost
```

```
    end
  end
```

Note que estamos creando una columna con **:precision => 6** y **:scale => 2**. Ahora ejecutamos **rake db:migrate** y vemos como se ve nuestra tabla en la base de datos:

Column	Type	Modifiers
id	integer	not null
descr	character varying(255)	
price	numeric(5,2)	
cost	numeric	

Vea la columna "cost" que acabamos de crear. Es un **numeric** común, pero debería ser como la columna "price" de arriba, más precisamente un **numeric(6,2)**. En Rails 2.1 este error no sucede más y la columa se crea correctamente.

MIME TYPES

Se arregló un bug que no permitía definir el atributo para **request.format** usando un símbolo. Ahora podemos usar el siguiente código:

```
request.format = :iphone
assert_equal :iphone, request.format
```

BUG FIXES EN CHANGE_COLUMN

Se arregló un bug en el uso del método **change_column** con **:null => true** en una columna creada usando **:null => false**. Debido a este bug no se realizó ningún cambio al usar este método.

Chapter 14

Información adicional

PROTECCIÓN CONTRA CROSS SITE SCRIPTING

En Rails 2.0 el archivo *application.rb* era así:

```
class ApplicationController < ActionController::Base
  helper :all

  protect_from_forgery
end
```

Vea la llamada al método **protect_from_forgery**.

¿Escuchó hablar sobre Cross Site Scripting? Este es el nombre de un fallo de seguridad que se puede encontrar en la mayoría de los sitios y aplicaciones web que permiten a gente mala (estoy hablando de adolescentes que

no tienen nada que hacer y tampoco tienen vida social) alterar el contenido de las páginas web, para llevar acabo 'fishing attacks', tomando el control del navegador a través de código javascript y en la mayoría de las veces forzando el usuario a ejecutar el comando que ellos quieran. Este último tipo de ataque se llama 'cross-site request forgery'.

Cross Site Request Forgery es un tipo de ataque fuerza al usuario a ejecutar comandos si que este lo sepa. Y con el incremento del uso de ajax, las cosas empeoran.

Actualmente, este método es útil para asegurar que todos los formularios de tu aplicación están recibiendo datos que vienen de la misma aplicación, y no desde un link perdido en otro sitio. Esto se hace inlcuyendo un token basado en la sesión en todos los formularios y peticiones ajax generadas por Rails, y luego verificando la autenticidad de este token en el controlador.

Recuerder que las peticiones GET no están protegidas. Pero esto no es un problema si usamos sólamente para obtener datos, y nunca para alterar o guardar nada en nuestra base de datos.

Si ud. quiere aprender más acerca de CSRF (Cross-Site Request Forgery) vaya a la siguiente dirección:

- http://www.nomedojogo.com/2008/01/14/como-um-garoto-chamado-samy-pode-derrubar-seu-site/
  isc.sans.org/diary.html?storyid=1750
- http://www.nomedojogo.com/2008/01/14/como-um-garoto-chamado-samy-pode-derrubar-seu-site/
  isc.sans.org/diary.html?storyid=1750

Pero recuerde que esta no es una solución definitiva al problema, o como decimos comunmente, no es una bala de plata.

USE METHOD_MISSING, SIN DEJAR CABOS SUELTOS

Debido a la naturaleza dinámica de Ruby, el método **respond_to?** es crucial. ¿Cuántas veces necesitamos verificar si un método existe en el objeto que estamos manipulando, o si un objeto es lo que estamos esperando (**is_a?**)?

Sin embargo hay algo muy importante que mucha gente olvida. Mire por ejemplo esta clase que usa el método **method_missing**:

```
class Perro
  def method_missing(method, *args, &block)
    if method.to_s =~ /^ladrar/
      puts "woofwoof!"
    else
      super
    end
  end
end

rex = Perro.new
rex.ladrar #=> woofwof!
rex.ladrar! #=> woofwoof!
rex.ladrar_y_correr #=> woofwoof!
```

Creo que ud conoce **method_missing**, o no? En el ejemplo de arriba estoy creando una instacia de la clase **Perro** y llamando a los métodos **ladrar**, **ladrar!** y **ladrarycorrer** que no existen. Entonces el método **method_missing** es invocado, donde uso una simple expresión regular para retornar "woofwoof!", en caso que el nombre del método comience con ladrar.

Pero veamos que pasa cuando intento usar el método **respond_to?**:

```
rex.respond_to? :ladrar #=> false
rex.ladrar #=> woofwoof!
```

Retorna false, y tiene mucho sentido dado que el método no existe realmente. Entonces es mi responsabilidad cambiar el método **respond_to?** para que funcione correctamente con esta regla especial. Voy a modificar mi clase de la siguiente forma:

```
class Perro
  METODO_LADRAR = /^ladrar/

  def respond_to?(method)
    return true if method.to_s =~ METODO_LADRAR
    super
  end

  def method_missing(method, *args, &block)
    if method.to_s =~ METODO_LADRAR
      puts "woofwoof!"
    else
      super
    end
  end
end

rex = Dog.new
rex.respond_to?(:ladrar) #=> true
rex.ladrar #=> woofwoof!
```

Ahora sí! Este es un error muy común que he visto en algunos códigos, inclusive en el propio Rails. Intente ejecutar el método **respond_to?** para verificar la existencia de métodos como **find_by_name**, por ejemplo.

Ruby es un lenguaje impresionante y altamente flexible, pero si no prestamos atención podemos dejar cabos sueltos como este.

Por supuesto que en Rails 2.1 este problema ya se solucionó. Podemos usar **respond_to?** para verificar la existencia de métodos como **find_by_algo**.

POSTGRESQL

En Rails 2.0, el adaptador para **PostgreSQL** tenía soporte solamente para las versiones 8.1 hasta 8.3. Se agregó soporte para las versiones 7.4 hasta la 8.3.

Chapter 15

CHANGELOG

ACTIONMAILER

* Fixed that a return-path header would be ignored #7572 [joost]

* Less verbose mail logging: just recipients for :info log level; the whole email for :debug only. #8000 [iaddict, Tarmo Tänav]

* Updated TMail to version 1.2.1 [raasdnil]

* Fixed that you don't have to call super in ActionMailer::TestCase#setup #10406 [jamesgolick]

ACTIONPACK

* InstanceTag#default_time_from_options overflows to DateTime [Geoff Buesing]

* Fixed that forgery protection can be used without session tracking (Peter Jones) [#139]

* Added session(:on) to turn session management back on in a controller subclass if the superclass turned it off (Peter Jones) [#136]

* Change the request forgery protection to go by Content-Type instead of request.format so that you can't bypass it by POSTing to "#{request.uri}.xml" [rick] * InstanceTag#default_time_from_options with hash args uses Time.current as default; respects hash settings when time falls in system local spring DST gap [Geoff Buesing]

* select_date defaults to Time.zone.today when config.time_zone is set [Geoff Buesing]

* Fixed that TextHelper#text_field would corrypt when raw HTML was used as the value (mchenryc, Kevin Glowacz) [#80]

* Added ActionController::TestCase#rescue_action_in_public! to control whether the action under test should use the regular rescue_action path instead of simply raising the exception inline (great for error testing) [DHH]

* Reduce number of instance variables being copied from controller to view. [Pratik]

* select_datetime and select_time default to Time.zone.now when config.time_zone is set [Geoff Buesing]

* datetime_select defaults to Time.zone.now when config.time_zone is set [Geoff Buesing]

* Remove ActionController::Base#view_controller_internals flag. [Pratik]

* Add conditional options to caches_page method. [Paul Horsfall]

* Move missing template logic to ActionView. [Pratik]

* Introduce ActionView::InlineTemplate class. [Pratik]

* Automatically parse posted JSON content for Mime::JSON requests. [rick]

```
POST /posts
    {"post": {"title": "Breaking News"}}

def create
    @post = Post.create params[:post]
    # ...
end
```

* add json_escape ERB util to escape html entities in json strings that are output in HTML pages. [rick]

* Provide a helper proxy to access helper methods from outside views. Closes #10839 [Josh Peek] e.g. ApplicationController.helpers.simple_format(text)

* Improve documentation. [Xavier Noria, leethal, jerome]

* Ensure RJS redirect_to doesn't html-escapes string argument. Closes #8546 [josh, eventualbuddha, Pratik]

* Support render :partial => collection of heterogeneous elements. #11491 [Zach Dennis]

* Avoid remote_ip spoofing. [Brian Candler]

* Added support for regexp flags like ignoring case in the :requirements part of routes declarations #11421 [NeilW]

* Fixed that ActionController::Base#read_multipart would fail if boundary was exactly 10240 bytes #10886 [ariejan]

* Fixed HTML::Tokenizer (used in sanitize helper) didn't handle unclosed CDATA tags #10071 [esad, packagethief]

* Improve documentation. [Radar, Jan De Poorter, chuyeow, xaviershay, danger, miloops, Xavier Noria, Sunny Ripert]

* Fixed that FormHelper#radio_button would produce invalid ids #11298 [harlancrystal]

* Added :confirm option to submit_tag #11415 [miloops]

* Fixed NumberHelper#number_with_precision to properly round in a way that works equally on Mac, Windows, Linux (closes #11409, #8275, #10090, #8027) [zhangyuanyi]

* Allow the #simple_format text_helper to take an html_options hash for each paragraph. #2448 [Francois Beausoleil, thechrisoshow]

* Fix regression from filter refactoring where re-adding a skipped filter resulted in it being called twice. [rick]

* Refactor filters to use Active Support callbacks. #11235 [Josh Peek]

* Fixed that polymorphic routes would modify the input array #11363 [thomas.lee]

* Added :format option to NumberHelper#number_to_currency to enable better localization support #11149 [lylo]

* Fixed that TextHelper#excerpt would include one character too many #11268 [Irfy]

* Fix more obscure nested parameter hash parsing bug. #10797 [thomas.lee]

* Added ActionView::Helpers::register_javascript/stylesheet_expansion to make it easier for plugin developers to inject multiple assets. #10350 [lotswholetime]

* Fix nested parameter hash parsing bug. #10797 [thomas.lee]

* Allow using named routes in ActionController::TestCase before any request has been made. Closes #11273 [alloy]

* Fixed that sweepers defined by cache_sweeper will be added regardless of the perform_caching setting. Instead, control whether the sweeper should be run with the perform_caching setting. This makes testing easier when you want to turn perform_caching on/off [DHH]

* Make MimeResponds::Responder#any work without explicit types. Closes #11140 [jaw6]

* Better error message for type conflicts when parsing params. Closes #7962 [spicycode, matt]

* Remove unused ActionController::Base.template_class. Closes #10787 [Pratik]

* Moved template handlers related code from ActionView::Base to ActionView::Template. [Pratik]

* Tests for div_for and content_tag_for helpers. Closes #11223 [thechrisoshow]

* Allow file uploads in Integration Tests. Closes #11091 [RubyRedRick]

* Refactor partial rendering into a PartialTemplate class. [Pratik]

* Added that requests with JavaScript as the priority mime type in the accept header and no format extension in the parameters will be treated as though their format was :js when it comes to determining which template to render. This makes it possible for JS requests to automatically render action.js.rjs files without an explicit respond_to block [DHH]

* Tests for distance_of_time_in_words with TimeWithZone instances. Closes #10914 [ernesto.jimenez]

* Remove support for multivalued (e.g., '&'-delimited) cookies. [Jamis Buck]

* Fix problem with render :partial collections, records, and locals. #11057 [lotswholetime]

* Added support for naming concrete classes in sweeper declarations [DHH]

* Remove ERB trim variables from trace template in case ActionView::Base.erb_trim_mode is changed in the application. #10098 [tpope, kampers]

* Fix typo in form_helper documentation. #10650 [xaviershay, kampers]

* Fix bug with setting Request#format= after the getter has cached the value. #10889 [cch1]

* Correct inconsistencies in RequestForgeryProtection docs. #11032 [mislav]

* Introduce a Template class to ActionView. #11024 [lifofifo]

* Introduce the :index option for form_for and fields_for to simplify multi-model forms (see http://railscasts.com/episodes/75). #9883 [rmm5t]

* Introduce map.resources :cards, :as => 'tarjetas' to use a custom resource name in the URL: cards_path == '/tarjetas'. #10578 [blj]

* TestSession supports indifferent access. #7372 [tamc, Arsen7, mhackett, julik, jean.helou]

* Make assert_routing aware of the HTTP method used. #8039 [mpalmer] e.g. assert_routing({ :method => 'put', :path => '/product/321' }, { :controller => "product", :action => "update", :id => "321" })

* Make map.root accept a single symbol as an argument to declare an alias. #10818 [bscofield]

e.g. map.dashboard '/dashboard', :controller=>'dashboard'

```
map.root      :dashboard
```

* Handle corner case with image_tag when passed 'messed up' image names. #9018 [duncanbeevers, mpalmer]

* Add label_tag helper for generating elements. #10802 [DefV]

* Introduce TemplateFinder to handle view paths and lookups. #10800 [Pratik Naik]

* Performance: optimize route recognition. Large speedup for apps with many resource routes. #10835 [oleganza]

* Make render :partial recognise form builders and use the _form partial. #10814 [djanowski]

* Allow users to declare other namespaces when using the atom feed helpers. #10304 [david.calavera]

* Introduce send_file :x_sendfile => true to send an X-Sendfile response header. [Jeremy Kemper]

* Fixed ActionView::Helpers::ActiveRecordHelper::form for when protect_from_forgery is used #10739 [jeremyevans]

* Provide nicer access to HTTP Headers. Instead of request.env["HTTP_REFERRER"] you can now use request.headers["Referrer"]. [Koz]

* UrlWriter respects relative_url_root. #10748 [Cheah Chu Yeow]

* The asset_host block takes the controller request as an optional second argument. Example: use a single asset host for SSL requests. #10549 [Cheah Chu Yeow, Peter B, Tom Taylor]

* Support render :text => nil. #6684 [tjennings, PotatoSalad, Cheah Chu Yeow]

* assert_response failures include the exception message. #10688 [Seth Rasmussen]

* All fragment cache keys are now by default prefixed with the "views/" namespace [DHH]

* Moved the caching stores from ActionController::Caching::Fragments::* to ActiveSupport::Cache::*. If you're explicitly referring to a store, like ActionController::Caching::Fragments::MemoryStore, you need to update that reference with ActiveSupport::Cache::MemoryStore [DHH]

* Deprecated ActionController::Base.fragment_cache_store for ActionController::Base.cache_store [DHH]

* Made fragment caching in views work for rjs and builder as well #6642 [zsombor]

* Fixed rendering of partials with layout when done from site layout #9209 [antramm]

* Fix atom_feed_helper to comply with the atom spec. Closes #10672 [xaviershay]

```
The tags created do not contain a date (http://feedvalidator.org/docs/error/InvalidTAG.html)
IDs are not guaranteed unique
```

```
A default self link was not provided, contrary to the documentation
NOTE:  This changes tags for existing atom entries, but at least they validate now.
```

* Correct indentation in tests. Closes #10671 [l.guidi]

* Fix that auto_link looks for =’s in url paths (Amazon urls have them). Closes #10640 [bgreenlee]

* Ensure that test case setup is run even if overridden. #10382 [Josh Peek]

* Fix HTML Sanitizer to allow trailing spaces in CSS style attributes. Closes #10566 [wesley.moxam]

* Add :default option to time_zone_select. #10590 [Matt Aimonetti]

ACTIVERECORD

* Add ActiveRecord::Base.sti_name that checks ActiveRecord::Base#store_full_sti_class? and returns either the full or demodulized name. [rick]

* Add first/last methods to associations/named_scope. Resolved #226. [Ryan Bates]

* Added SQL escaping for :limit and :offset #288 [Aaron Bedra, Steven Bristol, Jonathan Wiess]

* Added first/last methods to associations/named_scope. Resolved #226. [Ryan Bates]

* Ensure hm:t preloading honours reflection options. Resolves #137. [Frederick Cheung]

* Added protection against duplicate migration names (Aslak Hellesøy) [#112]

* Base#instantiate_time_object: eliminate check for Time.zone, since we can assume this is set if time_zone_aware_attributes is set to true [Geoff Buesing]

* Time zone aware attribute methods use Time.zone.parse instead of #to_time for String arguments, so that offset information in String is respected. Resolves #105. [Scott Fleckenstein, Geoff Buesing]

* Added change_table for migrations (Jeff Dean) [#71]. Example:

```
change_table :videos do |t|
  t.timestamps                        # adds created_at, updated_at
  t.belongs_to :goat                  # adds goat_id integer
  t.string :name, :email, :limit => 20  # adds name and email both with a 20 char limit
  t.remove :name, :email              # removes the name and email columns
end
```

* Fixed has_many :through .create with no parameters caused a "can't dup NilClass" error (Steven Soroka) [#85]

* Added block-setting of attributes for Base.create like Base.new already has (Adam Meehan) [#39]

* Fixed that pessimistic locking you reference the quoted table name (Josh Susser) [#67]

* Fixed that change_column should be able to use :null => true on a field that formerly had false [Nate Wiger] [#26]

* Added that the MySQL adapter should map integer to either smallint, int, or bigint depending on the :limit just like PostgreSQL [DHH]

* Change validates_uniqueness_of :case_sensitive option default back to true (from [9160]). Love your database columns, don't LOWER them. [rick]

* Add support for interleaving migrations by storing which migrations have run in the new schema_migrations table. Closes #11493 [jordi]

* ActiveRecord::Base#sum defaults to 0 if no rows are returned. Closes #11550 [kamal]

* Ensure that respond_to? considers dynamic finder methods. Closes #11538. [floehopper]

* Ensure that save on parent object fails for invalid has_one association. Closes #10518. [Pratik]

* Remove duplicate code from associations. [Pratik]

* Refactor HasManyThroughAssociation to inherit from HasManyAssociation. Association callbacks and _ids= now work with hm:t. #11516 [rubyruy]

* Ensure HABTM#create and HABTM#build do not load entire association. [Pratik]

* Improve documentation. [Xavier Noria, Jack Danger Canty, leethal]

* Tweak ActiveRecord::Base#to_json to include a root value in the returned hash: {"post": {"title": ...}} [rick]

```
Post.find(1).to_json # => {"title": ...}
config.active_record.include_root_in_json = true
Post.find(1).to_json # => {"post": {"title": ...}}
```

* Add efficient #include? to AssociationCollection (for has_many/has_many :through/habtm). [stopdropandrew]

* PostgreSQL: create_ and drop_database support. #9042 [ez, pedz, nicksieger]

* Ensure that validates_uniqueness_of works with with_scope. Closes #9235. [nik.wakelin, cavalle]

* Partial updates include only unsaved attributes. Off by default; set YourClass.partial_updates = true to enable. [Jeremy Kemper]

* Removing unnecessary uses_tzinfo helper from tests, given that TZInfo is now bundled [Geoff Buesing]

* Fixed that validates_size_of :within works in associations #11295, #10019 [cavalle]

* Track changes to unsaved attributes. [Jeremy Kemper]

* Switched to UTC-timebased version numbers for migrations and the schema. This will as good as eliminate the problem of multiple migrations getting the same version assigned in different branches. Also added rake db:migrate:up/down to apply individual migrations that may need to be run when you merge branches #11458 [jbarnette]

* Fixed that has_many :through would ignore the hash conditions #11447 [miloops]

* Fix issue where the :uniq option of a has_many :through association is ignored when find(:all) is called. Closes #9407 [cavalle]

* Fix duplicate table alias error when including an association with a has_many :through association on the same join table. Closes #7310 [cavalle]

* More efficient association preloading code that compacts a through_records array in a central location. Closes #11427 [danger]

* Improve documentation. [Radar, Jan De Poorter, chuyeow, xaviershay, danger, miloops, Xavier Noria, Sunny Ripert]

* Fixed that ActiveRecord#Base.find_or_create/initialize would not honor attr_protected/accessible when used with a hash #11422 [miloops]

* Added ActiveRecord#Base.all/first/last as aliases for find(:all/:first/:last) #11413 [nkallen, thechrisoshow]

* Merge the has_finder gem, renamed as 'named_scope'. #11404 [nkallen]

```
class Article < ActiveRecord::Base
    named_scope :published, :conditions => {:published => true}
    named_scope :popular, :conditions => ...
end

Article.published.paginate(:page => 1)
Article.published.popular.count
Article.popular.find(:first)
Article.popular.find(:all, :conditions => {...})
```

See http://pivots.pivotallabs.com/users/nick/blog/articles/284-hasfinder-it-s-now-easier-than-ever-to-create-complex-re-usable-sql-queries

* Add has_one :through support. #4756 [thechrisoshow]

* Migrations: create_table supports primary_key_prefix_type. #10314 [student, thechrisoshow]

* Added logging for dependency load errors with fixtures #11056 [stuthulhu]

* Time zone aware attributes use Time#in_time_zone [Geoff Buesing]

* Fixed that scoped joins would not always be respected #6821 [Theory/Danger]

* Ensure that ActiveRecord::Calculations disambiguates field names with the table name. #11027 [cavalle]

* Added add/remove_timestamps to the schema statements for adding the created_at/updated_at columns on existing tables #11129 [jramirez]

* Added ActiveRecord::Base.find(:last) #11338 [miloops]

* test_native_types expects DateTime.local_offset instead of DateTime.now.offset; fixes test breakage due to dst transition [Geoff Buesing]

* Add :readonly option to HasManyThrough associations. #11156 [miloops]

* Improve performance on :include/:conditions/:limit queries by selectively joining in the pre-query. #9560 [dasil003]

* Perf fix: Avoid the use of named block arguments. Closes #11109 [adymo]

* PostgreSQL: support server versions 7.4 through 8.0 and the ruby-pg driver. #11127 [jdavis]

* Ensure association preloading doesn't break when an association returns nil. ##11145 [GMFlash]

* Make dynamic finders respect the :include on HasManyThrough associations. #10998. [cpytel]

* Base#instantiate_time_object only uses Time.zone when Base.time_zone_aware_attributes is true; leverages Time#time_with_datetime_fallback for readability [Geoff Buesing]

* Refactor ConnectionAdapters::Column.new_time: leverage DateTime failover behavior of Time#time_with_datetime_fallback [Geoff Buesing]

* Improve associations performance by using symbol callbacks instead of string callbacks. #11108 [adymo]

* Optimise the BigDecimal conversion code. #11110 [adymo]

* Introduce the :readonly option to all associations. Records from the association cannot be saved. #11084 [miloops]

* Multiparameter attributes for time columns fail over to DateTime when out of range of Time [Geoff Buesing]

* Base#instantiate_time_object uses Time.zone.local() [Geoff Buesing]

* Add timezone-aware attribute readers and writers. #10982 [Geoff Buesing]

* Instantiating time objects in multiparameter attributes uses Time.zone if available. #10982 [rick]

* Add note about how ActiveRecord::Observer classes are initialized in a Rails app. #10980 [fxn]

* MySQL: omit text/blob defaults from the schema instead of using an empty string. #10963 [mdeiters]

* belongs_to supports :dependent => :destroy and :delete. #10592 [Jonathan Viney]

* Introduce preload query strategy for eager :includes. #9640 [Frederick Cheung, Aleksey Kondratenko, codafoo]

* Support aggregations in finder conditions. #10572 [Ryan Kinderman]

* Organize and clean up the Active Record test suite. #10742 [John Barnette]

* Ensure that modifying has_and_belongs_to_many actions clear the query cache. Closes #10840 [john.andrews]

* Fix issue where Table#references doesn't pass a :null option to a *_type attribute for polymorphic associations. Closes #10753 [railsjitsu]

* Fixtures: removed support for the ancient pre-YAML file format. #10736 [John Barnette]

* More thoroughly quote table names. #10698 [dimdenis, lotswholetime, Jeremy Kemper]

* update_all ignores scoped :order and :limit, so post.comments.update_all doesn't try to include the comment order in the update statement. #10686 [Brendan Ribera]

* Added ActiveRecord::Base.cache_key to make it easier to cache Active Records in combination with the new ActiveSupport::Cache::* libraries [DHH]

* Make sure CSV fixtures are compatible with ruby 1.9's new csv implementation. [JEG2]

* Added by parameter to increment, decrement, and their bang varieties so you can do player1.increment!(:points, 5) #10542 [Sam]

* Optimize ActiveRecord::Base#exists? to use #select_all instead of #find. Closes #10605 [jamesh, fcheung, protocool]

* Don't unnecessarily load has_many associations in after_update callbacks. Closes #6822 [stopdropandrew, canadaduane]

* Eager belongs_to :include infers the foreign key from the association name rather than the class name. #10517 [Jonathan Viney]

* SQLite: fix rename_ and remove_column for columns with unique indexes. #10576 [Brandon Keepers]

* Ruby 1.9 compatibility. #10655 [Jeremy Kemper, Dirkjan Bussink]

ACTIVERESOURCE

2.1.0 (May 31st, 2008)*

* Fixed response logging to use length instead of the entire thing (seangeo) [#27]

* Fixed that to_param should be used and honored instead of hardcoding the id #11406 [gspiers]

* Improve documentation. [Radar, Jan De Poorter, chuyeow, xaviershay, danger, miloops, Xavier Noria, Sunny Ripert]

* Use HEAD instead of GET in exists? [bscofield]

* Fix small documentation typo. Closes #10670 [l.guidi]

* find_or_create_resource_for handles module nesting. #10646 [xavier]

* Allow setting ActiveResource::Base#format before #site. [rick]

* Support agnostic formats when calling custom methods. Closes #10635 [joerichsen]

* Document custom methods. #10589 [Cheah Chu Yeow]

* Ruby 1.9 compatibility. [Jeremy Kemper]

ACTIVESUPPORT

* TimeZone#to_s shows offset as GMT instead of UTC, because GMT will be more familiar to end users (see time zone selects used by Windows OS, google.com and yahoo.com.) Reverts [8370] [Geoff Buesing]

* Hash.from_xml: datetime xml types overflow to Ruby DateTime class when out of range of Time. Adding tests for utc offsets [Geoff Buesing]

* TimeWithZone #+ and #- : ensure overflow to DateTime with Numeric arg [Geoff Buesing]

* Time#to_json: don't convert to utc before encoding. References #175 [Geoff Buesing]

* Remove unused JSON::RESERVED_WORDS, JSON.valid_identifier? and JSON.reserved_word? methods. Resolves #164. [Cheah Chu Yeow]

* Adding Date.current, which returns Time.zone.today if config.time_zone is set; otherwise returns Date.today [Geoff Buesing]

* TimeWithZone: date part getter methods (#year #mon #day etc) are defined on class; no longer relying on method_missing [Geoff Buesing]

* Time.zone.parse return nil for strings with no date information [Geoff Buesing]

* Time.zone.parse respects offset information in string. Resolves #105. [Scott Fleckenstein, Geoff Buesing]

* Added Ruby 1.8 implementation of Process.daemon

* Duration #since and #ago with no argument (e.g., 5.days.ago) return TimeWithZone when config.time_zone is set. Introducing Time.current, which returns Time.zone.now if config.time_zone is set, otherwise just returns Time.now [Geoff Buesing]

* Time#since behaves correctly when passed a Duration. Closes #11527 [kemiller]

* Add #getutc alias for DateTime#utc [Geoff Buesing]

* Refactor TimeWithZone: don't send #since, #ago, #+, #-, #advance through method_missing [Geoff Buesing]

* TimeWithZone respects config.active_support.use_standard_json_time_format [Geoff Buesing]

* Add config.active_support.escape_html_entities_in_json to allow disabling of html entity escaping. [rick]

* Improve documentation. [Xavier Noria]

* Modified ActiveSupport::Callbacks::Callback#call to accept multiple arguments.

* Time #yesterday and #tomorrow behave correctly crossing DST boundary. Closes #7399 [sblackstone]

* TimeWithZone: Adding tests for dst and leap day edge cases when advancing time [Geoff Buesing]

* TimeWithZone#method_missing: send to utc to advance with dst correctness, otherwise send to time. Adding tests for time calculations methods [Geoff Buesing]

* Add config.active_support.use_standard_json_time_format setting so that Times and Dates export to ISO 8601 dates. [rick]

* TZInfo: Removing unneeded TimezoneProxy class [Geoff Buesing]

* TZInfo: Removing unneeded TimezoneIndexDefinition, since we're not including Indexes::Timezones [Geoff Buesing]

* Removing unnecessary uses_tzinfo helper from tests, given that TZInfo is now bundled [Geoff Buesing]

* Bundling abbreviated version of TZInfo gem 0.3.8: only the classes and zone definitions required to support Rails time zone features are included. If a recent version of the full TZInfo gem is installed, this will take precedence over the bundled version [Geoff Buesing]

* TimeWithZone#marshal_load does zone lookup via Time.get_zone, so that tzinfo/Olson identifiers are handled [Geoff Buesing]

* Time.zone= accepts TZInfo::Timezone instances and Olson identifiers; wraps result in TimeZone instance [Geoff Buesing]

* TimeWithZone time conversions don't need to be wrapped in TimeOrDateTime, because TZInfo does this internally [Geoff Buesing]

* TimeWithZone#usec returns 0 instead of error when DateTime is wrapped [Geoff Buesing]

* Improve documentation. [Radar, Jan De Poorter, chuyeow, xaviershay, danger, miloops, Xavier Noria, Sunny Ripert]

* Ensure that TimeWithZone#to_yaml works when passed a YAML::Emitter. [rick]

* Ensure correct TimeWithZone#to_date [Geoff Buesing]

* Make TimeWithZone work with tzinfo 0.2.x: use TZInfo::Timezone#zone_identifier alias for #abbreviation, silence warnings on tests. Raise LoadError when TZInfo version is < 0.2 by sniffing for TZInfo::TimeOrDateTime constant. Move all tzinfo-dependent TimeZone tests into uses_tzinfo block [Geoff Buesing]

* Time, DateTime and TimeWithZone #in_time_zone defaults to Time.zone. Removing now unneeded #in_current_time_zone [Geoff Buesing]

* TZInfo caches Timezone instances in its own internal hash cache, so TimeZone::MAPPING doesn't need to cache them as well [Geoff Buesing]

* Adding TimeZone#parse [Geoff Buesing]

* Adding TimeZone#at and DateTime#to_f [Geoff Buesing]

* TimeWithZone responds to Ruby 1.9 weekday-named query methods [Geoff Buesing]

* TimeWithZone caches TZInfo::TimezonePeriod used for time conversion so that it can be reused, and enforces DST rules correctly when instance is created from a local time [Geoff Buesing]

* Fixed that BufferedLogger should create its own directory if one doesn't already exist #11285 [lotswholetime]

* Fix Numeric time tests broken by DST change by anchoring them to fixed times instead of Time.now. Anchor TimeZone#now DST test to time specified with Time.at instead of Time.local to work around platform differences with Time.local and DST representation [Geoff Buesing]

* Removing unneeded #change_time_zone method from Time, DateTime and TimeWithZone [Geoff Buesing]

* TimeZone #local and #now correctly enforce DST rules [Geoff Buesing]

* TimeWithZone instances correctly enforce DST rules. Adding TimeZone#period_for_utc [Geoff Buesing]

* test_time_with_datetime_fallback expects DateTime.local_offset instead of DateTime.now.offset [Geoff Buesing]

* Adding TimeWithZone #marshal_dump and #marshal_load [Geoff Buesing]

* Add OrderedHash#to_hash [josh]

* Adding Time#end_of_day, _quarter, _week, and _year. #9312 [Juanjo Bazan, Tarmo Tänav, BigTitus]

* Adding TimeWithZone#between? [Geoff Buesing]

* Time.=== returns true for TimeWithZone instances [Geoff Buesing]

* TimeWithZone #+ and #- behave consistently with numeric arguments regardless of whether wrapped time is a Time or DateTime; consistenty answers false to #acts_like?(:date) [Geoff Buesing]

* Add String#squish and String#squish! to remove consecutive chunks of whitespace. #11123 [jordi, Henrik N]

* Serialize BigDecimals as Floats when using to_yaml. #8746 [ernesto.jimenez]

* Adding TimeWithZone #to_yaml, #to_datetime, #eql? and method aliases for duck-typing compatibility with Time [Geoff Buesing]

* TimeWithZone #in_time_zone returns +self+ if zone argument is the same as #time_zone [Geoff Buesing]

* Adding TimeWithZone #to_a, #to_f, #to_i, #httpdate, #rfc2822 [Geoff Buesing]

* Pruning unneeded TimeWithZone#change_time_zone_to_current [Geoff Buesing]

* Time#zone=, #in_time_zone and #change_time_zone accept a Duration [Geoff Buesing]

* Time#in_time_zone handles Time.local instances correctly [Geoff Buesing]

* Pruning unneeded Time#change_time_zone_to_current. Enhanced docs to #change_time_zone to explain the difference between this method and #in_time_zone [Geoff Buesing]

* TimeZone#new method renamed #local; when used with Time.zone, constructor now reads: Time.zone.local() [Geoff Buesing]

* Added Base64.encode64s to encode values in base64 without the newlines. This makes the values immediately usable as URL parameters or memcache keys without further processing [DHH]

* Remove :nodoc: entries around the ActiveSupport test/unit assertions. #10946 [dancroak, jamesh]

* Add Time.zone_default accessor for setting the default time zone. Rails::Configuration.time_zone sets this. #10982 [Geoff Buesing]

* cache.fetch(key, :force => true) to force a cache miss. [Jeremy Kemper]

* Support retrieving TimeZones with a Duration. TimeZone[-28800] == TimeZone[-480.minutes]. [rick]

* TimeWithZone#- added, so that #- can handle a Time or TimeWithZone argument correctly [Geoff Buesing]

* with_timezone test helper renamed with_env_tz, to distinguish between setting ENV['TZ'] and setting Time.zone in tests [Geoff Buesing]

* Time#- coerces TimeWithZone argument to a Time instance so that difference in seconds can be calculated. Closes #10914 [Geoff Buesing, yyyc514]

* Adding UTC zone to TimeZone; TimeWithZone no longer has to fake UTC zone with nil [Geoff Buesing]

* Time.get_zone refactored to private method, given that the encapsulated logic is only useful internally [Geoff Buesing]

* Time.zone uses thread-local variable for thread safety. Adding Time.use_zone, for overriding Time.zone locally inside a block. Removing unneeded Time.zone_reset! [Geoff Buesing]

* TimeZone#to_s uses UTC rather than GMT; reapplying change that was undone in [8679]. #1689 [Cheah Chu Yeow]

* Time.days_in_month defaults to current year if no year is supplied as argument #10799 [Radar], uses Date.gregorian_leap? to determine leap year, and uses constant lookup to determine days in month [Geoff Buesing]

* Adding Time and DateTime #compare_with_coercion, which layers behavior on #<=> so that any combination of Time, DateTime and ActiveSupport::TimeWithZone instances can be chronologically compared [Geoff Buesing]

* TimeZone#now returns an ActiveSupport::TimeWithZone [Geoff Buesing]

* Time #in_current_time_zone and #change_time_zone_to_current return self when Time.zone is nil [Geoff Buesing]

* Remove unneeded #to_datetime_default_s alias for DateTime#to_s, given that we inherit a #to_default_s from Date that does exactly the same thing [Geoff Buesing]

* Refactor Time and DateTime #to_formatted_s: use ternary instead of nested if/else [Geoff Buesing]

* Adding Time and DateTime #formatted_offset, for outputting +HH:MM utc offset strings with cross-platform consistency [Geoff Buesing]

* Adding alternate_utc_string option to TimeZone#formatted_offset. Removing unneeded TimeZone#offset. [Geoff Buesing]

* Introduce ActiveSupport::TimeWithZone, for wrapping Time instances with a TimeZone. Introduce instance methods to Time for creating TimeWithZone instances, and class methods for managing a global time zone. [Geoff Buesing]

* Replace non-dst-aware TimeZone class with dst-aware class from tzinfo_timezone plugin. TimeZone#adjust and #unadjust are no longer available; tzinfo gem must now be present in order to perform time zone calculations, via #local_to_utc and #utc_to_local methods. [Geoff Buesing]

* Extract ActiveSupport::Callbacks from Active Record, test case setup and teardown, and ActionController::Dispatcher. #10727 [Josh Peek]

* Introducing DateTime #utc, #utc? and #utc_offset, for duck-typing compatibility with Time. Closes #10002 [Geoff Buesing]

* Time#to_json uses Numeric#to_utc_offset_s to output a cross-platform-consistent representation without having to convert to DateTime. References #9750 [Geoff Buesing]

* Refactor number-to-HH:MM-string conversion logic from TimeZone#formatted_offset to a reusable Numeric#to_utc_offset_s method. [Geoff Buesing]

* Continue evolution toward ActiveSupport::TestCase. #10679 [Josh Peek]

* TestCase: introduce declared setup and teardown callbacks. Pass a list of methods and an optional block to call before setup or after teardown. Setup callbacks are run in the order declared; teardown callbacks are run in reverse. [Jeremy Kemper]

* Added ActiveSupport::Gzip.decompress/compress(source) as an easy wrapper for Zlib [Tobias Luetke]

* Included MemCache-Client to make the improved ActiveSupport::Cache::MemCacheStore work out of the box [Bob Cottrell, Eric Hodel]

* Added ActiveSupport::Cache::* framework as an extraction from ActionController::Caching::Fragments::* [DHH]

* Fixed String#titleize to work for strings with 's too #10571 [trek]

* Changed the implementation of Enumerable#group_by to use a double array approach instead of a hash such that the insert order is honored [DHH/Marcel]

* remove multiple enumerations from ActiveSupport::JSON#convert_json_to_yaml when dealing with date/time values. [rick]

* Hash#symbolize_keys skips keys that can't be symbolized. #10500 [Brad Greenlee]

* Ruby 1.9 compatibility. #1689, #10466, #10468, #10554, #10594, #10632 [Cheah Chu Yeow, Pratik Naik, Jeremy Kemper, Dirkjan Bussink, fxn]

* TimeZone#to_s uses UTC rather than GMT. #1689 [Cheah Chu Yeow]

* Refactor of Hash#symbolize_keys! to use Hash#replace. Closes #10420 [ReinH]

* Fix HashWithIndifferentAccess#to_options! so it doesn't clear the options hash. Closes #10419 [ReinH]

RAILTIES

* script/dbconsole fires up the command-line database client. #102 [Steve Purcell]

* Fix bug where plugin init.rb files from frozen gem specs weren't being run. (pjb3) [#122 state:resolved]

* Made the location of the routes file configurable with config.routes_configuration_file (Scott Fleckenstein) [#88]

* Rails Edge info returns the latest git commit hash [Francesc Esplugas]

* Added Rails.public_path to control where HTML and assets are expected to be loaded from (defaults to Rails.root + "/public") #11581 [nicksieger]

* rake time:zones:local finds correct base utc offset for zones in the Southern Hemisphere [Geoff Buesing]

* Don't require rails/gem_builder during rails initialization, it's only needed for the gems:build task. [rick]

* script/performance/profiler compatibility with the ruby-prof >= 0.5.0. Closes #9176. [Catfish]

* Flesh out rake gems:unpack to unpack all gems, and add rake gems:build for native extensions. #11513 [ddollar]

```
rake gems:unpack              # unpacks all gems
rake gems:unpack GEM=mygem    # unpacks only the gem 'mygem'

rake gems:build               # builds all unpacked gems
rake gems:build GEM=mygem     # builds only the gem 'mygem'
```

* Add config.active_support for future configuration options. Also, add more new Rails 3 config settings to new_rails_defaults.rb [rick]

* Add Rails.logger, Rails.root, Rails.env and Rails.cache shortcuts for RAILS_* constants [pratik]

* Allow files in plugins to be reloaded like the rest of the application. [rick]

Enables or disables plugin reloading.

```
config.reload\_plugins = true
```

You can get around this setting per plugin.

If #reload_plugins? == false (DEFAULT), add this to your plugin's init.rb to make it reloadable:

```
Dependencies.load_once_paths.delete lib\_path
```

If #reload_plugins? == true, add this to your plugin's init.rb to only load it once:

```
Dependencies.load_once_paths << lib_path
```

* Small tweak to allow plugins to specify gem dependencies. [rick]

```
# OLD open\_id\_authentication plugin init.rb
require 'yadis'
```

```
require 'openid'
ActionController::Base.send :include, OpenIdAuthentication

# NEW
config.gem "ruby-openid", :lib => "openid", :version => "1.1.4"
config.gem "ruby-yadis",  :lib => "yadis",  :version => "0.3.4"

config.after_initialize do
    ActionController::Base.send :include, OpenIdAuthentication
end
```

* Added config.gem for specifying which gems are required by the application, as well as rake tasks for installing and freezing gems. [rick]

```
Rails::Initializer.run do |config|
    config.gem "bj"
    config.gem "hpricot", :version => '0.6', :source => "http://code.whytheluckystiff.net"
    config.gem "aws-s3", :lib => "aws/s3"
end

# List required gems.
    rake gems

# Install all required gems:
    rake gems:install

# Unpack specified gem to vendor/gems/gem\_name-x.x.x
    rake gems:unpack GEM=bj
```

* Removed the default .htaccess configuration as there are so many good deployment options now (kept it as an example in README) [DHH]

* config.time_zone accepts TZInfo::Timezone identifiers as well as Rails TimeZone identifiers [Geoff Buesing]

* Rails::Initializer#initialize_time_zone raises an error if value assigned to config.time_zone is not recognized. Rake time zone tasks only require ActiveSupport instead of entire environment [Geoff Buesing]

* Stop adding the antiquated test/mocks/* directories and only add them to the path if they're still there for legacy reasons [DHH]

* Added that gems can now be plugins if they include rails/init.rb #11444 [jbarnette]

* Added Plugin#about method to programmatically access the about.yml in a plugin #10979 [lazyatom]

```
plugin = Rails::Plugin.new(path\_to\_my\_plugin)
plugin.about["author"] # => "James Adam"
plugin.about["url"] # => "http://interblah.net"
```

* Improve documentation. [Radar, Jan De Poorter, chuyeow, xaviershay, danger, miloops, Xavier Noria, Sunny Ripert]

* Added config.time_zone = 'UTC' in the default environment.rb [Geoff Buesing]

* Added rake tasks time:zones:all, time:zones:us and time:zones:local for finding time zone names for config.time_zone option [Geoff Buesing]

* Add config.time_zone for configuring the default Time.zone value. #10982 [Geoff Buesing]

* Added support for installing plugins hosted at git repositories #11294 [danger]

* Fixed that script/generate would not look for plugin generators in plugin_paths #11000 [glv]

* Fixed database rake tasks to work with charset/collation and show proper error messages on failure. Closes #11301 [matt]

* Added a -e/--export to script/plugin install, uses svn export. #10847 [jon@blankpad.net)]

* Reshuffle load order so that routes and observers are initialized after plugins and app initializers. Closes #10980 [rick]

* Git support for script/generate. #10690 [ssoroka]

* Update scaffold to use labels instead of bold tags. Closes #10757 [zach-inglis-lt3]

* Resurrect WordNet synonym lookups. #10710 [tom./, matt]

* Added config.cache_store to environment options to control the default cache store (default is FileStore if tmp/cache is present, otherwise MemoryStore is used) [DHH]

* Added that rails:update is run when you do rails:freeze:edge to ensure you also get the latest JS and config files #10565 [jeff]

* SQLite: db:drop:all doesn't fail silently if the database is already open. #10577 [Cheah Chu Yeow, mrichman]

* Introduce native mongrel handler and push mutex into dispatcher. [Jeremy Kemper]

* Ruby 1.9 compatibility. #1689, #10546 [Cheah Chu Yeow, frederico]

RUBY ON RAILS 2.1

¿QUE HAY DE NUEVO?

CARLOS BRANDO

revisión: MARCOS TAPAJÓS - tapa: DANIEL LOPES - traducción a cargo de: GASTÓN RAMOS y LUCAS FLORIO