RUBY ON RAILS 2.1

WHAT'S NEW?

CARLOS BRANDO REVIEW: MARCOS TAPAJÓS - COVER: DANIEL LOPES

Ruby on Rails 2.1

WHAT'S NEW

Second Edition

Ruby on Rails 2.1

WHAT'S NEW

Second Edition

Carlos Brando
Marcos Tapajós

Second edition: June 2008

Carlos Brando
Website: www.nomedojogo.com

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

Chapter 1

Introduction

Around July of 2004 David Heinemeier Hansson publicly released the Ruby on Rails framework, which had been extracted from a web application called Basecamp. More than three years later, on the December 7th, 2007 Ruby on Rails version 2.0 was released with numerous important changes.

Six months have passed since then, and during this time more than **1400 developers** from all around the world have contributed **1600 patches** to the framework. Today, June 1st 2008, version 2.1 of the Ruby on Rails framework was released.

Major new features according to David:

- Timezones
- Dirty tracking
- Gem Dependencies
- Named scope

- UTC-based migrations
- Better caching

As always, to update or install the new version:

```
gem install rails
```

ACKNOWLEDGMENT

To Marcos Tapajós, co-author of this book. If it wasn't for him, you probably you wouldn't be reading this right now.

To Daniel Lopes who made the beautiful cover for this edition.

To all of the Ruby on Rails Brazilian community that helped directly or indirectly with this book, commenting on blog posts and giving suggestions. It's like I always say, the best of Rails is its community! Keep creating, inventing, and specially sharing.

TRANSLATORS

This book was proudly translated to english by these Brazilian guys:

Pedro Pimentel - http://www.pedropimentel.com

Chapters 3-8 and 10-13

Rafael Barbosa - http://www.act-as-newbie.com

Introduction and chapter 1

Ricardo S Yasuda - http://blog.shadowmaru.org

Chapter 14

Caike Souza - http://tech-death.blogspot.com

Chapter 2

Abraão Coelho - http://abrcoelho.net

Chapter 9

Bruno Miranda - http://brunomiranda.com

Reviser

Chapter 2

ActiveRecord

ActiveRecord is an object-relational mapping layer responsible for the interoperability among application and database and also responsible for data abstraction. (wikipedia)

THE SUM METHOD

Expressions in the sum method

Now we can use expressions in **ActiveRecord** methods that deal with calculation, like **sum**, for example:

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

Change in the default return value of the sum method

On previous versions, when we used **ActiveRecord**'s **sum** method to calculate the addition of all rows in a table and no row matched the conditions expressed during the method invocation, then the default return value would be **nil**.

In Rails 2.1 the default return value (that is when no row is found) is 0. See the example:

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

HAS_ONE

Support for the option through

The **has_one** method now has the option **through**. It works just like **has_many :through**, but it represents the association to a single **ActiveRecord** object.

```
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 with :source_type

The **has_one :through** method, just mentioned above, can also take **:source_type**. I will try to explain this through some examples. Let's start with these two classes:

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

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

What we are looking at here is a **Client** class which **has_many** kinds of contacts, since the **ContactCard** class has a polymorphic relationship.

Next step in our example, let's create two classes to represent 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 and **Business** relate to my **Client** class through the **ContactCard** table. In other words, I have two kinds of contacts, personal and business.

This is not going to work, however. Watch what happens when I try to retrieve a contact:

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

To make this work we have to use **:source_type**. Let's change our **Client** class:

```
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
```

Notice how we now have two different ways of retrieving our contacts and we can say what contact **:source_type** we are expecting.

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

NAMED_SCOPE

The *has_finder* gem has been added to Rails with a different name: **named_scope**.

To fully understand what this adition brought to Rails let's look at the following examples:

```
class Article < ActiveRecord::Base
  named_scope :published, :conditions => {:published => true}
  named_scope :containing_the_letter_a, :conditions => "body LIKE '%a%’"
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 => {…})
```

Instead of creating a new method named **published** to return all published posts, I'm using a **named_scope** to do it for me. But it can go even further than this. Let's look at another example of how it can be used:

```
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]
```

TESTING NAMED_SCOPE WITH PROXY_OPTIONS

Named scopes is a very interesting new feature for Rails 2.1, but after using it awhile you might have a hard time creating tests for more complex situations.

Let's look at an example:

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

How to create a test that validates the generation of the scope ?

To solve this issue, the method **proxy_options** was created. It allows us to examine the options used in **named_scope**. To test the code above we could write:

```
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
```

INCREMENT AND DECREMENT

ActiveRecord's methods **increment**, **increment!**, **decrement** and **decrement!** can now take a new optional parameter. On previous verions of Rails you could use these methods to add or subtract 1 (one) from a given column. In Rails 2.1 you can tell which value that is to be added or subtracted. Like this:

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

In the above example I am adding 5 points to player1 and subtracting 2 points from player2. Since this is an optional parameter, legacy code is not affected.

FIND

Conditions

From now on, you can pass an object as a parameter to **ActiveRecord**'s **find** method. See this example:

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

In this case, you can pass an instance of **Money** as a parameter to the **find** method from the **Account** class, like this:

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

Last

Up to now we could only use three operators to look for data using **ActiveRecord**'s **find** method. These are: **:first**, **:all** and the object's own id (in this case whe don't pass any argument to **find** besides the id itself)

In Rails 2.1 there is a fourth operator named **:last**. A few examples:

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

To fully understand how this new operator works, just look at the following test:

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

All

The static method **all** is an alias to the also static **find(:all)**. Example:

```
Topic.all is the same as Topic.find(:all)
```

First

The static method **first** is an alias to the also static **find(:first)**. Example:

```
Topic.first is the same as Topic.find(:first)
```

Last

The static method **last** is an alias to the also static **find(:last)**. Example:

```
Topic.last is the same as Topic.find(:last)
```

USING FIRST AND LAST METHODS IN NAMED_SCOPE

All the methods mentioned above also work in **named_scope**. Suppose we create a **named_scope** named **recent**. The following is legal:

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

EAGER LOADING

To explain this new funcionality, let's look at the following code:

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

I'm searching through table **authors** and also including tables **posts** and **comments** in my query through the **author_id** column, which is the default column name according to Rails' convention for foreign_key names. This search used to generate SQL queries like this:

```
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
```

Exactly one long SQL query with **joins** between tables **authors**, **posts** and **comments**. We call this **cartesian product**.

This type of query is not always good performance-wise, so it was changed for Rails 2.1. The same query for **Author** class now uses a different approach to retrieve information from all three tables. Instead of using one SQL query with all three tables, Rails now uses three different queries - one for each table - which are shorter queries than the former that used to be generated. The result can be seen in the log after executing the previous ruby on rails code:

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

In **most cases** three simpler queries will run faster than a complex and long query.

BELONGS_TO

The **belongs_to** method was changed in order to allow the use of **:dependent => :destroy** and **:delete** in associations. For example:

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

POLYMORPHIC URL

Helper methods for polymorphic URL are used as a more elegant solution to renamed routes when you're working with **ActiveRecord**.

These methods come in handy when you want to generate the URL for a **RESTful** resource without specifying the type it is going to be associated with.

It is very simple to work with them. Take a look at a few examples (commented out is how the same thing is done in versions of Rails prior to 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()
```

Notice how the **polymorphic_url** method is able to identify the type that is given to him and generates the correct routes. **Nested resources** and **namespaces** are also supported:

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

You can also use prefixes such as **new**, **edit** and **formatted**. Take a look at a few examples:

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

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

READONLY RELATIONSHIPS

A new feature is added to the relationship among models. To avoid change in a models' state you can now use **:readonly** when describing associations. Let's take a look at a few examples:

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

has_one :boss, :readonly => :true

belongs_to :project, :readonly => true

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

This way your associated models are safe from being edited from within this model. If you try editing any of them you will get an **ActiveRecord::ReadOnlyRecord** exception.

METHODS ADD_TIMESTAMPS AND REMOVE_TIMESTAMPS

We now have two new methods: **add_timestamps** and **remove_timestamps**. They add and remove, respectively, **timestamp** columns. Let's take a look at an example:

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

def self.down
  remove_timestamps :urls
  remove_timestamps :feeds
end
```

CALCULATIONS

ActiveRecord::Calculations has changed a bit to support table names. This comes in handy when we have relationships among different tables with the same column name. You have these two options now:

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

ACTIVERECORD::BASE.CREATE ACCEPTS BLOCKS

We are already used to **ActiveRecord::Base.new** accepting blocks. Now we can do the same thing in the **create** method:

```
# Creating an object and passing it a block describing its attributes
User.create(:first_name => 'Jamie') do |u|
  u.is_admin = false
end
```

We can also use the same method to create many objects at once:

```
# Creating an array of new objects using a block.
# The block is executed once for each of object that is created.
User.create([{:first_name => 'Jamie'}, {:first_name => 'Jeremy'}]) do |u|
  u.is_admin = false
end
```

And it also works with associations:

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

# ou

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

CHANGE_TABLE

The creation of **migrations** in Rails 2.0 was a lot sexier than on previous verions, but to alter a table using **migrations** was not sexy at all.

In Rails 2.1, alter table became also sexy with the new method **change_table**. Let's take a look at an example:

```
change_table :videos do |t|
  t.timestamps # this adds columns created_at and updated_at
  t.belongs_to :goat # this adds column goat_id (integer)
  t.string :name, :email, :limit => 20 # this adds columns name and email
  t.remove :name, :email # this removes columns name and email
end
```

The new method **change_table** works just like his cousin **create_table** but instead of creating a new table it just alters an already existing table by adding or removing columns and indexes.

```
change_table :table do |t|
  t.column # adds an ordinary column. Ex: t.column(:name, :string)
  t.index # adds a new index.
  t.timestamps
  t.change # changes the column definition. Ex: t.change(:name, :string, :limit => 80)
  t.change_default # changes the column default value.
  t.rename # changes the name of the column.
  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

Now in Rails we are able to keep track of changes made to **ActiveRecord**. It is possible to know if an object has been changed or not. In case it has been changed, we can track down its latest changes. Let's take look at a few examples:

article = Article.find(:first)

```
  article.changed?  #=> false

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

  # shows title before change
  article.title_was  #=> "Title"

  # before and after the change
  article.title_change  #=> ["Title", "New Title"]
```

As you can see it is very simple. You can also list all changes made to the object in one of two ways:

```
  # returns a list with all of the attributes that were changed
  article.changed  #=> ['title']

  # returns a hash with attributes that were changed
  # along with its values before and after
  article.changes  #=> { 'title’ => ["Title", "New Title"] }
```

Notice that when an object is saved, its status changes:

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

In case you want to change an object's state without using **attr=**, you will need to explicitly inform that the attribute was changed by using the method **attr_name_will_change!** (replace **attr** with an object's real attribute). Let's look at one last example:

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

PARTIAL UPDATES

The implementation of **Dirty Objects** was the starting point for another very interesting feature.

Since we can now track down what has changed in an object's state, why not use it to avoid unnecessary updates to the database ?

On previous versions of Rails when we called **save** from an already existing **ActiveRecord** object, all of its fields would be updated in the database. Even the ones that had not suffered any change.

This action could be greatly enhanced with the use of **Dirty Objects** and it is exactly what happened. Take a look at the SQL query generated in Rails 2.1 when trying to save an object that suffered a slight change:

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

# Let's change the title
article.title = "New Title"

# it creates the following SQL
article.save
#=>  "UPDATE articles SET title = 'New Title' WHERE id = 1"
```

Notice how only the fields that were changed in the application were updated in the database. If no field had been updated in the application, then **ActiveRecord** would not execute any update.

To enable/disable this new feature you change the **partial_updates** property related to your model.

```
# To enable it
MyClass.partial_updates = true
```

If you wish to enable/disable this feature to all of your models, then you must edit the file *config/initializers/new_rails_defaults.rb*:

```
# Enable it to all models
ActiveRecord::Base.partial_updates = true
```

Don't forget to also inform Rails through *config/initializers/new_rails_defaults.rb* if you plan to edit a field without using the method **attr=**, like this:

```
# If you use **attr=**,
# then it's ok not informing
person.name = 'bobby'
person.name_change    # => ['bob', 'bobby']
```

```
# But you must inform that the field will be changed
# if you plan not to use **attr=**
person.name_will_change!
person.name << 'by'
person.name_change    # => ['bob', 'bobby']
```

If you don't inform changes like these will be occurring, then they won't be able to be tracked down and your database table won't be correctly updated.

SMALLINT, INT OR BIGINT IN MYSQL?

The **MySQL** adapter for **ActiveRecord** is now smarter when creating or altering columns in the database using integer types. According to the option **:limit**, it will now tell if the column will be a **smallint**, **int** or **bigint**. Let's take a look at an example that does just that:

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

Now let's map it in a **migration** file and see what column type will be created for each column:

```
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)

  # if :limit is not informed: int(11)
  t.integer :column_four # int(11)
end
```

The **PostgreSQL** adapter had this feature already and **MySQL** just caught up.

OPTION :SELECT IN HAS_ONE AND BELONGS_TO

The already known methods **has_one** and **belongs_to** just got a now option: **:select**.

Its default value is "" *(as in "SELECT* FROM table"), but you can edit it to retrieve only the columns you are going to be using.

Don't forget to include the **primary** and **foreign keys**, otherwise you will get an error.

The **belongs_to** method does not have the option **:order** anymore. But don't worry, because it didn't really have a use.

STORING THE COMPLETE NAME OF A CLASS WHEN USING STI

Whenever we use **models** with **namespace** and **STI**, **ActiveRecord** stores just the name of the class, without its **namespace** (*demodulized*). This will only work when all of the classes in the **STI** are in the same **namespace**. Let's look at an example:

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

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

item2 = CollectionItem.find(item.id)
# returns an error, because it can't find
# the class Item
```

This change adds a new option that makes **ActiveRecord** store the whole name of the class

To enable/disable this feature, you should include or edit the following in your **environment.rb**.

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

Its default value is true.

METHOD TABLE_EXISTS?

New method for the **AbstractAdapter** class: **table_exists**. It is very simple to use:

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

TIMESTAMPED MIGRATIONS

When you are just starting Rails or developing something on your own, **migrations** seem to be the best solution to all of your problems. However, when with a team of developers on a project, you will find out (if you haven't already) that it becomes a bit more troublesome to handle race conditions on migrations. The new timestamped migrations in Rails 2.1 to the rescue.

Before the introduction of **timestamped migrations**, each new migration created had a number which prefaced the migration name. If two migrations were generated by different developers and not committed instantly, they could end up having the same number preface by different migration info. At this point your schema_info is out of date and you have a conflict in your source control.

There were many ways to "try" to solve this problem. Many plugins were created with different approaches to solve this issue. Despite the plugins available, one thing was clear: the old way simply didn't work.

If you were using Git, then you would be digging an even deeper hole, since your team would probably have a couple of working branches and out-of-date **migrations** in all of them. You would have serious conflict problems when merging branches.

To solve this huge problem, the core team changed how **migrations** works in Rails. Instead of prefacing each migration file with a number from corresponding to the current schema_info's version count, it is now prefaced with a string based on the **UTC** time and following the format YYYYMMDDHHMMSS.

Also a new table called **schema_migrations** was created and it stores which **migrations** that have already been executed. That way, if anyone creates a **migration** with a smaller number, rails will **rollback** migrations until the previous version and then run everything up to the current version.

Apparently, it solves the conflict problem with **migrations**.

There is an option to disable this feature by including the following line in **environment.rb**:

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

There are also new rake tasks to "walk through" **migrations**:

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

Chapter 3

ActiveSupport

Active Support is a collection of useful classes and default libraries extensions which were considered useful for Ruby on Rails Applications. (wikipedia)

ACTIVESUPPORT::COREEXTENSIONS::DATE::CALCULATIONS

Time#end_of_day

Returns the current date with the time set to 11:59:59 PM.

Time#end_of_week

Returns the end of the week (Sunday 11:59:59 PM).

Time#end_of_quarter

Returns a Date object representing the end of the trimester. In other words, it returns the last day of either march, june, september or december, depending on the current date.

Time#end_of_year

Returns December 31 at 11:59:59 PM

Time#in_time_zone

This method is similar to **Time#localtime**, except by the fact that it uses **Time.zone** instead of the underlying operating system's timezone. You can pass either **TimeZone** or **String** as a parameter. Look at some examples:

```
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

A bug in the method **days_in_month** was fixed, which returned the wrong number of days in february when the year was was not specified.

Changes comprise in using the current year as the default value when not specifying the year in method call. Suppose you were in a leap year. Look the following example:

```
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

DateTime class received a new method called **to_f** which returns the date as a type float number representing the number of seconds since the Unix epoch (number of seconds since january 1st, 1970, midnight).

Date.current

Date class received a new method called **current** which must now be used instead of **Date.today**, because it considers the timezone set in **config.time_zone** in case it is set, returning a **Time.zone.today**. If it is not set, then it returns a **Date.today**.

FRAGMENT_EXIST?

Two new methods were added to **cache_store**: **fragment_exist?** and **exist?**.

The method **fragment_exist?** does exactly what you would expect, it verifies if a cached fragment informed by one key exists. Basically replacing the famous:

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

exist? method was added to **cache_store**, while **fragment_exist?** is a helper which you could use inside your controller.

UTC OR GMT?

An amendment, but interesting. Until now Rails has been using the UTC acronym a lot, but when the method **to_s** from **TimeZone** is called, it will print GMT, not UTC. This is due to the fact that the GMT acronym is the most common among end users.

If you take a look in Windows control panel, where you can choose timezone, you'll notice the acronym used is GMT. Google and Yahoo also have been using GMT within their products.

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

JSON ESCAPE

json_escape method works like **html_escape**. It's very useful when we need to show **JSON** strings in a **HTML** page, for example, in a documentation process.

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

We can also use the shortcut **j** when in ERB:

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

If you want all **JSON** code to be 'escaped' by default, include the following line in your *environment.rb* file:

```
ActiveSupport.escape_html_entities_in_json = true
```

MEM_CACHE_STORE NOW ACCEPTS OPTIONS

The inclusion of **Memcache-Client** inside **ActiveSupport::Cache** made things easier than ever, but it also took away its flexibility in not allowing us to customize nothing more than the IP of the **memcached** server.

Jonathan Weiss made a patch which was included in Rails allowing extra options like these:

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

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

or

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

TIME.CURRENT

A new method for **Time** class. The **current** method's return depends on **config.time_zone**, if it was specified before, the method will return a **Time.zone.now**, otherwise will be a **Time.now**.

```
# return value depends on config.time_zone
Time.current
```

since and **ago** methods also had their returning values changed, returning a **TimeWithZone** in case **config.time_zone** as specified.

It makes the **Time.current** method as new default method to get the actual time, replacing the **Time.now** (which keeps existing, but it doesn't consider the specified timezone).

The **datetime_select** methods, **select_datetime** and **select_time** also have been updated to have their default returning as **Time.current**.

REMOVING WHITESPACES WITH SQUISH METHOD

Two new methods added to the **String** object, **squish** and **squish!**.

These methods do the same as **strip** method. It removes white spaces from the beginning and the end of the text. It also removes unused white-spaces (multiple white-spaces) from the middle of the text Look the example:

```
“    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 is a layer responsible by the client side implementation of RESTful systems. Through ActiveResource is possible to consume RESTful services by using objects that work like a proxy for remote services.

USING EMAIL AS USERNAME.

Some services use the e-mail as username, which forces us to use an URL like the following:

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

But this was causing a problem. Because we have two (@), the interpreter got lost when reading this. For this reason, **ActiveResource** was extended a little bit more, envisioning to make easier to use e-emails for authentication. Now you can do the following:

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

THE CLONE METHOD

Now we can clone an existing resource:

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

Please note the copied object doesn't clone any of the class attributes, just the resource attributes.

```
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 uses **HTTP** to access RESTful API's and because of that it is susceptible to problems with slow server responses or non-reachable servers. In some cases, calls to ActiveResource can expire (timeout). Now you can control the expiration time with the timeout property.

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

In this example above a timeout of 5 seconds was configured. A low value is recommended to allow your system to fail-fast, preventing cascading fails which could incapacitate your server.

Internally, ActiveResource shares resource from the Net:HTTP library to make HTTP requests. When you define a value for the timeout property, the same value is defined for the **read_timeout** of the Net:HTTP object instance which is being used.

The default value is 60 seconds.

Chapter 5

ActionPack

Comprises ActionView (visualization generation for end user, like HTML, XML, JavaScript, and etc) and ActionController (business flow control) (adapted from wikipedia)

TIMEZONE

Defining a default timezone

One new option was added to **time_zone_select** method, you can now present a default value in cases which your user didn't select any **TimeZone**, or when the database column is null. To achieve this, a **:default** option was created and can be used as such:

```
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)")
```

In cases where we use the **:default** option, it must be shown with the informed **TimeZone** already selected.

The formatted_offset method

The **formatted_offset** method was included in the **Time** and **DateTime** classes to return with the format **+HH:MM** the deviation of UTC time. For example, in our timezone (Brasilia time) the deviation value returned by the method would be a string with its value set to **"-03:00"**.

Let's see some examples:

Getting the deviation from a DateTime:

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

Now from Time:

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

Note this method returns **string**, which can be either formatted or not depending of the value given as parameter.

The with_env_tz method

The **with_env_tz** method allows us to make tests with different timezones in a very simple way:

```
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
```

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!

Was removed for not being used anymore

Time#in_current_time_zone

Was modified to return **self** when **Time.zone** is null.

Time#change_time_zone_to_current

Was modified to return **self** when **Time.zone** is null.

TimeZone#now

The **TimeZone#now** method was modified to return an **ActiveSupport::TimeWithZone** representing the current time in the configured timezone as defined in **Time.zone**. For example:

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

Compare_with_coercion

The method **compare_with_coercion** (with an alias for <=>) was created in **Time** e **DateTime** classes, becoming possible to make a chronological comparison between the **Time**, **DateTime** classes and instances of **ActiveSupport::TimeWithZone** objects. For a better understanding, take a look the examples bellow (each line result is in the comment placed following the code):

```
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?

The **between?** method was included in the **TimeWithZone** class to verify if an instance is found between two dates. Example:

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

TimeZone#parse

This method creates a new instance of **ActiveSupport::TimeWithZone** from a string. For example:

```
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

This method can be used to create a new instance of **ActiveSupport::TimeWithZone** from the number of seconds since Unix epoch. For example:

```
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
```

More methods

The **to_a**, **to_f**, **to_i**, **httpdate**, **rfc2822**, **to_yaml**, **to_datetime** and **eql?** methodos were added in the **TImeWithZone** class. For more information about these methodos please head to **Rails** documentation.

TimeWithZone class preparing itself for Ruby 1.9

In Ruby 1.9 we'll have some new methods in the **Time** class, methods such as:

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

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

A respective method exists for each day of the week.

Another interesting things is that **to_s** method of **Time** object will have a different returning value. Today when we execute **Time.new.to_s**, we have the following:

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

In Ruby 1.9 we will have:

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

What's all of that have to do with Rails 2.1? Everything. Rails is being prepared to deal with these modifications. The **TimeWithZone** class, for example, just received some enhancements to work with the methods of the first example.

AUTO LINK

The **auto_link** method receives any text given as parameter, and if the texts has some e-mail address or an website, it returns the same text, but with hyperlinks.

For example:

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

Some sites, like Amazon, use the "=" symbol in their URL's. This method doesn't recognize that symbol. Look how the method behaves in such case:

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

Note the method finished the hyperlink exactly before the "=" symbol, before Rails 2.1 that symbol was not supported.

The same method was updated later to also allow the use of URL's with parenthesis.

An URL example using parenthesis:

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

LABELS

When creating a new form using **scaffold** it will be created with the following code:

```
<% 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 %>
```

The **label** method was included. This method returns a *string* with the column title inside an HTML **<label>** tag.

```
>> 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>
```

Did you notice the **for** parameter inside the tag ? "post_title" is the textbox title which contains our post title. The **<label>** tag is in fact a label associated to the **post_title** object. When someone clicks on the label (which isn't a link) the associated HTML controller receives focus.

Robby Russell wrote an interesting post in his blog about this subject. You can read it on: http://www.robbyonrails.com/articles/2007/12/02/that-checkbox-needs-a-label

Also included was the **label_tag** method in **FormTagHelper**. This method works like label, but in a simpler way:

```
>> 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>
```

The method also accepts the **:for** option, Look an example:

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

This will return something like this:

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

A NEW WAY OF USING PARTIALS

Something very common in Rails software development is the use of **partials** to avoid code repetition. Here is a usage example:

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

Partial is a code fragment (a template). The advantage of using a **partial** is to avoid unnecessary code repetition. Using a **partial** is very simple, you can start with something like this: **render :partial => "name"**. You must create a file with the same name of your **partial**, but using an underscore in front of it.

The code above shows ho we are used to do it, in Rails 2.1 you'll do the same thing in a slightly different way:

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

In this example we render the partial "users/_form", which will receive a variable called "form" with the references created by the **FormBuilder**.

The old way will continue to work in Rails 2.1.

NEW NAMESPACES IN ATOM FEED

Do you know the **atom_feed** method? It is one of the new features of Rails 2.0, making easier the creation of Atom feeds. See an example of use:

In a *index.atom.builder* file:

```
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
```

What is an Atom feed ? Atom is the name of XML based style and meta data. In other words is a protocol to publish content in Internet that is often updated, like a blog, for example. Feeds are always published in XML and in Atom it is identified as an application/atom+xml media type.

In the first versions of Rails 2.0 this method used to accept as parameter **:language**, **:root_url** and **:url** options, you can obtain more information about these methods in Rails Documentation. But with the update made, we can now include new namespaces in the root element of the feed. For example:

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

Will return:

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

Modifying the example used before, we could use this way:

```
  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

All **fragment_cache_key** methods now return by default the namespace 'view/' as prefix.

All caching stores were removed from **ActionController::Caching::Fragments::** *and now they can be found in* ***ActiveSupport::Cache::***. In this case, if you made a reference to a store, like **ActionController::Caching::Fragments::MemoryStore**, for example, you'll have to change its reference to **ActiveSupport::Cache::MemoryStore**.

ActionController::Base.fragment_cache_store is no more and **ActionController::Base.cache_store** takes its place.

It was included in the **ActiveRecord::Base** the **cache_key** method to facilitate the storing cache of Active Records by the new libraries **ActiveSupport::Cache::***. It works this way:

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

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

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

ActiveSupport::Gzip.decompress/compress was included to make easier the use as a wrapper for **Zlib**.

Now you can use among environment options the **config.cache_store** to specify the default place of caching store. It is worth mentioning, if the **tmp/cache** directory exists, the default is **FileStore**, in other case the **MemoryStore** will be used. You can configure in the following ways:

```
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")
```

To make things even easier, the comment bellow is included in *environments/production.rb* file, in order to remind you of this option.

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

APPLYING TITLE FORMATTING IN STRINGS

There was a bug when using **String#titleize** method in a string containing 's . The bug made the method return the 's in uppercase. See an example:

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

See the same example, but with the bug fixed:

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

ACTION_NAME

In order to find out which view was called during running time of your view, we can use the **action_name** method:

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

The return value will be the same as using **params[:action]**, but in a more elegant way.

CACHES_ACTION ACCEPTS CONDITIONALS

The **caches_action** method now accepts the **:if** option, allowing the use of conditionals to specify when an **action** can be **cached**. For example:

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

In the example above, the **action index** will go to the **cache** only if it's not accessed by a JSON request.

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 is the layer responsible by receiving web requests and taking decisions of what is going to be run and rendered or to redirect the request to another action. An Action is defined as public methods within controllers which are automatically available through routes.

ACTIONCONTROLLER::ROUTING

Map.root

Now, when using **map.root** you can be much more **DRY** using an alias for that.

In the earlier versions of rails you used to do it like this:

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

Now you can do it this way:

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

Routes recognition

Routes recognition's old implementation used to sweep all routes, one by one, and often turned to be very time consuming. A new and smarter implementation was developed. It creates a tree for routes and the route recognition is made by prefixing, skipping similar routes. This approach lowers recognition time in approximately 2.7 times.

All the new implementations are in the file **recognition_optimisation.rb** and its working details are well explained in the comments. See the the documentation inside the source code itself for more information about its implementation.

Assert_routing

Now it's possible to test a route with an HTTP method. Look at the following example:

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

Map.resources

Imagine you have a site written in a language other than english, and you want to taylor your routes to use the same language. In other words, instead of having:

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

You wished to have something like this:

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

This was already possible, but not in a simple way and without compromising some rails conventions.

Now we have the option **:as** within **map.resources** to personalize our routes. Look our example to get the URL above in portuguese:

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

ACTIONCONTROLLER::CACHING::SWEEPING

In the earlier versions of rails, when we declared a **sweeper**, we had to inform the class using symbols:

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

Now it's possible to explicitly declare a class instead of using a symbol. This is necessary if your **sweeper** is inside a module. Though you can still use symbols for other cases from now on you can also do it this way:

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

Chapter 7

ActionView

ActionView is the layer responsible by the generation of the viewable interface visible to users through conversion of ERB templates.

ACTIONVIEW::HELPERS::FORMHELPER

fields_for form_for with index option.

The **#fields_for** and **form_for** methods received the **:index** option, removing the need of using **:index => nil** on each form object.

This how you used to code it:

```
<% 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 %>
```

The following is the new method:

```
<% 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

Now, all these module methods dealing with dates (**date_select**, **time_select**, **select_datetime**, etc.) are accepting **HTML** options. Look an example using **date_select**

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

date_helper

The **date_helper** method was updated to use **Date.current** in order to define its default value.

ACTIONVIEW::HELPERS::ASSETTAGHELPER

register_javascript_expansion

This method registers one or more javascript files to be included when a symbol, defined by the programmer, is given as a parameter to the **javascript_include_tag** method. The idea is to call this method inside the **init.rb** of your plugin, in order to register the javascript files which your plugin places in the folder **public/javascripts**. Let's see how it works:

```
# In the init.rb file
ActionView::Helpers::AssetTagHelper.register_javascript_expansion
    :monkey => ["head", "body", "tail"]

# In our view:
javascript_include_tag :monkey

# We are going to have:
<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

This method does exactly the same as the **ActionView::Helpers::AssetTagHelper#register_javascript_expansion** method, but it creates a symbol to be used later when making calls to **stylesheet_link_tag** method. Look an example:

```
# In the init.rb file
ActionView::Helpers::AssetTagHelper.register_stylesheet_expansion
    :monkey => ["head", "body", "tail"]
```

```
# In our view:
stylesheet_link_tag :monkey

# We are going to have:
<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

A **:confirm** option was added to the parameters of **#submit_tag** method. This option works the same way like the method **link_to**. Look an example:

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

ACTIONVIEW::HELPERS::NUMBERHELPER

number_to_currency

The **number_to_currency** method now accepts the **:format** option as a parameter, letting us to format the method's returning value. In earlier versions, when we had to format values for our local currency, we needed to include a space in front of **:unit** option to make the output format correct. See the examples:

```
# R$ is the symbol for Brazilian currency
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"
```

Besides that, we can customize in other forms, for example:

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

When creating your own formatting string, you can use the following parameters:

```
%u For the currency
%n For the number
```

ACTIONVIEW::HELPERS::TEXTHELPER

excerpt

The **excerpt** method is a helper to find a word inside a phrase and return an abbreviation of that phrase with the number of given characters as parameters before and after the word, adding, when necessary the "…". See the following example:

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

But the problem is it was buggy. If you count, you'll see the method returned 6 chars and not 5. This bug was fixed. Look at the example of the correct output for this method:

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

simple_format

The **simple_format** method basically receives as a parameter any text and formats it in a simple way to **HTML**. It takes the text and replaces line breaks (\n) by **HTML** tag "< br />". And when we have two line breaks one after other (\n\n) it separates the text in paragraphs using "< p>"tag.

In Rails 2.1 this method receives an additional parameter. Besides text, we are going to be able to inform which **HTML** attributes we would like "< p>" tag had. Look the examples:

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

The **HTML** attributes will be added in all "< p>" tags created by the method.

Chapter 8

Railties

CONFIG.GEM

Now it's possible to configure all necessary gems to get a project running by using a new feature called **config.gem**. In **environment.rb** file you can specify which gems your project depends to run. Look at the example:

```
config.gem "bj"

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

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

To install all dependencies at once, we just use a Rake task:

```
# Installs all specified gems
rake gems:install
```

It's also possible to list which gems are being used in the running project by using:

```
# Listing all gem dependencies
rake gems
```

If one of the gems have a **rails/init.rb** file and you want to take the gem with your application, you can use:

```
# Copy the specified gem to vendor/gems/nome_do_gem-x.x.x
rake gems:unpack GEM=gem_name
```

Then, the gem will be copied to the directory **vendor/gems/gem_name-x.x.x**. In case you don't specify gem name, Rails will copy all gems to the directory **vendor/gem**

CONFIG.GEM IN PLUGINS

The **config.gem** feature is also available for use with plugins.

Until Rails 2.0 the **init.rb** file of a plugin used to look like this:

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

But in Rails 2.1 the **init.rb** file would be:

```
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
```

So when you run the task to install all necessary gems, these gems will be among them.

GEMS:BUILD

The **gems:build** task compiles all native extensions of gems which were installed through **gems:unpack**. The syntax is the following:

```
rake gems:build # For all gems
rake gems:build GEM=mygem # I'm specifing the gem
```

NEW MESSAGE WHEN STARTING SERVER

There was a little improvement when starting Rails server, now it shows which Rails version is being loaded:

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

RAILS.PUBLIC_PATH

The **Rails.public_path** shortcut was added to recover the path of the project's **"public"** directory.

```
Rails.public_path
```

RAILS.LOGGER, RAILS.ROOT, RAILS.ENV AND RAILS.CACHE

In Rails 2.1 instead of using the constants: **RAILS_DEFAULT_LOGGER**, **RAILS_ROOT**, **RAILS_ENV** and **RAILS_CACHE** you can use:

```
# RAILS_DEFAULT_LOGGER
Rails.logger

# RAILS_ROOT
Rails.root

# RAILS_ENV
Rails.env

# RAILS_CACHE
Rails.cache
```

RAILS.VERSION

In earlier versions to discover at runtime time which Rails version is in use the following was used:

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

In Rails 2.1 this was changed to shorter:

```
Rails.version
```

GETTING INFORMATION ABOUT A PLUGIN

This is one of the new Rails 2.0 features which you probably never used. I say "probably", because in some very specific cases it can be useful, for example, to know a plugin version.

To test it, we need to create a new file called *about.yml* in the plugin directory, something like this:

```
author: Carlos Brando
version: 1.2.0
description: A description about the plugin
url: http://www.nomedojogo.com
```

We can get this information later this way:

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

If you find some good use for this feature and want to share with me, maybe I can change my mind about its real need.

Chapter 9

Rake Tasks, Plugins and Scripts

TASKS

rails:update

From now on every time you run the task **rake rails:freeze:edge** it will also run **rails:update**, updating the config files and *JavaScripts*.

Database in 127.0.0.1

The databases.rake used to look only in localhost for local databases, it will now also consider the IP **127.0.0.1**. This works for both **create** and **drop** tasks. The databases.rake file was also refactored to make the code less repetitive.

Freezing a specific Rails release.

Until Rails 2.1 it wasn't possible to freeze a specific Rails release inside your project, you could only use its Revision as a parameter. In Rails 2.1, we can freeze a specific release using the command below:

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

TIMEZONE

rake time:zones:all

Return all the time zones known to Rails, grouped by offset. You can also filter the return value using the optional parameter OFFSET, for instance: OFFSET=-6.

rake time:zones:us

Shows a list with all US time zones. The OFFSET option is still valid here.

rake time:zones:local

Return all the time zones known to Rails that are in the same offset of your OS.

SCRIPTS

plugin

The command *script/plugin install* now allows the use of –e/--export option, so that it issues a svn export. Added support for plugins hosted in GIT repositories.

dbconsole

This script does the same thing as script/console but for your database. In other words it connects to the command line client of your database.

Looking at the code, this apparently will only work for mysql, postgresql and sqlite(3). When another database is configured in database.yml, this script will show: "not supported for this database type".

PLUGINS

Gems can now be plugins

Now, any gem that has a **rails/init.rb** file can be installed inside the **vendor** directory of your Rails project just like a **plugin**.

Using generators in plugins

It's possible to configure **Rails** to search for **plugins** in places other than the **vendor/plugins** directory, just including this line of code in your **environment.rb**.

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

Rails 2.0 had a bug in this configuration that showed up when the plugin had generators. Because of that bug Rails only found generators in plugins that were inside the **vendor/plugins** directory. In 2.1 this bug was squashed.

Chapter 10

Prototype and script.aculo.us

PROTOTYPE

Rails 2.1 now uses Prototype version 1.6.0.1. It serves as a preparatory to version 1.8.1 of script.aculo.us.

Chapter 11

Ruby 1.9

DETAILS

The main focus of Rails changes was Ruby 1.9, even minor details were analyzed to increase Rails compatibility with the new Ruby version. Details like changing from **File.exists?** to **File.exist?** were not kept aside.

Also, in Ruby 1.9, the module **Base64** (base64.rb) was removed, because of that, all references to it were replaced by **ActiveSupport::Base64**.

NEW METHODOS FOR DATETIME CLASS

In order to keep compatibility (duck-typing) with **Time** class, three new methods were added to **DateTime** class. The methods are **#utc**, **#utc?** and **#utc_offset**. Look an example for each one:

```
>> 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

NATIVE RUBY-DEBUG

It was enabled again the option for using **ruby-debug** in Rails tests. Since you already have the gem installed, it only needs the **debugger** method.

Chapter 13

Bugs and Fixes

ADD COLUMNS IN POSTGRESQL

There was a bug when using **PostgreSQL**. It occurred when creating a migration for adding a column in an existing table. Look an example:

File: *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 we are creating a column with **:precision => 6** and **:scale => 2**. Now run **rake db:migrate** and let's see how is our table in database:

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

See "cost" column which we just created. It is a common **numeric**, but it was supposed to be a column like "price", above it, more precisely a **numeric(6,2)**. In Rails 2.1 this error doesn't occur anymore and the column will be created in the correct way.

MIME TYPES

A bug which didn't allow you to define the assigned attribute for **request.format** using a symbol was fixed. Now you can use the code bellow:

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

BUG FIXES IN CHANGE_COLUMN

An existing bug when using the **change_column** method with **:null => true** in a column created using **:null => false** was also fixed. Because of this bug no changes were made when using the method.

Chapter 14

Additional Information

PROTECTING FROM CROSS SITE SCRIPTING

In Rails 2.0 the file *application.rb* looks like this:

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

  protect_from_forgery
end
```

Notice the call to the method **protect_from_forgery**.

Have you heard about Cross Site Scripting? This is the name of a security failure easily found in most websites and web applications on the web today. It allows evil people (such as teenagers with nothing to do and no social life) to alter the content of web pages, perform fishing attacks, take over the control of the browser through JavaScript code. In most cases force the user to execute a command without knowledge. That last type of attack is called cross-site request forgery.

Cross Site Request Forgery is a type of attack that forces legit users to execute a series of commands without even knowing. And with the increasing use of Ajax, things are getting even worse.

Actually, this method is useful to make sure that all forms your application is receiving are coming from only your application, and not from a link in another site. It achieves this by including a token based on the session in all forms and Ajax requests generated by Rails. Later it checks the authenticity of this token in the controller.

Remember that GET requests are not protected. But it won't be a problem if we use it only to bring data, and never to alter or save anything in our database.

If you want to learn more about CSRF (Cross-Site Request Forgery) use the addresses below:

- 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

But remember that this is not a definitive solution to our problem, or like we usually say, it's not a silver bullet.

USED METHOD_MISSING, THEN DON'T LEAVE LOOSE ENDS

Due to Ruby's dynamic nature, the method **respond_to?** is crucial. How many times have you checked if a method exists in the object ? Or how often have you checked if the object is what we are expecting (**is_a?**)?

However there's something really important that many people forget. Look at this class's usage of the method **method_missing**:

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

rex = Dog.new
rex.bark #=> woofwof!
rex.bark! #=> woofwoof!
rex.bark_and_run #=> woofwoof!
```

I think you already know **method_missing**, don't you? In the example above I'm creating an instance of the class **Dog** and calling the methods **bark**, **bark!** e **bark_and_run** that don't exist. Then the method **method_missing** is called, where I use a simple regular expression to return "woofwoof!", whenever the name of the method begins with bark.

But look what happens when I try to use the method **respond_to?**:

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

It returns false, and that makes sense since the method doesn't really exist. Then it's my responsibility to change the method **respond_to?** to work properly using my special rule. I'll change my class to this:

```
class Dog
  METHOD_BARK = /^bark/
```

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

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

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

Now we're talking! This is a common mistake that I've seen often including in the Rails codebase itself. Try to execute a **respond_to?** to check the existence of methods like **find_by_name**, for example.

Ruby is an amazing and highly flexible language, but if you don't watch it you can leave loose ends like these.

Of course that in Rails 2.1 this problem was fixed, you can use **respond_to?** to check the existence of methods like **find_by_something**.

POSTGRESQL

In Rails 2.0, the adapter for **PostgreSQL** had support only for versions 8.1 up to 8.3. Support for versions 7.4 up to 8.3 has been added to Rails 2.1.

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]

```ruby
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

WHAT'S NEW?

```
def temp_pa
  p = temp_     s.  st
  p.respond_       (:  th)     p.path :     o_s
end
def render_partial(partial_path, object_assigns = nil, local_assigns = n
  case partial_path
  when String, Symbol, NilClass
    path, partial_name = partial_pieces(partial_path)
    object = extracting_object(partial_name, object_assigns)
    local_assigns = local_assigns ? local_assigns.clone : {}
    add_counter_to_local_assigns!(partial_name, local_assigns)
    add_object_to_local_assigns!(partial_name, local_assigns, object)

    if logger && logger.debug?
      ActionController::Base.benchmark("Rendered #{path}/_#{partial_name
        render("#{path}/_#{partial_name}", local_assigns)
      end
    else
```

CARLOS BRANDO REVIEW: MARCOS TAPAJÓS - COVER: DANIEL LOPES