RUBY ON RAILS 2.1

O QUE HÁ DE NOVO?

CARLOS BRANDO

REVISÃO: MARCOS TAPAJÓS - CAPA/C. CAPA: DANIEL LOPES

Ruby on Rails 2.1

O QUE HÁ DE NOVO

Primeira Edição

Escrito por Carlos Brando
Revisado por Marcos Tapajós

Primeira edição: Junho de 2008

Carlos Brando
Site: www.nomedojogo.com

Marcos Tapajós
Site: www.improveit.com.br/empresa/tapajos

Capítulo 1

Introdução

Por volta do mês de julho de 2004 David Heinemeier Hansson lançou publicamente o framework Ruby on Rails, que havia sido extraído de um software chamado Basecamp. Mais de três anos depois, no dia 7 de dezembro de 2007 o Ruby on Rails chegou a sua versão 2.0 com diversas alterações importantes.

De lá para cá se passaram seis meses, e neste tempo mais de **1400 programadores** do mundo todo contribuiram criando **1600 patches**. E hoje, 1 de junho de 2008 o Ruby on Rails chega à sua versão 2.1.

De acordo com David as principais novidades nesta versão são:

- Timezones
- Dirty tracking
- Gem Dependencies
- Named scope
- UTC-based migrations

- Better caching

Para atualizar ou instalar a nova versão, é o de sempre:

```
gem install rails
```

AGRADECIMENTOS

Ao Marcos Tapajós que é o co-autor deste livro. Se não fosse por ele acho que você não estaria lendo isto.

Ao Daniel Lopes que fez uma linda capa para esta edição.

A toda a comunidade brasileira de Ruby on Rails que colaborou direta ou indiretamente com este livro, comentando os textos no blog e dando sugestões. É como sempre costumo dizer, o melhor do Rails é a comunidade! Continuem criando, inventando e principalmente compartilhando...

Capítulo 2

ActiveRecord

O Active Record é uma camada de mapeamento objeto-relacional (object-relational mapping layer), responsável pela interoperabilidade entre a aplicação e o banco de dados e pela abstração dos dados. (wikipedia)

O MÉTODO SUM

Expressões no método sum

Agora podemos usar expressões em métodos de cálculo do **ActiveRecord**, como no método **sum**, por exemplo.

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

Alteração no retorno padrão do método sum

Nas versões anteriores, quando usávamos o método **sum** do **ActiveRecord** para calcular a soma de uma determinada coluna para todos os registros de uma tabela, e nenhum registro correspondia às condições expostas na execução do método, o retorno padrão era **nil**.

No Rails 2.1 o retorno padrão (quando nenhum registro é encontrado) é 0. Veja um exemplo:

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

HAS_ONE

Suporte à opção through

O método **has_one** recebeu suporte à opção **through**. Ele funciona exatamente como o **has_many :through**, mas para apenas um relacionamento. Exemplo:

```
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 com :source_type

O método **has_one :through**, citado acima, também suporta a opção **:source_type**. Vou tentar explicar isto através de exemplos. Vamos começar com estas duas classes:

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

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

O que nos interessa aqui é que tenho uma classe **Client** que possui (**has_many**) muitos contatos, que pode ser de qualquer tipo, já que a classe **ContactCard** possui um relacionamento polimórfico.

Para seguir com nosso exemplo, vamos criar duas classes que representarão um **ContactCard**:

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

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

Person e **Business** estão relacionados com a minha classe **Client**, através da tabela **ContactCard**, ou seja eu tenho dois tipos de contatos, os pessoais e os de negócios. O problema é que isto não vai funcionar, veja o que acontece quando tento recuperar algum contato:

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

Para fazer isto funcionar teremos de usar a opção **:source_type**. Vamos alterar nossa classe **Client**:

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

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

Note que agora temos duas formas diferentes de recuperar nossos contatos, onde eu deixo claro qual tipo (**:source_type**) de contato estou esperando.

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

NAMED_SCOPE

O gem *has_finder* foi incorporado ao Rails, mas com um nome diferente: **named_scope**.

Para entender o que isto acrescentou de novo ao Rails veja os exemplos abaixo:

```
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 => {…})
```

Ao invés de criar um método **published** para retornar os posts já publicados, estou usando o **named_scope** para fazer isto. Mas o método é um pouco mais robusto do que isto. Veja mais alguns exemplos de como ele pode ser usado:

```
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]
```

TESTANDO NAMED_SCOPE COM PROXY_OPTIONS

Named scopes é uma novidade muito interessante no Rails 2.1, mas após usar por um tempo este recurso, você pode descobrir que criar testes para estruturas mais complexas pode ser muito díficil.

Vamos pegar um exemplo:

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

Como criar um teste que valide a geração correta do escopo?

Para facilitar isto foi criado o método **proxy_options**, que permite examinar as opções que estão sendo usadas no **named_scope**. Para testar o exemplo acima, poderíamos fazer assim:

```
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 E DECREMENT

Os métodos **increment**, **increment!**, **decrement** e **decrement!** do **ActiveRecord** receberam mais um parâmetro como opcional. Nas versões anteriores do Rails você podia usar estes métodos para aumentar ou diminuir o valor de uma coluna em 1 (um). Mas a partir desta versão você poderá especificar o valor a ser adicionado ou subtraído se desejar. Assim:

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

No exemplo acima estou somando 5 à pontuação atual do jogador 1 e subtraindo 2 da pontuação atual do jogador 2. Como este parâmetro é opcional, os seus códigos antigos não serão afetados.

FIND

Conditions

A partir de agora é possível passar um objeto como parâmetro no método **find** de uma classe **ActiveRecord**. Veja este caso como exemplo:

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

Nesse caso, você pode passar um objeto **Money** como parâmetro no método **find** da classe **Account**, assim:

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

Last

Até agora podíamos usar apenas três operadores para procurar dados usando o método **find** do **ActiveRecord**: **:first**, **:all** e o próprio id do objeto (neste caso não usamos um operador especifico, mas a falta de um significa que estamos passando o id).

Agora teremos um quarto operador o **:last**. Veja alguns exemplos:

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

Para entender como esse método foi implementado basta olhar um dos seus testes:

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

All

O método estático **all** é um alias para o, também estático, **find(:all)**. Exemplo:

```
Topic.all é equivalente ao Topic.find(:all)
```

First

O método estático **first** é um alias para o, também estático, **find(:first)**. Exemplo:

```
Topic.first é equivalente ao Topic.find(:first)
```

Last

O método estático **last** é um alias para o, também estático, **find(:last)**. Exemplo:

```
Topic.last é equivalente ao Topic.find(:last)
```

USANDO OS MÉTODOS FIRST E LAST EM NAMED_SCOPE

Os métodos mencionados acima também funcionam em **named_scopes**. Imagine que eu criei um **named_scope** chamado **recent**, então eu poderei fazer isto:

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

EAGER LOADING

Para explicar esta nova funcionalidade vou precisar mostrar na prática. Vamos pegar como exemplo o código abaixo:

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

Estou fazendo uma pesquisa na tabela **authors**, mas incluindo na minha query as tabelas **posts** e **comments**, relacionado-as pela coluna **author_id**. Para entender melhor veja a *query* gerada pelo Rails:

```
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
```

Uma única query SQL foi criada contendo **joins** entre as tabelas **authors**, **posts** e **comments**. Chamamos isto de **produto cartesiano**.

Acontece que isto nem sempre é performático, por isto foi alterado. Nesta versão do Rails ao executar a mesma pesquisa na classe **Author**, o Rails usará uma outra estratégia para recuperar os dados das três tabelas. Ao invés de usar apenas uma query com todas as tabelas relacionadas, ele usará três querys menores, uma para cada tabela. Veja o resultado no log, após executar o mesmo código acima:

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

Na **"maioria"** dos casos executar três querys simples é mais rápido que executar uma única query gigante.

BELONGS_TO

O método **belongs_to** foi modificado para permitir o uso de **:dependent => :destroy** e **:delete** em associações.

Exemplos:

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

POLYMORPHIC URL

Os helpers para URLs polimórficas são métodos usados para resolver de uma forma mais inteligente uma rota nomeada quando você tem uma instancia de um modelo do **ActiveRecord**.

Estes métodos são úteis quando você quer gerar uma URL correta para um recurso **RESTful** sem precisar saber exatamente qual é o tipo do registro em questão.

O funcionamento é bem simples, veja alguns exemplos (nos comentários estão as chamadas equivalentes, nas versões antigas):

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

record = Comment.find(:first)
```

```
polymorphic_url(record)  #->  comment_url(record)

# ele também reconhece quando é um novo registro
record = Comment.new
polymorphic_url(record)  #->  comments_url()
```

Veja que o método reconhece o registro usado e monta a rota corretamente. **Nested resources** e **namespaces** também são suportados:

```
polymorphic_url([:admin, @article, @comment])
#-> vai devolver:
admin_article_comment_url(@article, @comment)
```

Você também pode usar prefixos como **new**, **edit** e **formatted**, veja alguns exemplos:

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

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

RELACIONAMENTOS COM READONLY

Uma nova opção está presente nos relacionamentos entre modelos. Afim de impedir que seus registros possam ser alterados, podemos usar a opção **:readonly** ao associar modelos. Veja alguns exemplos:

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

has_one :boss, :readonly => :true

belongs_to :project, :readonly => true
```

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

Feito isto, os registros protegidos não poderão ser alterados. Se você tentar, terá uma excessão do tipo **ActiveRecord::ReadOnlyRecord** disparada.

OS MÉTODOS ADD_TIMESTAMPS E REMOVE_TIMESTAMPS

Agora temos dois novos métodos: **add_timestamps** e **remove_timestamps**, que cria e remove (respectivamente) as colunas de **timestamps**. Veja um exemplo:

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

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

CALCULATIONS

O **ActiveRecord::Calculations** mudou um pouquinho para aceitar além do nome da coluna, também o nome da tabela. Isto é útil quando temos relacionamentos entre tabelas que contém uma ou mais colunas com o mesmo nome. Os métodos afetados são métodos como **sum** ou **maximum** do **ActiveRecord**. Resumindo, você pode fazer destas duas formas:

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

ACTIVERECORD::BASE.CREATE ACEITA BLOCOS

Já estamos acostumados com o método **ActiveRecord::Base.new** que aceita o uso de blocos na criação de um novo registro. Agora podemos fazer o mesmo com o método **create**:

```
# Criando um objeto, usando um bloco para informar seus atributos.
User.create(:first_name => 'Jamie') do |u|
  u.is_admin = false
end
```

Também podemos usar o mesmo método para criar vários objetos de uma vez:

```
# Criando um Array de novos objetos usando um bloco,
# onde o bloco é executado uma vez para cada objeto:
User.create([{:first_name => 'Jamie'}, {:first_name => 'Jeremy'}]) do |u|
  u.is_admin = false
end
```

Isto também funciona para associações:

```
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

Criar **migrations** ficou muito mais sexy depois do lançamento do Rails 2.0, mas alterar uma tabela usando **migrations** continuou sendo da forma antiga, nada sexy.

No Rails 2.1, alterar uma tabela também é sexy, com o novo método **change_table**. Veja um exemplo:

```
change_table :videos do |t|
  t.timestamps # adiciona as colunas created_at e updated_at
  t.belongs_to :goat # adiciona a coluna goat_id (integer)
  t.string :name, :email, :limit => 20 # adiciona duas colunas: name e email
  t.remove :name, :email # remove as colunhas name e email
end
```

Funciona como o **create_table**, mas ao invés de criar uma nova tabela, apenas altera uma tabela existente, adicionando ou removendo colunas e índices. Veja uma lista das opções existentes:

```
change_table :table do |t|
  t.column # cria uma coluna simples. Ex: t.column(:name, :string)
  t.index # Adiciona um novo índice à tabela
  t.timestamps
  t.change # muda a definição da coluna. Ex: t.change(:name, :string, :limit => 80)
  t.change_default # muda o valor padrão da coluna.
  t.rename # muda o nome da coluna
  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

Agora no Rails podemos rastrear alterações feitas no **ActiveRecord**. Podemos saber se um objeto foi alterado ou não e se foi alterado podemos identificar o que mudou e até mesmo fazer um comparativo do tipo antes e depois. Vamos aos exemplos:

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

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

# Recupera o valor anterior do atributo
article.title_was  #=> "Title"

# Veja o antes de depois da alteração
article.title_change  #=> ["Title", "New Title"]
```

Como você pode ver é bem simples. Você também pode listar todas as alterações no objeto de duas formas. Continuando do código anterior:

```
# Devolve uma lista com os atributos alterados
article.changed  #=> ['title']

# Devolve um Hash com os atributos alterados e um antes e depois
article.changes  #=> { 'title' => ["Title", "New Title"] }
```

Note que quando o objeto é salvo, o status dele é alterado. Veja:

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

Caso você vá alterar um objeto sem usar o operador **attr=**, você precisará informar manualmente que o atributo foi alterado usando o método **attr_name_will_change!** (no lugar de **attr**, vai o nome do atributo), veja mais um último exemplo:

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

PARTIAL UPDATES

A implementação de **Dirty Objects** foi o gancho para outra novidade também muito interessante.

Já que agora podemos rastrear quais atributos foram alterados na instancia do objeto, porque não usar isto para evitar atualizações desnecessários no banco de dados?

Nas versões anteriores do Rails quando executávamos o método **save** em um **ActiveRecord** já existente, era executado no banco de dados um **UPDATE** em todas as colunas da tabela, mesmo para as que não sofreram alterações.

Com o **Dirty Objects** isto poderia ser melhorado, e foi exatamente o que aconteceu. Veja o SQL gerado ao tentar salvar um registro que sofreu apenas uma leve alteração no Rails 2.1:

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

# Vamos atualizar um atributo…
article.title = "New Title"

# Veja o SQL criado ao persistir o objeto
article.save
#=>  "UPDATE articles SET title = 'New Title' WHERE id = 1"
```

Note que apenas o atributo alterado será atualizado no banco de dados. Se nenhum atributo fosse alterado, o **ActiveRecord** não executaria nenhum update.

Para habilitar/desabilitar esta nova funcionalidade você deve alterar a propriedade **partial_updates** dos seus **models**.

```
# Para habilitar a funcionalidade…
MinhaClasse.partial_updates = true
```

Se deseja habilitar/desabilitar esta funcionalidade para todos os models do seu sistema altere esta linha no arquivo *config/initializers/new_rails_defaults.rb*:

```
# Habilitando para todos os meus models
ActiveRecord::Base.partial_updates = true
```

Atenção: Não se esqueça de informar o *config/initializers/new_rails_defaults.rb* quando for alterar um atributo sem usar o método **attr=**, assim:

```
# Se eu fizer assim, tudo bem…
person.name = 'bobby'
person.name_change    # => ['bob', 'bobby']


# Mas se eu não alterar o atributo usando o sinal de '='
# então preciso avisar que vou fazer uma alteração
person.name_will_change!
person.name << 'by'
person.name_change    # => ['bob', 'bobby']
```

Se não fizer isto este tipo de alteração não será rastreado, e sua tabela não será atualizada corretamente.

SMALLINT, INT OU BIGINT NO MYSQL?

O adaptador de **MySQL** do ActiveRecord ficou um pouco mais esperto na hora de criar ou alterar colunas no banco de dados usando inteiros. De acordo com a opção **:limit**, ele define se a coluna será um **smallint**, **int** ou **bigint**. Veja um trecho do código que faz isto:

```
case limit
when 0..3
  "smallint(#{limit})"
when 4..8
  "int(#{limit})"
```

```
when 9..20
  "bigint(#{limit})"
else
  'int(11)'
end
```

Para ficar mais claro, vamos mapear isto em um **migration** e ver que tipo de coluna será criado para cada caso:

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

  # de 0 à 3: smallint
  t.integer :coluna1, :limit => 2 # smallint(2)

  # de 4 à 8: int
  t.integer :coluna2, :limit => 6 # int(6)

  # de 9 à 20: bigint
  t.integer :coluna3, :limit => 15 # bigint(15)

  # se a opção :limit não for informada: int(11)
  t.integer :coluna4 # int(11)
end
```

O adaptador do **PostgreSQL** já fazia assim, o do **MySQL** apenas seguiu a tendência.

OPCIONAL :SELECT NO HAS_ONE E BELONGS_TO

Os famosos métodos **has_one** e **belongs_to** acabam de receber mais uma opção, o já conhecido **:select**.

Por padrão esta opção é o "" *do **SELECT* FROM**, mas você pode mudar isto e recuperar somente as colunas que serão usadas, ou o que sua imaginação inventar.

Só um detalhe, não esqueça de colocar a **primary** e as **foreign keys**, senão você terá um lindo erro.

Outra alteração é que a opção **:order** do **belongs_to** foi removida. Mas não se preocupe, porque ela nem servia para nada mesmo.

ARMAZENANDO O NOME COMPLETO DA CLASSE AO USAR STI

Quando usamos **models** com **namespace** e **STI**, o **ActiveRecord** armazena apenas o nome da classe, sem o **namespace** (*demodulized*). Isto vai funcionar se todas as classes no **STI** estiverem no mesmo **namespace**, mas irá falhar em outros casos. Exemplo:

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

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

item2 = CollectionItem.find(item.id)
# retorna um erro, porque não encontrou a classe Item
```

Esta alteração adiciona uma nova opção de configuração que faz com que o **ActiveRecord** armazene o nome completo da classe.

Para ligar ou desligar esta funcionalidade você deve incluir ou alterar a seguinte linha no seu arquivo *environment.rb*:

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

Por padrão esta funcionalidade estará ligada.

MÉTODO TABLE_EXISTS?

Novo método para a classe **AbstractAdapter**: **table_exists?**. Seu uso é muito simples:

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

TIMESTAMPED MIGRATIONS

Quando se está estudando Rails ou desenvolvendo algo sozinho, **migrations** parecem ser a solução para todos os problemas. Mas quando se tem uma equipe trabalhando no mesmo projeto e todo mundo criando **migrations** ao mesmo tempo, você descobrirá (ou já descobriu) que **migrations** simplesmente não funcionam, pelo menos nas versões anteriores do Rails.

O problema é que quando se criava uma **migration**, ela recebia um número. Mas o que acontecia se duas pessoas criassem uma **migration** ao mesmo tempo, ou pior ainda, se várias pessoas começassem a criar **migrations** e só dessem commit depois? Você tinha um monte de **migrations** com o mesmo número com códigos diferentes. Conflito!

Já existia várias formas de "tentar" solucionar isto. Havía alguns plugins com abordagens diferentes para resolver este impasse. Mas independente do plugin ou abordagem que você usesse, uma coisa fica bem clara, a forma antiga simplesmente não funcionava.

Se você estivesse usando Git isto é pior ainda, porque provavelmente sua equipe teria alguns branches de trabalho e poderiam criar **migrations** em todos eles, e você teria os mesmo conflitos na hora de fazer o merge.

Por isto, o coreteam alterou a criação de migrations no Rails para usar não mais um número, mas uma string baseada na hora **UTC** no formato: YYYYMMDDHHMMSS.

Além disso foi criada uma nova tabela chamada **schema_migrations** que armazena quais **migrations** já foram executadas. Assim, se alguém criar uma migration com um número menor, será feito um **rollback** até a versão anterior e depois executado tudo de novo até a versão corrente.

Aparentemente isto resolve o problema de conflito de **migrations**.

Se você quiser, pode desligar esta funcionalidade incluindo esta linha no arquivo environment.rb:

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

Também foram criadas novas tarefas rake para "andar" pelos **migrations**:

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

Capítulo 3

ActiveSupport

Active Support é uma coleção de várias classes úteis e extensões de bibliotecas padrões, que foram considerados úteis para aplicações em Ruby on Rails. (wikipedia)

ACTIVESUPPORT::COREEXTENSIONS::DATE::CALCULATIONS

Time#end_of_day

Retorna o dia de hoje com o horário 23:59:59.

Time#end_of_week

Retorna o fim da semana (domingo 23:59:59).

Time#end_of_quarter

Retorna um Date representando o final do trimestre. Em outras palavras o última dia de março, junho, setembro, dezembro, o que vier primeiro.

Time#end_of_year

Retorna dia 31 de dezembro às 23:59:59

Time#in_time_zone

Este método é similar ao **Time#localtime**, exceto pelo fato de que usa o **Time.zone** no lugar do fuso-horário do sistema operacional. Você pode passar como parâmetro um **TimeZone** ou uma **String**. Vejamos alguns exemplos:

```
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

Foi corrigido um bug no método **days_in_month** que informava o número de dias no mês de fevereiro de forma errônea quando o ano não era informado.

A alteração consiste em usar o ano corrente quando não se informa um ano ao chamar o método. Supondo que você esteja em um ano bissexto, veja:

```
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

A classe **DateTime** ganhou um novo método o **to_f** que retorna a data como um ponto flutuante que representa a quantidade de segundos desde o Unix epoch (época Unix). Isto é, a quantidade de segundos desde 1 de janeiro de 1970 às zero hora.

Date.current

A classe **Date** ganhou um método **current** que deve ser usado com substituto do **Date.today**, pois leva em conta o fuso-horário caso o **config.time_zone** tenha sido configurado, retornando um **Time.zone.today**. Caso não tenha sido configurado ele retornará um **Date.today**.

FRAGMENT_EXIST?

Dois novos métodos foram adicionados ao **cache_store**: **fragment_exist?** e **exist?**.

O método **fragment_exist?** faz exatamente o que promete, verifica se um cached fragment, informado através de uma chave existe. Básicamente substituindo o famoso:

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

O método **exist?** foi adicionado ao **cache_store**, enquanto que o **fragment_exist?** é um helper para que você vai poder usar no seu controller.

UTC OU GMT?

Uma alteração simples, mas interessante. Até agora o Rails tem usado muito a sigla UTC, mas quando se executa o método **to_s** do objeto **TimeZone** ele mostrará GMT e não UTC. Isto se dá porque a sigla GMT é mais familiar para o usuário final.

Se você olhar no painel de controle do Windows, onde você escolhe o fuso-horário, verá que a sigla usada é GMT. Google e Yahoo também usam GMT em seus produtos.

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

JSON ESCAPE

O método **json_escape** funciona como o **html_escape**. É muito útil para quem precisa exibir strings **JSON** em uma página **HTML**, como no caso de uma documentação, por exemplo.

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

Também podemos usar o atalho **j** no ERB:

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

Se quiser que todo o código **JSON** seja "escapado" por padrão, incluía a seguinte linha no seu arquivo *environment.rb*:

```
ActiveSupport.escape_html_entities_in_json = true
```

MEM_CACHE_STORE AGORA ACEITA OPÇÕES

A inclusão do **Memcache-Client** no **ActiveSupport::Cache** facilitou muito as coisas, mas também removeu a flexibilidade, não deixando personalizar mais nada além do IP do servidor do **memcached**.

Jonathan Weiss criou um patch, que foi incluído no Rails, incluindo opções extras, como estas:

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

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

ou

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

TIME.CURRENT

Novo método para a classe **Time**. O retorno do método **current** depende do **config.time_zone**, se ele foi especificado antes, o método retornará um **Time.zone.now**, caso contrário será um **Time.now**.

```
# o retorno depende do config.time_zone
Time.current
```

Os métodos **since** e **ago** também tiveram seus retornos alterados, devolvendo um **TimeWithZone** caso o **config.time_zone** tiver sido especificado.

Isto torna o método **Time.current** o novo método padrão para se recuperar a hora atual, substituindo o **Time.now** (que continua existindo, mas não leva em conta o fuso-horário especificado).

Os métodos **datetime_select**, **select_datetime** e **select_time** também foram atualizados para terem seus valores default como **Time.current**.

REMOVENDO ESPAÇOS EM BRANCO COM O MÉTODO SQUISH

Dois novos métodos foram acrescentados ao objeto **String**, o **squish** e o **squish!**.

Estes métodos fazem o mesmo que o método **strip**, removendo espaços em branco do inicio e fim do texto, mas além disso também arrumam casos onde no meio do texto temos mais de um espaço deixando com apenas um.

Veja um exemplo:

```
"   Um   texto   cheio   de   espaços   ".strip
#=> "Um   texto   cheio   de   espaços"
```

```
“    Um    texto    cheio    de    espaços    “.squish
#=> “Um texto cheio de espaços”
```

Capítulo 4

ActiveResource

O ActiveResource é uma camada de mapeamento responsável pela implementação do lado cliente de sistemas RESTful. Através do ActiveResource é possível consumir serviços RESTful através do uso de objetos que funcionam como um proxy para serviços remotos.

USANDO E-MAIL COMO NOME DE USUÁRIO.

Alguns serviços usam o e-mail como nome do usuário, o que nos obrigaria a usar uma URL mais ou menos assim:

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

Mas isto gerava um problema, porque temos dois arrobas (@) e o interpretador se perde para entender isto. Por este motivo a forma de usar o **ActiveResource** foi estendida um pouco, afim de facilitar o uso de emails na autenticação. Agora você também pode fazer assim:

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

O MÉTODO CLONE

Agora poderemos clonar um resource existente da seguinte forma:

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

Só vale tomar nota que a cópia não clona nenhum atributo da classe, apenas os atributos do resource.

```ruby
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

O Active Resource usa HTTP para acessar APIs RESTful e por isto está suscetível a problemas de lentidão ou servidores fora do ar. Em alguns casos, suas chamadas ao ActiveResource podem expirar (timeout). Agora você pode controlar o tempo de expiração com a propriedade timeout.

```
class Person < ActiveResource::Base
  self.site = "http://api.people.com:3000/"
  self.timeout = 5 # espera 5 segundos antes de expirar
end
```

Neste exemplo foi configurado o tempo de timeout para 5 segundos. É recomendado que este valor seja baixo, para permitir que seu sistema falhe rápido (ou fail-fast), impedindo falhas em cascata que poderiam incapacitar o seu servidor.

Internamente o Active Resource se baseia na biblioteca Net::HTTP para fazer requests HTTP. Quando você define um valor para a propriedade timeout, o mesmo valor é definido para o read_timeout da instância do objeto Net::HTTP que está sendo usado.

O valor padrão é de 60 segundos.

Capítulo 5

ActionPack

Compreende o Action View (geração de visualização de usuário, como HTML, XML, JavaScript, entre outros) e o Action Controller (controle de fluxo de negócio). (wikipedia)

TIMEZONE

Definindo um fuso-horário padrão

Uma nova opção foi acrescentada ao método **time_zone_select**, agora você pode indicar um valor padrão para os casos em que o seu usuário ainda não tenha selecionado nenhum **TimeZone**, ou quando a coluna no banco de dados for nula. Para isto foi criada a opção **:default**, então você poderá usar o método das seguintes maneiras:

```
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)")
```

Nos casos onde usamos a opção **:default** deve aparecer com o **TimeZone** informado já selecionado.

O método formatted_offset

O método **formatted_offset** foi incluído nas classes **Time** e **DateTime** para retornar no formato **+HH:MM** o desvio da hora UTC. Por exemplo, em nosso fuso-horário (hora de Brasília) o desvio retornado pelo método seria uma string com o valor **"-03:00"**.

Vamos aos exemplos:

Recuperando o desvio a partir de um DateTime:

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

Agora a partir de um Time:

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

Note que este método retorna uma **string**, que pode ser formatada ou não dependendo do valor passado como parâmetro.

O método with_env_tz

O método **with_env_tz** permite realizar testes com fusos-horários diferentes de um uma forma bem simples:

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

Este helper era para se chamar **with_timezone**, mas foi renomeado para **with_env_tz** para evitar uma confusão com o fuso-horário informado via **ENV['TZ']** e **Time.zone**.

Time.zone_reset!

Esse método foi removido poque não estava mais sendo usado.

Time#in_current_time_zone

Esse método foi modificado para retornar **self** quando **Time.zone** for nulo.

Time#change_time_zone_to_current

Esse método foi modificado para retornar **self** quando **Time.zone** for nulo.

TimeZone#now

o método **TimeZone#now** foi alterado para retornar um **ActiveSupport::TimeWithZone** representando a hora corrente no fuso horário configurado no **Time.zone**. Exemplo:

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

Compare_with_coercion

Foi criado o método **compare_with_coercion** (com um alias para <=>) nas classes **Time** e **DateTime**, tornando possível realizar uma comparação cronológica entre as classes **Time**, **DateTime** e instâncias do **ActiveSupport::TimeWithZone**. Para entender melhor como funciona, veja os exemplos abaixo (o resultado de cada linha está no comentário logo depois do código):

```
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?

Foi incluído o método between? na classe TimeWithZone para verificar se a instância está entre duas datas. Exemplo:

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

TimeZone#parse

Este método cria uma nova instância de **ActiveSupport::TimeWithZone** à partir uma string. Exemplos:

```
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

Esse método serve para criar uma nova instância de **ActiveSupport::TimeWithZone** à partir do número de segundos desde o Unix epoch. Exemplo:

```
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
```

Mais métodos

Os métodos **to_a**, **to_f**, **to_i**, **httpdate**, **rfc2822**, **to_yaml**, **to_datetime** e **eql?** foram adicionados na classe TimeWithZone. Para maiores informações sobre esses métodos verifique na documentação do **Rails**

TimeWithZone se preparando para o Ruby 1.9

No Ruby 1.9 teremos alguns métodos novos na classe **Time**, métodos como:

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

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

Existe um para cada dia da semana.

Outra curiosidade é que o método **to_s** do objeto **Time** também vai ter um retorno um pouco diferente. Hoje quando executamos **Time.new.to_s**, temos o seguite:

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

No Ruby 1.9 teremos:

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

O que isto tem há ver com Rails 2.1? Tudo, já que o Rails já está sendo preparado para lidar com estas alterações. A classe **TimeWithZone**, por exemplo, acabou de receber uma implementação para funcionar com os métodos do primeiro exemplo.

AUTO LINK

Para quem não conhece, o método **auto_link** recebe um texto qualquer como parâmetro e se o texto tiver algum endereço de email ou de um site ele retorna o mesmo texto com hyperlinks.

Por exemplo:

```
auto_link("Acesse este endereço: http://www.rubyonrails.com")
# => Acesse este endereço: http://www.rubyonrails.com
```

Acontece que alguns sites como o Amazon estão usando também o sinal de "=" (igual) em suas URLs, e este método não reconhece este sinal. Veja como o método se comporta neste caso:

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

Note que o método terminou o hyperlink exatamente antes do sinal de "=", pois ele não suporta este sinal. Quer dizer, não suportava. Nesta nova versão do Rails já temos este problema resolvido.

O mesmo método foi alterado mais tarde para também permitir o uso de URLs com o sinal de parênteses.

Um exemplo de URL com parênteses:

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

RÓTULOS

Ao criar um novo formulário usando **scaffold** ele será criado com o seguinte código:

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

Desta forma faz muito mais sentido. O método **label** foi incluído. Este método retorna uma *string* com o título da coluna dentro de uma tag HTML **<label>**.

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

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

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

Percebeu o parâmetro **for** dentro da tag? O "post_title" é o nome da caixa de texto que contém o título do nosso post. A tag **<label>** é na verdade um rótulo associado ao objeto **post_title**. Quando se clica no rótulo (ele não é um link) o controle associado à ele recebe o foco.

Robby Russell escreveu um artigo interessante em seu blog sobre este assunto. Você pode lê-lo no endereço: http://www.robbyonrails.com/articles/2007/12/02/that-checkbox-needs-a-label

Também foi incluído o método **label_tag** no **FormTagHelper**. Este método funciona exatamente como o label mas de uma forma mais simplista:

```
>> label_tag 'nome'
=> <label for="nome">Nome</label>

>> label_tag 'nome', 'Seu nome'
=> <label for="nome">Seu Name</label>

>> label_tag 'nome', nil, :class => 'small_label'
=> <label for="nome" class="small_label">Nome</label>
```

O método também aceita a opção **:for**, veja um exemplo:

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

Isto vai retornar algo assim:

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

UMA NOVA FORMA DE USAR PARTIALS

Algo muito normal no desenvolvimento de softwares em Rails é o uso de **partials** para evitar a repetição de código. Vejamos um exemplo de seu uso:

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

Partial é um fragmento de código (um template). A vantagem de se usar uma **partial** é evitar a repetição desnecessária de código. Para usar uma **partial** é muito simples, você pode começar com algo mais ou menos assim: **render :partial => "name"**. Depois deve criar um arquivo com o mesmo nome da **partial**, mas com um underscore na frente, só isso.

O código acima é a forma como estamos acostumados a fazer hoje, mas nesta nova versão do Rails, faremos a mesma coisa de uma forma um pouco diferente, assim:

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

Neste exemplo nós vamos renderizar a partial "users/_form", que receberá uma variável chamada form com as referências criadas pelo **FormBuilder**.

A forma antiga também vai continuar funcionando.

NOVOS NAMESPACES NO ATOM FEED

Conhece o método **atom_feed**? Ele é uma novidade no Rails 2.0, que facilitou muito a criação de feeds Atom. Veja um exemplo de uso:

Em um arquivo *index.atom.builder*:

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

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

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

O que é um atom feed? Atom é o nome de um estilo baseado em XML e meta data. Em outras palavras é um protocolo quer serve para publicar conteúdo na internet que é sempre atualizado, como um blog, por exemplo. Os feeds sempre são publicados em XML e no caso do Atom Feed ele é identificado como application/atom+xml media type.

Nas primeiras versões do Rails 2.0 este método aceitava como parâmetros as opções **:language**, **:root_url** e **:url**, você pode obter mais informações sobre estes métodos na documentação do Rails. Mas com a alteração realizada, agora podemos incluir novos namespaces ao elemento root do feed. Por exemplo, se fizermos assim:

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

Ele retornará isto:

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

Adaptando o exemplo anterior, poderíamos usá-lo assim:

```
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

Todos os métodos **fragment_cache_key** agora retornam por padrão o namespace 'view/' como prefixo.

Todos os caching stores foram retirados de **ActionController::Caching::Fragments::** *e agora estão em* ***ActiveSupport::Cache::***. Neste caso se você faz referência a um store, como **ActionController::Caching::Fragments::MemoryStore**, por exemplo, será necessário alterar sua referência para **ActiveSupport::Cache::MemoryStore**.

ActionController::Base.fragment_cache_store deixa de existir e dá lugar à **ActionController::Base.cache_store**.

Foi incluído no **ActiveRecord::Base** o método **cache_key** para facilitar o armazenamento em cache de Active Records pelas novas bibliotecas **ActiveSupport::Cache::***. Este método funciona assim:

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

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

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

Foi incluído o **ActiveSupport::Gzip.decompress/compress** para facilitar o wrapper para o **Zlib**.

Agora você pode usar entre as opções de environment o **config.cache_store** para informar o local padrão de armazenamento do cache. Vale lembrar que se o diretório **tmp/cache** existir o padrão é o **FileStore**, caso contrário o **MemoryStore** é usado. Você pode configurar das seguintes formas:

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

Para facilitar as coisas, foi incluído o comentário abaixo no arquivo *environments/production.rb*, afim de lembrá-lo desta opção.

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

APLICANDO FORMATAÇÃO DE TÍTULO EM STRINGS

Existia um bug ao se utilizar o método **String#titleize** em uma string que continha 's. O bug retornava o 's maiúsculo. Veja um exemplo:

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

Veja como ficou o mesmo exemplo só que agora com a correção desse bug.

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

ACTION_NAME

Agora para saber qual action foi chamada durante a execução de sua view ficou mais fácil, basta usar o método **action_name**:

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

O retorno será o mesmo que pegar o **params[:action]**, mas de uma forma mais elegante.

CACHES_ACTION ACEITA CONDICIONAIS

O método **caches_action** agora também aceita a opção **:if**, permitindo o uso de condicionais para especificar quando uma **action** pode ir para o **cache**. Por exemplo:

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

No exemplo acima, a **action index** só irá para o **cache** se não foi acessada via um request JSON.

CONDICIONAL NO MÉTODO CACHES_PAGE

O método **caches_page** ganhou uma opção para condicionar seu uso (**:if**). Veja um exemplo:

```
# forma atual
caches_page :index

# No Rails 2.1 você pode usar a opção :if
caches_page :index, :if => Proc.new { |c| !c.request.format.json? }
```

FLASH.NOW AGORA FUNCIONA EM TESTES

Quem nunca quebrou a cabeça por causa disto? O problema é que durante os testes você nunca podia confirmar se uma mensagem foi armazenada no flash, porque ela era limpa pelo Rails antes de cair no seu script de teste.

No Rails 2.1 este problema foi resolvido. Agora você pode incluir linhas como esta em seus testes:

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

ACESSANDO HELPERS FORA DAS VIEWS

Quantas vezes você não criou um **helper** e depois quis usá-lo dentro do meu **controller**? Para conseguir isto só incluindo o módulo do **helper** no **controller**, mas isto sempre deixa seu código com aparência de sujo.

Para o Rails 2.1 foi criado um proxy para acessar os helpers de fora das views. Funciona de maneira muito simples:

```
# Para acessar o método simple_format, por exemplo
ApplicationController.helpers.simple_format(text)
```

Simples e limpo!

JSON

Rails agora aceita POSTs de conteúdo JSON. Por exemplo, você pode enviar uma requisição POST assim:

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

E o tudo vai para dentro da variável **params**. Isto aqui funciona, por exemplo:

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

Para quem não sabe o JSON é um "concorrente" do XML, e é muito usado para tráfego de dados em JavaScript, porque é representado nesta linguagem. Daí o seu nome: **JavaScript Object Notation**.

PATH NAMES

Os leitores do meu blog (http://www.nomedojogo.com) devem conhecer o meu plugin **Custom Resource Name**. Acho que ele pode estar com seus dias contados... :(

No Rails você já podia incluir a opção **:as** na suas rotas (coisa que fiz questão de implementar igual no meu plugin, para manter a compatibilidade), agora você também terá a opção **:path_names** para alterar os nomes das **actions**.

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

Claro que meu plugin ainda continua sendo útil para usuários de versões anteriores do Rails.

DEFININDO A LOCALIZAÇÃO DE SEU ARQUIVO DE ROTAS

No Rails 2.1 você pode definir em que arquivo estão suas rotas incluindo a seguinte linha no seu arquivo *environment.rb*:

```
config.routes_configuration_file
```

Isto pode ser útil em um cenário onde você tem dois front-ends separados que usam os mesmos módulos, bibliotecas e plugins.

Por exemplo, getsatisfaction.com e api.getsatisfaction.com compartilham os mesmos modelos, mas não os controllers, helpers e views. getsatisfaction.com tem o seu arquivo de rotas com otimizações para melhorar o seu SEO, enquanto o arquivo de rotas da API não precisaria de nada disto.

SESSION(:ON)

Talvez você não saiba mas no Rails é possível desligar o uso de sessões assim:

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

Veja que no exemplo acima estou desligando a sessão para todos os controllers (**ApplicationController**), mas eu também poderia fazer isto para apenas um controller específico.

Mas e se eu quiser que um controller "ligue" o uso de sessões para ele? No Rails 2.1 o método aceita a opção **:on**, assim:

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

TESTANDO HELPERS DE FORMA SIMPLES

Uma coisa muito chata de ser fazer nas versões anteriores do Rails era testar **helpers**. Eu mesmo já sofri um bocado para garantir 100% de cobertura, criando testes para alguns **helpers**.

Isto ficou muito simples no Rails 2.1 com a nova classe **ActionView::TestCase**. Vamos pegar o código abaixo como exemplo:

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

  def homepage_path
    people_path
  end
end
```

Veja como faremos para testar isto no 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
```

Capítulo 6

ActionController

O ActionController é a camada responsável por receber as requisições web e de tomar as decisões do quê será executado e renderizado ou de redirecionar para outra ação. Uma ação é definido como métodos públicos nos controladores que são automaticamente disponíveis através das rotas.

ACTIONCONTROLLER::ROUTING

Map.root

Agora na hora de informar o **map.root** você pode ser um pouco mais **DRY** usando um alias para isto.

Nas versões anteriores do Rails você fazia assim:

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

Agora você pode fazer assim:

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

Reconhecimento de rotas

A antiga implementação do reconhecimento de rotas percorria todas as rotas, uma-a-uma, e isto consumia muito tempo. Uma nova implementação mais inteligente foi desenvolvida onde se cria uma árvore de rotas e o reconhecimento é feito pelo prefixo, pulando rotas semelhantes. Isto já diminui em aproximadamente 2.7 vezes o tempo de reconhecimento.

Toda a nova implementação está no arquivo **recognition_optimisation.rb** e os detalhes de seu funcionamento estão bem explicados nos comentários. Para mais informações sobre isto veja a documentação no próprio código fonte.

Assert_routing

Agora é possível testar uma rota com um método HTTP. Veja o exemplo:

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

Map.resources

Imagine que você tem um site todo em português e quer deixar suas rotas no mesmo idioma. Em outras palavras ao invés de ter algo como:

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

Você gostaria de ter algo assim:

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

Isto já era possível, mas não de uma forma muito simples, não pelo menos sem comprometer algumas convenções do Rails.

Agora ganharemos a opção **:as** no **map.resources** para personalizar nossas rotas. Veja um exemplo, para conseguir a URL acima totalmente em português:

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

ACTIONCONTROLLER::CACHING::SWEEPING

Nas versões anteriores, quando íamos declarar um **sweeper** tínhamos de informar a classe usando símbolos, assim:

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

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

Agora é possível declarar explicitamente uma classe ao invés de usar um símbolo. Isto é necessário se o seu **sweeper** estiver em um módulo, por exemplo. Embora você possa continuar usando símbolos para os demais casos, a partir de agora você também pode fazer assim:

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

Capítulo 7

ActionView

O ActionView é a camada responsável pela geração da interface visível ao usuário através da conversão dos templates ERB.

ACTIONVIEW::HELPERS::FORMHELPER

fields_for form_for com a opção index.

Os métodos **#fields_for** e **form_for** receberam a opção **:index**, removendo a necessidade de usar o **:index => nil** em cada objeto do formulário. Veja os exemplos:

O código ficava assim:

```
<% 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 %>
```

Agora fica assim:

```
<% 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

Agora todos os métodos desse módulo que tratam datas (**date_select**, **time_select**, **select_datetime**, etc.) aceitam HTML options. Veja um exemplo usando o **date_select**

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

date_helper

O método date_helper foi alterado para usar o **Date.current** para determinar o seu valor padrão.

ACTIONVIEW::HELPERS::ASSETTAGHELPER

register_javascript_expansion

Este método registra um ou mais arquivos javascript para serem incluídos quando um símbolo, determinado por você, for passado como parâmetro para o método **javascript_include_tag**. A idéia é que este método seja chamado no arquivo **init.rb** do seu plugin para registrar os arquivos javascript que seu plugin instalou na pasta **public/javascripts**. Veja como funciona:

```
# No arquivo init.rb
ActionView::Helpers::AssetTagHelper.register_javascript_expansion
    :monkey => ["head", "body", "tail"]

# Depois, fazendo assim:
javascript_include_tag :monkey

# Teremos isto:
<script type="text/javascript" src="/javascripts/head.js"></script>
<script type="text/javascript" src="/javascripts/body.js"></script>
<script type="text/javascript" src="/javascripts/tail.js"></script>
```

register_stylesheet_expansion

Este método faz exatamente a mesma coisa que o método **ActionView::Helpers::AssetTagHelper#register_javascript_expansion**, mas criando um símbolo para ser usado nas chamadas ao método **stylesheet_link_tag**. Veja um exemplo:

```
# No arquivo init.rb
ActionView::Helpers::AssetTagHelper.register_stylesheet_expansion
```

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

# Depois, fazendo assim:
stylesheet_link_tag :monkey

# Teremos isto:
<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

Foi adicionada a opção **:confirm** nos parametros do método **#submit_tag**. Essa opção funciona exatamente como no método **link_to**. Veja um exemplo:

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

ACTIONVIEW::HELPERS::NUMBERHELPER

number_to_currency

O método **number_to_currency** passou a aceitar a opção **:format** como parâmetro, permitindo a formatação do valor retornado pelo método. Nas versões anteriores, na hora de formatarmos valores em reais precisamos incluir um espaço na frente da opção **:unit** para que o formato ficasse correto. Veja os exemplos:

```
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"
```

Além disto, podemos personalizar de outras formas, por exemplo:

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

Ao montar sua própria string de formatação, você pode usar os seguintes parâmetros:

```
%u Para a unidade monetária
%n Para o número
```

ACTIONVIEW::HELPERS::TEXTHELPER

excerpt

O método **excerpt** é um helper para localizar uma palavra dentro de uma frase e retornar uma abreviação desta frase com o número de caracteres passado como parâmetro antes e depois da palavra, acrescentando, quando necessário, "…". Veja este exemplo:

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

O problema é que existia um BUG. Se você contar verá que depois a palavra ele trouxe na verdade seis caracteres e não cinco. Esse bug foi corrigido. Veja o exemplo do funcionamento correto desse método:

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

simple_format

O método **simple_format** basicamente recebe como parâmetro um texto qualquer e o formata de uma forma simples em HTML. O que ele faz é pegar o texto e substituir uma quebra de linha (\n) pela tag "< br />". E quando temos duas quebras de linha seguidas (\n\n) ele separa o texto entre parágrafos, usando a tag "< p>".

No Rails 2.1 este método recebe um novo parâmetro opcional. Além do texto poderemos também informar quais atributos HTML gostaríamos que a tag "< p>" tivesse. Vamos aos exemplos:

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

Os atributos HTML informados serão acrescentados em todas as tags "< p>" criadas pelo método.

Capítulo 8

Railties

CONFIG.GEM

Agora é possível configurar todas as gems necessárias para o funcionamento correto de um projeto através do novo recurso **config.gem**. No arquivo **environment.rb** você pode adicionar quais gems seu projeto depende, como nesse exxemplo:

```
config.gem "bj"

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

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

Pode instalar todas as dependências de uma só vez basta usar o comando:

```
# Instala todos os gems listados
rake gems:install
```

Também é possível listar quais gems estão sendo usados no projeto executando:

```
# Lista todos os gems dependentes
rake gems
```

E se algum dos gems tiver um arquivo **rails/init.rb** e você quiser levar o gem junto com sua aplicação, rode:

```
# Copia o gem especifícado para vendor/gems/nome_do_gem-x.x.x
rake gems:unpack GEM=nome_do_gem
```

E o gem será copiado para o diretório **vendor/gems/gem_name-x.x.x**. Caso você não especifique o nome do gem o Rails irá copiar todos os gems para o diretório **vendor/gem**

CONFIG.GEM EM PLUGINS

O recurso do **config.gem** também está disponível para uso com plugins.

Até a versão 2.0 o arquivo **init.rb** de um plugin se parecia com isto:

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

Mas no Rails 2.1 o arquivo **init.rb** seria:

```
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
```

Assim, quando você rodar a tarefa para instalar todos os gems necessários, estes gems estarão entre eles.

GEMS:BUILD

A task **gems:build** compila todas as extensões nativas do gems que foram instaladas através do gems:unpack. A sintaxe é:

```
rake gems:build # Para todas as gems
rake gems:build GEM=mygem # Estou informando a gem
```

NOVA MENSAGEM AO INICIAR O SERVIDOR

Ouve uma pequena alteração na mensagem de inicio do Rails, agora ela mostra também qual é a versão do Rails que está sendo carregada:

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

RAILS.PUBLIC_PATH

Foi incluído no Rails o atalho **Rails.public_path** para recuperar o caminho do diretório **"public"** do projeto.

```
Rails.public_path
```

RAILS.LOGGER, RAILS.ROOT, RAILS.ENV E RAILS.CACHE

No Rails 2.1 ao invés de usarmos as constantes: **RAILS_DEFAULT_LOGGER**, **RAILS_ROOT**, **RAILS_ENV** e **RAILS_CACHE**. Usaremos os atalhos:

```
# RAILS_DEFAULT_LOGGER
Rails.logger

# RAILS_ROOT
Rails.root

# RAILS_ENV
Rails.env

# RAILS_CACHE
Rails.cache
```

RAILS.VERSION

Nas versões anteriores para descobrir, durante a execução do seu código, qual é a versão do Rails em uso, bastava usar:

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

No Rails 2.1 isso mudou para:

```
Rails.version
```

OBTENDO INFORMAÇÕES SOBRE UM PLUGIN

Aí está uma das novas funcionalidades do Rails 2.1 que provavelmente você nunca usará. Digo "talvez", porque pode ser que em algum caso muito especifico seja interessante obter o número da versão do plugin, por exemplo.

Para testar isto, precisamos criar um novo arquivo chamado *about.yml* no diretório do plugin, algo mais ou menos assim:

```
author: Carlos Brando
version: 1.2.0
description: Uma descrição qualquer sobre o plugin
url: http://www.nomedojogo.com
```

Depois podemos recuperar estas informações via código, assim:

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

Se alguém encontrar algum uso para isto e quiser compartilhar comigo, talvez eu mude de idéia quanto à sua real necessidade.

Capítulo 9

Rake Tasks, Plugins e Scripts

TASKS

rails:update

A partir de agora toda vez que se executar a tarefa **rake rails:freeze:edge** também será executado o **rails:update**, atualizando os arquivos de configuração e *JavaScript*.

Banco de dados em 127.0.0.1

Foi feita uma alteração no arquivo databases.rake que antes só considerava um banco de dados local como estando em localhost para considerar também o IP **127.0.0.1**. Isto funciona tanto para a tarefa **create** como para **drop**. O arquivo databases.rake também foi refeito para tornar o código menos repetitivo.

Congelando um release específico do Rails

Até ao Rails 2.1 não era possível congelar o Rails em seu projeto pela versão, somente pela revisão. No Rails 2.1, poderemos congelar um release específico com o comando abaixo:

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

TIMEZONE

rake time:zones:all

Retorna todos os time zones que o Rails reconhece, agrupados por offset. Você também pode filtrar o retorno usando o parâmetro opcional OFFSET, por exemplo: OFFSET=-6.

rake time:zones:us

Exibe uma lista com todos os time zones dos USA. A opção OFFSET também vale aqui.

rake time:zones:local

Retorna os time zones que o Rails conhece que estão no mesmo offset do seu sistema operacional.

SCRIPTS

plugin

O comando script/plugin install agora permite o uso da opção –e/--export, para que ele use um svn export. Foi adicionado o suporte a plugins hospedados em repositórios GIT.

dbconsole

Esse script faz a mesma coisa que o script/console mas para o banco de dados. Em outras palavras ele entra no cliente de linha de comandos do banco de dados.

Analisando o código, aparentemente isto só vai funcionar para bancos de dados mysql, postgresql, sqlite(3). Quando um banco diferente estiver configurado no arquivo database.yml e este comando for executado, a mensagem "not supported for this database type" será exibida no terminal.

PLUGINS

Gems podem ser plugins

Agora, qualquer gem que possua um arquivo **rails/init.rb** na sua árvore de diretório pode ser instalado dentro do diretório **vendor** do seu projeto Rails como se fosse um **plugin**.

Usando generators em plugins

É possível configurar o **Rails** para procurar por **plugins** em outro lugar diferente do diretório **vendor/plugins**, apenas incluindo uma linha de código no arquivo **environment.rb**.

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

Porém na versão 2.0 do Rails, havia um bug com esta configuração que se manifestava quando o plugin tinha generators. Por causa desse bug o Rails só encontrava generators em plugins que estivessem no diretório **vendor/plugins**. Na versão 2.1 esse bug foi corrigido.

Capítulo 10

Prototype e script.aculo.us

PROTOTYPE

O Rails passa a usar a partir de agora a versão 1.6.0.1 do Prototype. Isto serve como um preparatório para a versão 1.8.1 do script.aculo.us.

Capítulo 11

Ruby 1.9

DETALHES

O principal foco das alterações do Rails foi o Ruby 1.9, mesmo os menores detalhes foram analisados para deixar o Rails o mais compatível possível com a nova versão do Ruby. Detalhes como alterar de **File.exists?** para **File.exist?** não foram deixados de fora.

Também, no Ruby 1.9, o módulo **Base64** (base64.rb) foi removido, por isto todas as referencias a ele foram substituídas por **ActiveSupport::Base64**.

NOVOS MÉTODOS PARA A CLASSE DATETIME

Para manter a compatibilidade (duck-typing) com a classe **Time**, três métodos novos foram adicionados à classe **DateTime**. Os métodos são **#utc**, **#utc?** e **#utc_offset**. Vamos ver um exemplo de uso de cada um:

```
>> 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
```

Capítulo 12

Debug

RUBY-DEBUG NATIVO

Foi habilitada novamente a opção de usar o **ruby-debug** nos testes do Rails. Agora basta usar o método **debugger**, desde que você já tenha o gem instalado.

Capítulo 13

Bugs e Correções

ADICIONAR COLUNAS NO POSTGRESQL

Havia um bug ao se usar o banco de dados **PostgreSQL**. O bug ocorria quando se criava uma migration para adicionar uma coluna em uma tabela já existente, veja um exemplo:

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

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

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

```
    end
  end
```

Note que estamos criando uma coluna com **:precision => 6** e **:scale => 2**. Agora é hora de rodar o **rake db:migrate** e vamos ver como ficou nossa tabela no banco:

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

Veja a coluna "cost" que acabamos de criar. Ela é um **numeric** comum, mas deveria ser uma coluna como a "price", logo acima dela, mais precisamente um **numeric(6,2)**. No Rails 2.1 este erro não existe mais, a coluna será criada da forma correta.

MIME TYPES

Foi corrigido um bug que não permitia determinar o mime type atribuido ao **request.formt** usando um símbolo. Agora o código abaixo já pode ser utilizado:

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

CORREÇÃO DE BUG NO CHANGE_COLUMN

Foi corrigido um bug existente quando se tentava usar o método **change_column** com **:null => true** em uma coluna que foi criada usando **:null => false**. Por causa deste bug nenhuma alteração era feita.

Capítulo 14

Informações Adicionais

PROTEGENDO-SE DE CROSS SITE SCRIPTING

No Rails 2.0 o arquivo *application.rb* ficou desta maneira:

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

  protect_from_forgery
end
```

Note a chamada para o método **protect_from_forgery**.

Já ouviu falar de Cross Site Scripting? Este é o nome de uma falha de segurança encontrada facilmente em grande parte dos websites e aplicações web que permite à pessoas maldosas (aqui estou me referindo à adolescentes sem nada para fazer e

sem vida social) alterarem o conteúdo de páginas web, incluírem conteúdo hostil, executarem ataques de phishing, obterem o controle do navegador através de códigos JavaScript e na maioria dos casos forçarem o usuário a executar algum comando que eles desejem. Este último tipo de ataque se chama crosssite request forgeries.

O Cross Site Request Forgeries é um tipo de ataque que consiste em obrigar usuários legítimos a executarem uma série de comandos sem nem mesmo saberem disto. E agora com o aumento do uso de Ajax, a coisa tem ficado ainda pior.

Na verdade, este método serve para nos assegurar de que todos os formulários que sua aplicação está recebendo estão vindo dela mesma, e não de um link perdido de algum outro site. Ele consegue isto incluindo um token baseado na sessão em todos os formulários e requisições Ajax geradas pelo Rails, e depois verifica a autenticidade deste token no controller.

Lembre-se que requisições via GET não são protegidas. Mas isto não será um problema se somente à usarmos para nos trazer dados, e nunca para alterar ou gravar algo em nosso banco de dados.

Se quiser aprender mais sobre CSRF(Cross-Site Request Forgery) use os endereços abaixo:

- 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

Mas lembre-se que isto não é uma solução definitiva para nosso problema, ou como costumamos dizer, não é uma bala de prata.

USOU O METHOD_MISSING, ENTÃO NÃO DEIXE PONTAS SOLTAS

Devido a natureza dinâmica do Ruby, o método **respond_to?** é crucial. Quantas vezes não precisamos checar se um método existe no objeto que estamos manipulando, ou mesmo verificar se o objeto é mesmo aquele que estamos esperando (**is_a?**)?

Porém tem algo muito importante que muita gente se esquece. Veja por exemplo esta minha classe que faz uso do método **method_missing**:

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

rex = Cachorro.new
rex.latir #=> auau!
rex.latir! #=> auau!
rex.latir_e_correr #=> auau!
```

Acho que você já deve conhecer o **method_missing**, não? Veja que no exemplo acima eu estou criando uma instância da classe **Cachorro** e chamando os métodos **latir**, **latir!** e **latir_e_correr** que não existem. Por isto o método **method_missing** é disparado, onde eu uso uma expressão regular simples para retornar "auau!" caso o nome do método comece com a expressão latir.

Mas veja o que acontece quando tento usar o método **respond_to?**:

```
rex.respond_to? :latir #=> false
rex.latir #=> auau!
```

Ele retorna false, e isto faz todo o sentido já que o método realmente não existe. Então fica na minha responsabilidade alterar o método **respond_to?** para que ele funcione direito usando esta minha regra especial. Vou alterar minha classe para isto:

```
class Cachorro
  METODO_LATIR = /^latir/

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

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

rex = Cachorro.new
rex.respond_to?(:latir) #=> true
rex.latir #=> auau!
```

Agora sim! Este é um erro comum que tenho visto em alguns códigos, inclusive no próprio Rails, tente executar um **respond_to?** para verificar a existência de métodos como **find_by_name**, por exemplo.

Ruby é uma linguagem impressionante e flexível ao máximo, mas se não tomarmos cuidado podemos deixar pontas soltas como esta.

Claro que no Rails 2.1 corrigiram este problema, poderemos usar o **respond_to?** para verificar a existência de métodos como o **find_by_alguma_coisa**.

POSTGRESQL

No Rails, o adapter atual para **PostgreSQL** somente tem suporte para as versões 8.1 até a 8.3. Agora foi acrescentando suporte desde a versão 7.4 até a 8.3.

Capítulo 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 [lrfy]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
map.root      :dashboard
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
The tags created do not contain a date (http://feedvalidator.org/docs/error/InvalidTAG.html)
IDs are not guaranteed unique
A default self link was not provided, contrary to the documentation
NOTE:  This changes tags for existing atom entries, but at least they validate now.
```

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

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

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

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

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

ACTIVERECORD

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

* Remove duplicate code from associations. [Pratik]

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

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

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

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

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

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

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

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

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

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

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

* Track changes to unsaved attributes. [Jeremy Kemper]

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

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

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

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

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

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

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

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

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

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

Article.published.paginate(:page => 1)
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ACTIVERESOURCE

2.1.0 (May 31st, 2008)*

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

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

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

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

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

* find_or_create_resource_for handles module nesting. #10646 [xavier]

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

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

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

* Ruby 1.9 compatibility. [Jeremy Kemper]

ACTIVESUPPORT

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

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

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

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

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

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

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

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

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

* Added Ruby 1.8 implementation of Process.daemon

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

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

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

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

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

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

* Improve documentation. [Xavier Noria]

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

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

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

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

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

* TZInfo: Removing unneeded TimezoneProxy class [Geoff Buesing]

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

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

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

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

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

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

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

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

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

* Ensure correct TimeWithZone#to_date [Geoff Buesing]

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

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

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

* Adding TimeZone#parse [Geoff Buesing]

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

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

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

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

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

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

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

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

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

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

* Add OrderedHash#to_hash [josh]

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

* Adding TimeWithZone#between? [Geoff Buesing]

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

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

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

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

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

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

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

* Pruning unneeded TimeWithZone#change_time_zone_to_current [Geoff Buesing]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

RAILTIES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Enables or disables plugin reloading.

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

You can get around this setting per plugin.

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

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

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

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

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

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

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

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

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

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

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

# List required gems.
    rake gems

# Install all required gems:
    rake gems:install

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

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

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

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

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

* Added that gems can now be plugins if they include rails/init.rb #11444 [jbarnette]

* Added Plugin#about method to programmatically access the about.yml in a plugin #10979 [lazyatom]

```
plugin = Rails::Plugin.new(path\_to\_my\_plugin)
plugin.about["author"] # => "James Adam"
plugin.about["url"] # => "http://interblah.net"
```

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

* Added config.time_zone = 'UTC' in the default environment.rb [Geoff Buesing]

* Added rake tasks time:zones:all, time:zones:us and time:zones:local for finding time zone names for config.time_zone option [Geoff Buesing]

* Add config.time_zone for configuring the default Time.zone value. #10982 [Geoff Buesing]

* Added support for installing plugins hosted at git repositories #11294 [danger]

* Fixed that script/generate would not look for plugin generators in plugin_paths #11000 [glv]

* Fixed database rake tasks to work with charset/collation and show proper error messages on failure. Closes #11301 [matt]

* Added a -e/--export to script/plugin install, uses svn export. #10847 [jon@blankpad.net)]

* Reshuffle load order so that routes and observers are initialized after plugins and app initializers. Closes #10980 [rick]

* Git support for script/generate. #10690 [ssoroka]

* Update scaffold to use labels instead of bold tags. Closes #10757 [zach-inglis-lt3]

* Resurrect WordNet synonym lookups. #10710 [tom./, matt]

* Added config.cache_store to environment options to control the default cache store (default is FileStore if tmp/cache is present, otherwise MemoryStore is used) [DHH]

* Added that rails:update is run when you do rails:freeze:edge to ensure you also get the latest JS and config files #10565 [jeff]

* SQLite: db:drop:all doesn't fail silently if the database is already open. #10577 [Cheah Chu Yeow, mrichman]

* Introduce native mongrel handler and push mutex into dispatcher. [Jeremy Kemper]

* Ruby 1.9 compatibility. #1689, #10546 [Cheah Chu Yeow, frederico]

RUBY ON RAILS 2.1

O QUE HÁ DE NOVO?

CARLOS BRANDO REVISÃO: MARCOS TAPAJÓS - CAPA/C. CAPA: DANIEL LOPES