

AssociationProxy Tutorial

2007-09-08

김 석 준
(sjoonk@gmail.com)

inspect do |association|

```
class User < ActiveRecord::Base
  has_many :articles
end
```

```
class Article < ActiveRecord::Base
  belongs_to :user
end
```

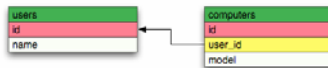
Has One and Belongs To

Picture has_one :thumbnail
Thumbnail belongs_to :picture



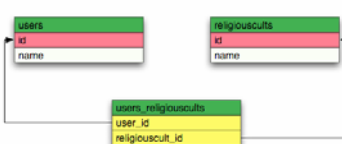
Has Many

User has_many :computers



Has and Belongs to Many

User has_and_belongs_to_many :religiouscults



association.use_case 1

■ Scoped Access (or Assigning Ownership)

```
> Article.find_all_by_user_id(user)
# SELECT * FROM articles WHERE (articles.user_id = 1)

> user.articles
# SELECT * FROM articles WHERE (articles.user_id = 1)
```

```
> Article.create(:user => user).merge(:title => "Bingo!")
# INSERT INTO articles ("title", "user_id", ...) VALUES("Bingo!", ...)

> user.articles.create(:title => "Ruby rocks!")
# INSERT INTO articles ("title", "user_id", ...) VALUES('Ruby rocks!', ...)
```

3

association.use_case 2

■ Special Queries (or Custom Finders)

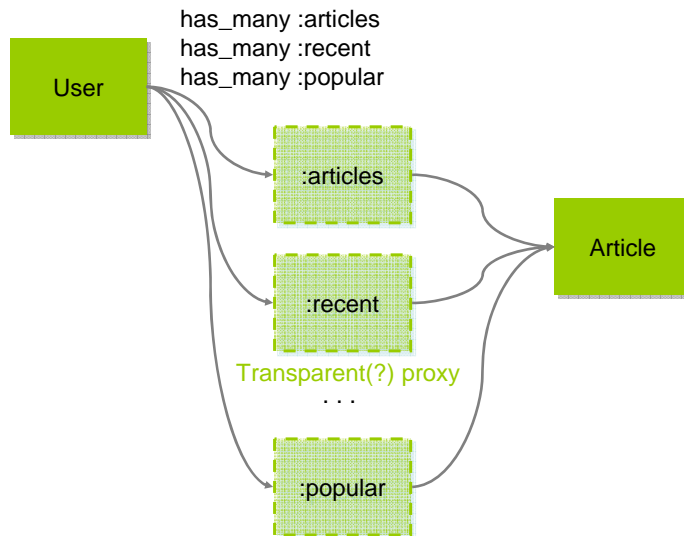
```
class Article < ActiveRecord::Base
  belongs_to :user
  def self.recent
    find(:all, :conditions => ['created_at > ?', 10.days.ago])
  end
end

class User < ActiveRecord::Base
  has_many :recent_articles, :class_name => 'Article',
    :conditions => ['created_at > ?', 10.days.ago]
end
```

```
> Article.recent
> user.articles.recent
> user.recent_articles
# SELECT * FROM articles WHERE (articles.user_id = 1 AND
created_at > ...)
```

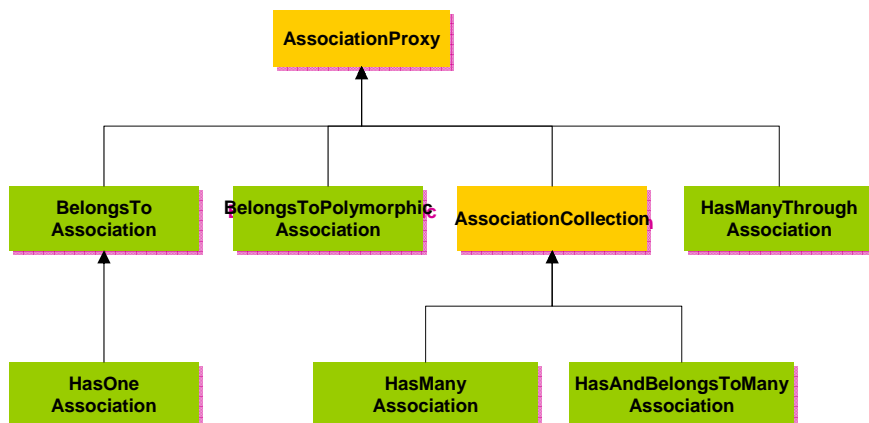
4

association.proxy



5

association.class == AssociationProxy



6

association.extend!

```
has_many :articles do
  def recent(reload=false)
    @recent = nil if reload
    @recent ||= find(:all, :conditions => ['created_at > ?', 10.days.ago])
  end
end
```

```
> user.articles.recent
> user.articles.recent(true)
# SELECT * FROM articles WHERE (articles.user_id = 1 AND
created_at > ...)
```

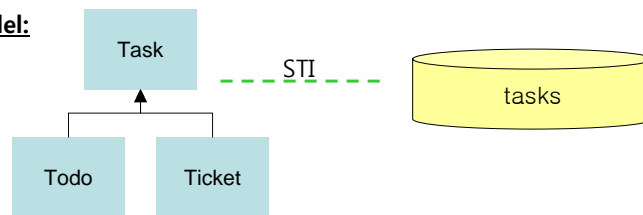
cf. proxy owner, proxy reflection, proxy target

```
has_many :neighborhoods do
  def loaded?
    return true if (proxy_owner.population.nil? or proxy_owner.population < 50000)
  super
  end
end
```

7

association.example :task_management

Model:



```
class User < ActiveRecord::Base
  has_many :todos, :foreign_key => 'created_by', :order => 'created_at DESC'
  has_many :inbounds, :class_name => 'Ticket', :foreign_key => 'asked_to'
  has_many :outbounds, :class_name => 'Ticket', :foreign_key => 'created_by'
```

Routes:

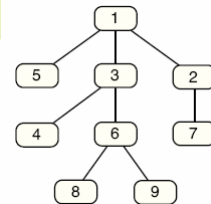
```
map.resources :todos, :new => { :take => :post }
map.resources :inbounds, :member => { :take => :get }
map.resources :outbounds
```

8

association.example :acts_as_tree

```
class Category < ActiveRecord::Base
  acts_as_tree :order => "name"
end
```

```
# root = Category.create("name" => "root")
# child1 = root.children.create("name" => "child1")
# subchild1 = child1.children.create("name" => "subchild1")
#
# root.parent # => nil
# child1.parent # => root
# root.children # => [child1]
# root.children.first.children.first # => subchild1
```



```
def acts_as_tree(options = {})
```

```
// ...
```

```
  belongs_to :parent, :class_name => name, :foreign_key => configuration[:foreign_key],
              :counter_cache => configuration[:counter_cache]
```

```
  has_many :children, :class_name => name, :foreign_key => configuration[:foreign_key],
              :order => configuration[:order], :dependent => :destroy
```

9

association.plugins.related

■ scope_out plugin

```
scope_out :published, :conditions => { :published => 1 }
scope_out :recent do
  { :conditions => ['created_at > ?', 10.days.ago] }
end
```

```
> user.articles.with_recent { user.articles }
> user.articles.find_recent :all
> user.articles.calculate_recent :count, :all
```

■ scoped_proxy plugin

■ HasFinder gem

```
has_finder :published, :conditions => { :published => 1 }
has_finder :recent, :conditions => ['created_at > ?', 10.days.ago]
```

```
> user.articles.published.recent
```

10

end # thank you!

■ **References**

- <http://api.rubyonrails.com/classes/ActiveRecord/Associations/ClassMethods.html>