NESTED ATTRIBUTES

Hiep Le Tuan veu, Sun-Asterisk
le.tuan.hiep@sun-asterisk.com

Ngày 17 tháng 4 năm 2020

Mục lục

I. Tổng quát

Nested attribute cho phép lưu các thuộc tính trên các bản ghi liên quan thông qua bản ghi cha mẹ. Theo mặc định, việc cập nhật nested thuộc tính không thực hiện. Bạn có thể kích hoạt nó bằng cách sử dụng class method **accepts_nested_attributes_for**. Khi bạn bật nested attribute, một thuộc tính dùng cho việc gán giá trị sẽ được khai báo trên model. Thuộc tính dùng cho việc gán giá trị được đặt tên theo liên kết.
Ví dụ, hai phương thức mới được thêm vào model của bạn: **author_attributes=(attributes)** và **pages_attributes=(attributes)**

```
class Book < ActiveRecord::Base
  has_one :author
  has_many :pages

  accepts_nested_attributes_for :author, :pages
end
```

Lưu ý rằng tùy chọn **:autosave** được tự động sử dụng trên mọi liên kết mà **accepts_nested_attributes_for** được sử dụng.

II. Quan hệ 1-1

Ví dụ trường hợp một Member có một Avatar:

```
class Member < ActiveRecord::Base
  has_one :avatar
  accepts_nested_attributes_for :avatar
end
```

Sử dụng nested thuộc tính trong mối quan hệ 1-1 cho phép bạn có thể tạo member và avatar trong cùng 1 lần:

```
params = { member: { name: 'Jack', avatar_attributes: { icon: 'smiling' } } }
member = Member.create(params[:member])
member.avatar.id # => 2
member.avatar.icon # => 'smiling'
```

Nó cho phép chúng ta update avatar thông qua member

```
params = { member: { avatar_attributes: { id: '2', icon: 'sad' } } }
member.update params[:member]
member.avatar.icon # => 'sad'
```

Nếu bạn muốn update avatar hiện tại mà không cần thông qua id, bạn có thể sử dụng option**:update_only**

```
class Member < ActiveRecord::Base
  has_one :avatar
  accepts_nested_attributes_for :avatar, update_only: true
end

params = { member: { avatar_attributes: { icon: 'sad' } } }
member.update params[:member]
member.avatar.id # => 2
member.avatar.icon # => 'sad'
```

Theo mặc định, bạn sẽ chỉ có thể thiết lập và cập nhật các thuộc tính trên model được liên kết. Nếu bạn muốn xóa model được liên kết thông qua hash thuộc tính, trước tiên bạn phải cho phép nó bằng cách dùng option **:allow_destroy**

```
class Member < ActiveRecord::Base
  has_one :avatar
  accepts_nested_attributes_for :avatar, allow_destroy: true
end
```

Khi ta thêm khóa **_destroy** vào hash thuộc tính, với giá trị được thiết lập là true, bạn sẽ xóa model liên kết với nó.

```
member.avatar_attributes = { id: '2', _destroy: '1' }
member.avatar.marked_for_destruction? # => true
member.save
member.reload.avatar # => nil
```

Lưu ý rằng model sẽ không bị xóa cho tới khi model cha mẹ được lưu. Cũng chú ý rằng model sẽ không bị xóa nếu ta không chỉ định id trong updated hash.

III. Quan hệ 1-nhiều

Ví dụ một member có nhiều post:

```
class Member < ActiveRecord::Base
  has_many :posts
  accepts_nested_attributes_for :posts
end
```

Bạn có thể thiết lập và cập nhật thuộc tính qua những post đã liên kết thông qua 1 hash thuộc tính cho member: bao gồm cả key **:posts_attributes** với một hash thuộc tính của post làm giá trị.
Đối với mỗi hash không có khóa id, một bản ghi mới sẽ được tạo ngay lập tức, trừ khi hash cũng chứa khóa **_destroy** thiết lập là true.

```
member = Member.create(params[:member])
```

```
member.posts.length # => 2
member.posts.first.title # => 'Kari, the awesome Ruby documentation browser!'
member.posts.second.title # => 'The egalitarian assumption of the modern citizen'
```

Bạn cũng có thể thiết lập **:reject_if** proc để âm thầm bỏ qua mọi hash bản ghi mới nếu chúng không vượt qua những điều kiện mà bạn đặt ra. Ví dụ trước có thể được viết lại thành:

```
class Member < ActiveRecord::Base
  has_many :posts
  accepts_nested_attributes_for :posts, reject_if: proc { |attributes| attributes['title'].blank? }
end

params = { member: {
  name: 'joe', posts_attributes: [
    { title: 'Kari, the awesome Ruby documentation browser!' },
    { title: 'The egalitarian assumption of the modern citizen' },
    { title: '' } # this will be ignored because of the :reject_if proc
  ]
}}

member = Member.create(params[:member])
member.posts.length # => 2
member.posts.first.title # => 'Kari, the awesome Ruby documentation browser!'
member.posts.second.title # => 'The egalitarian assumption of the modern citizen'
```

Hoặc **:reject_if** cũng chấp nhận một symbol để sử dụng method

```
class Member < ActiveRecord::Base
  has_many :posts
  accepts_nested_attributes_for :posts, reject_if: :new_record?
end

class Member < ActiveRecord::Base
  has_many :posts
  accepts_nested_attributes_for :posts, reject_if: :reject_posts

  def reject_posts(attributes)
    attributes['title'].blank?
  end
end
```

Nếu hash chứa một key id tương ứng với một bản ghi đã được liên kết, bản ghi phù hợp sẽ được sửa đổi.

```
member.attributes = {
  name: 'Joe',
  posts_attributes: [
    { id: 1, title: '[UPDATED] An, as of yet, undisclosed awesome Ruby documentation browser!' },
    { id: 2, title: '[UPDATED] other post' }
  ]
}

member.posts.first.title # => '[UPDATED] An, as of yet, undisclosed awesome Ruby documentation browser!'
member.posts.second.title # => '[UPDATED] other post'
```

Tuy nhiên, những điều trên cũng áp dụng nếu model mẹ cũng đang được cập nhật. Ví dụ: Nếu bạn muốn tạo một thành viên có tên joe và muốn cập nhật các bài đăng cùng một lúc, điều đó sẽ gây ra lỗi **ActiveRecord :: RecordNotFound**.
Theo mặc định, các bản ghi liên quan được bảo vệ khỏi bị xóa. Nếu bạn muốn xóa bất kỳ bản ghi liên quan nào thông qua hash thuộc tính, trước tiên bạn phải bật nó bằng option **:allow_destroy**. Điều này sẽ cho phép bạn cũng sử dụng khóa **_destroy** để hủy các bản ghi hiện có:

```ruby
class Member < ActiveRecord::Base
  has_many :posts
  accepts_nested_attributes_for :posts, allow_destroy: true
end

params = { member: {
  posts_attributes: [{ id: '2', _destroy: '1' }]
}}

member.attributes = params[:member]
member.posts.detect { |p| p.id == 2 }.marked_for_destruction? # => true
member.posts.length # => 2
member.save
member.reload.posts.length # => 1
```

Nested attributes cho một tập hợp các object liên kết cũng có thể được chuyển sang dạng hash các hash thay vì một mảng các hash.

```ruby
Member.create(
  name: 'joe',
  posts_attributes: {
    first: { title: 'Foo' },
    second: { title: 'Bar' }
  }
)
```

Tác dụng tương đương nhau:

```ruby
Member.create(
  name: 'joe',
  posts_attributes: [
    { title: 'Foo' },
    { title: 'Bar' }
  ]
)
```

Các keys của hash mà là giá trị cho **:post_attributes** bị bỏ qua trong trường hợp này. Tuy nhiên, không được phép sử dụng 'id' hoặc **:id** cho một trong các khóa như vậy, nếu không, hash sẽ được gói trong một mảng và được hiểu là thuộc tính hash cho một bài đăng.
Việc truyền các thuộc tính cho một collection model được liên kết dưới dạng hash có thể được sử dụng với các giá hash được tạo từ các tham số HTTP / HTML, trong đó có thể không có cách nào tự nhiên để gửi một mảng hash.

1. Lưu model

Tất cả các thay đổi đối với model, bao gồm cả việc xóa các model được đánh dấu để hủy, được tự động lưu và hủy bỏ tách biệt ta khi model mẹ được lưu. Điều này xảy ra bên trong giao dịch được khởi tạo bởi phương thức save của model mẹ.

2. Xác nhận sự tồn tại của model mẹ

Nếu bạn muốn xác nhận một record con có được liên kết với một record cha không, bạn có thể sử dụng method **validates_presence_of** và **:inverse_of** key như ví dụ minh họa sau:

```ruby
class Member < ActiveRecord::Base
  has_many :posts, inverse_of: :member
  accepts_nested_attributes_for :posts
end

class Post < ActiveRecord::Base
```

```
  belongs_to :member, inverse_of: :posts
  validates_presence_of :member
end
```

Chú ý: nếu bạn không chỉ định option **:inverse_of**, sau đó Active Record sẽ cố gắng tự động đoán liên kết ngược dựa trên heuristics.
Đối với liên kết nested 1-1, nếu bạn build một object con mới (trong bộ nhớ) trước khi được gán, module này sẽ không ghi đè.

```
class Member < ActiveRecord::Base
  has_one :avatar
  accepts_nested_attributes_for :avatar

  def avatar
    super || build_avatar(width: 200)
  end
end

member = Member.new
member.avatar_attributes = {icon: 'sad'}
member.avatar.width # => 200
```

IV. Instance Public Methods

accepts_nested_attributes_for(attr_names) Định nghĩa một thuộc tính writer cho liên kết cụ thể.
Options hỗ trợ:
:allow_destroy
Nếu thiết lập là true, xóa bất kỳ members nào trong hash thuộc tính có key **_destroy** và giá trị là true.
:reject_if
Cho phép bạn chỉ định Proc hoặc method trỏ đến một method kiểm tra xem bản ghi có nên được xây dựng cho một hash thuộc tính nhất định hay không. Hash được truyền cho Proc hoặc phương thức và nó sẽ trả về giá trị true hoặc false. Khi **:reject_if** không được chỉ định, một bản ghi sẽ được tạo cho tất cả các giá trị trong hash thuộc tính những cái mà không có giá trị **_destroy**: true. Truyền **all_blank** thay vì Proc sẽ tạo ra một Proc dùng để từ chối một bản ghi trong đó tất cả các thuộc tính đều trống, không bao gồm bất kỳ gía trị nào của **_destroy**.
:limit
Cho phép bạn chỉ định số lượng bản ghi liên quan tối đa có thể được xử lý với nested attribute. Limit cũng có thể được chỉ định là Proc hoặc method chỉ đến một method (method sẽ trả về một số). Nếu kích thước của nested attributes vượt quá giới hạn đã chỉ định, ngoại lệ **NestedAttributes::TooManyRecords** được sinh ra. Nếu bỏ qua, bất cứ số lượng bản ghi nào cũng được chấp nhận. Lưu ý rằng tùy chọn **:limit** chỉ áp dụng cho quan hệ một-nhiều.
:update_only
Đối với liên kết một-một, tùy chọn này cho phép bạn chỉ định cách các nested attribute sẽ được sử dụng khi bản ghi liên quan đã tồn tại. Nói chung, một bản ghi hiện tại có thể được cập nhật với bộ giá trị thuộc tính mới hoặc được thay thế bằng một bản ghi hoàn toàn mới có chứa các giá trị đó. Theo mặc định, tùy chọn **:update_only** là false và nested attribute được sử dụng để cập nhật bản ghi hiện tại chỉ khi chúng bao gồm giá trị id của bản ghi. Nếu không, một bản ghi mới sẽ được khởi tạo và sử dụng để thay thế bản ghi hiện có. Tuy nhiên, nếu tùy chọn **:update_only** là true, nested attribute được sử dụng để luôn cập nhật các thuộc tính của bản ghi, bất kể có id có hay không. Tùy chọn được bỏ qua cho đối collection association.

```
# creates avatar_attributes=
accepts_nested_attributes_for :avatar, reject_if: proc { |attributes| attributes['name'].blank? }
# creates avatar_attributes=
accepts_nested_attributes_for :avatar, reject_if: :all_blank
# creates avatar_attributes= and posts_attributes=
accepts_nested_attributes_for :avatar, :posts, allow_destroy: true
```

Tài liệu

[1] *https://api.rubyonrails.org/classes/ActiveRecord/NestedAttributes/ClassMethods.html#methoị-accepts_nested_attributes_for*,