
deep_cloneable



This gem gives every ActiveRecord::Base object the possibility to do a deep clone that includes user specified associations. It is a rails 3+ upgrade of the deep_cloning plugin.

Requirements

- Ruby 2.3.0, 2.4.4, 2.5.5, 2.6.3, 2.7.5 (tested)
- TruffleRuby 21.3.0
- ActiveRecord 3.2, 4.0, 4.1, 4.2, 5.0, 5.1, 5.2, 6.0, 7.0 (tested)
- Rails 2.x/3.0 users, please check out the 'rails2.x-3.0' branch

Installation

- Add deep_cloneable to your Gemfile:

```
1 gem 'deep_cloneable', '~> 3.2.0'
```

Upgrade details

Upgrading from v2

There are two breaking changes that you might need to pay attention to:

- When using an optional block (see below), the block used to be evaluated before `deep_cloneable` had performed its changes (inclusions, exclusions, includes). In v3, the block is evaluated after all processing has been done, just before the copy is about to be returned.
- When a defined association is not found, `deep_cloneable` raises an exception. The exception class has changed namespace: the class definition used to be `ActiveRecord::Base::DeepCloneable::AssociationNotFoundException` and this has changed to `DeepCloneable::AssociationNotFoundException`.

Upgrading from v1

The `dup` method with arguments has been replaced in `deep_cloneable 2` by the method `deep_clone`. Please update your sources accordingly.

Usage

The `deep_clone` method supports a couple options that can be specified by passing an options hash. Without options, the behaviour is the same as ActiveRecord's `dup` method.

Association inclusion

Associations to be included in the dup can be specified with the `include` option:

```
1 # Single include
2 pirate.deep_clone include: :mateys
3
4 # Multiple includes
5 pirate.deep_clone include: [ :mateys, :treasures ]
6
7 # Deep includes
8 pirate.deep_clone include: { treasures: :gold_pieces }
9 pirate.deep_clone include: [ :mateys, { treasures: :gold_pieces } ]
10
11 # Disable validation for a performance speedup when saving the dup
12 pirate.deep_clone include: { treasures: :gold_pieces }, validate: false
13
14 # Conditional includes
15 pirate.deep_clone include: [
16   {
17     treasures: { gold_pieces: { if: lambda{|piece| piece.is_a?(Parrot)}
18                               } } },
19   mateys: { unless: lambda{|matey| matey.is_a?(GoldPiece)} }
20 ]
21
22 ship.deep_clone include: [
23   pirates: [ :treasures, :mateys, if: lambda {|pirate| pirate.name == '
24     Jack Sparrow' } ] ]
```

The Dictionary (Object Reusage) The dictionary ensures that models are not duped multiple times when it is associated to nested models. It does this by storing a mapping of the original object to its duped object. It can be used as follows:

```
1 # Enables the dictionary (empty on initialization)
2 pirate.deep_clone include: [ :mateys, { treasures: [ :matey, :
3   gold_pieces ] } ], use_dictionary: true
4
5 # Deep clones with a prefilled dictionary
6 dictionary = { mateys: {} }
```

```
6 pirate.mateys.each{|m| dict[:mateys][m] = m.deep_clone }
7 pirate.deep_clone include: [ :mateys, { treasures: [ :matey, :
  gold_pieces ] } ], dictionary: dictionary
```

Attribute Exceptions & Inclusions

The `deep_clone` method supports both `except` and `only` for specifying which attributes should be duped:

Exceptions

```
1 # Single exception
2 pirate.deep_clone except: :name
3
4 # Multiple exceptions
5 pirate.deep_clone except: [ :name, :nick_name ]
6
7 # Nested exceptions
8 pirate.deep_clone include: :parrot, except: [ :name, { parrot: [ :name
  ] } ]
```

Inclusions

```
1 # Single attribute inclusion
2 pirate.deep_clone only: :name
3
4 # Multiple attribute inclusions
5 pirate.deep_clone only: [ :name, :nick_name ]
6
7 # Nested attribute inclusions
8 pirate.deep_clone include: :parrot, only: [ :name, { parrot: [ :name ]
  } ]
```

Pre- and postprocessor

You can specify a pre- and/or a postprocessor to modify a duped object after duplication:

```
1 pirate.deep_clone(include: :parrot, preprocessor: ->(original, kopy) {
  kopy.cloned_from_id = original.id if kopy.respond_to?(:
  cloned_from_id) })
2 pirate.deep_clone(include: :parrot, postprocessor: ->(original, kopy) {
  kopy.cloned_from_id = original.id if kopy.respond_to?(:
  cloned_from_id) })
```

Note: Specifying a postprocessor is essentially the same as specifying an optional block (see below).

Note: Using `deep_clone` with a processors will pass all associated objects that are being cloned to the processor, so be sure to check whether the object actually responds to your method of choice.

Optional Block

Pass a block to `deep_clone` to modify a duped object after duplication:

```
1 pirate.deep_clone include: :parrot do |original, kopy|
2   kopy.cloned_from_id = original.id if kopy.respond_to?(:cloned_from_id)
3 end
```

Note: Using `deep_clone` with a block will also pass the associated objects that are being cloned to the block, so be sure to check whether the object actually responds to your method of choice.

Cloning models with files

Carrierwave If you are cloning models that have associated files through Carrierwave these will not get transferred automatically. To overcome the issue you need to explicitly set the file attribute.

Easiest solution is to add the code in a clone block as described above.

```
1 pirate.deep_clone include: :parrot do |original, kopy|
2   kopy.thumbnail = original.thumbnail
3 end
```

ActiveStorage For ActiveStorage, you have two options: you can either make a full copy, or share data blobs between two records.

Full copy example

```
1 # Rails 5.2, has_one_attached example 1
2 pirate.deep_clone include: [:parrot, :avatar_attachment, :avatar_blob]
3
4 # Rails 5.2, has_one_attached example 2
5 pirate.deep_clone include: :parrot do |original, kopy|
6   if kopy.is_a?(Pirate) && original.avatar.attached?
7     attachment = original.avatar
8     kopy.avatar.attach \
9       :io           => StringIO.new(attachment.download),
10      :filename      => attachment.filename,
11      :content_type  => attachment.content_type
12   end
13 end
14
15 # Rails 5.2, has_many_attached example 1 (attach one by one)
16 pirate.deep_clone include: :parrot do |original, kopy|
17   if kopy.is_a?(Pirate) && original.crew_members_images.attached?
18     original.crew_members_images.each do |attachment|
19       kopy.crew_members_images.attach \
```

```

20         :io          => StringIO.new(attachment.download),
21         :filename     => attachment.filename,
22         :content_type => attachment.content_type
23     end
24 end
25 end
26
27 # Rails 5.2, has_many_attached example 2 (attach bulk)
28 pirate.deep_clone include: :parrot do |original, kopy|
29     if kopy.is_a?(Pirate) && original.crew_members_images.attached?
30         all_attachments_arr = original.crew_members_images.map do |
31             attachment|
32             {
33                 :io          => StringIO.new(attachment.download),
34                 :filename     => attachment.filename,
35                 :content_type => attachment.content_type
36             }
37         end
38         kopy.crew_members_images.attach(all_attachments_arr) # attach all
39                     at once
40     end
41 end
42
43 # Rails 6
44 pirate.deep_clone include: :parrot do |original, kopy|
45     if kopy.is_a?(Pirate) && original.avatar.attached?
46         original.avatar.open do |tempfile|
47             kopy.avatar.attach({
48                 io: File.open(tempfile.path),
49                 filename: original.avatar.blob.filename,
50                 content_type: original.avatar.blob.content_type
51             })
52         end
53     end
54 end
55 end

```

Shallow copy example

```

1 pirate.deep_clone include: :parrot do |original, kopy|
2     kopy.avatar.attach(original.avatar.blob) if kopy.is_a?(Pirate) &&
3         original.avatar.attached?
4 end

```

Skipping missing associations

By default, `deep_cloneable` will throw a `DeepCloneable::AssociationNotFoundException` error when an association cannot be found. You can also skip missing associations by specifying `skip_missing_associations` if needed, for example when you have associations on some

(but not all) subclasses of an STI model:

```
1 pirate.deep_clone include: [:parrot, :rum], skip_missing_associations:  
  true
```

Note on Patches/Pull Requests

- Fork the project.
- Make your feature addition or bug fix.
- Add tests for it. This is important so I don't break it in a future version unintentionally.
- Commit, do not mess with rakefile, version, or history. (if you want to have your own version, that is fine but bump version in a commit by itself I can ignore when I pull)
- Send me a pull request. Bonus points for topic branches.

Copyright

Copyright © 2021 Reinier de Lange. See LICENSE for details.