Cheatography

Rails 4 Cheat Sheet

by entaro via cheatography.com/4988/cs/931/

Activer record queries

```
Post.where(author: 'admin')
Post.where(author: 'admin').last
Post.find_by(title: 'Rails 4', author: 'admin')
Post.find_or_initialize_by(title: 'Rails 4')
Post.find_or_create_by(title: 'Rails 4')
@post.update(post_params) - preferred
@post.update_columns(post_params) - executes directly in database(skip validation)
```

Scopes

```
scope :sold, ->{ where(state: 'sold') }
default_scope ->{ where(state: 'available') }
scope :recent, ->{ where(published_at: 2.weeks.ago) }
scope :recent_red, ->{ recent.where(color: 'red') }
Post.where.not(author: author)
User.order(:name, created_at: :desc)
Post.includes(:comments).where(comments: { name: 'foo' })
Post.includes(:comments).where('comments.name' => 'foo')
Post.includes(:comments).order('comments.name')
```

Flash types

```
class ApplicationController < ActionController::Base
  add_flash_types :grunt, :snarl
end
flash[:grunt] = 'braaains...'
redirect_to @user, grunt: 'braaains...'
<div id="grunt"><%= grunt %></div>
```

Concerns

```
concern :sociable do |options|
  resources :comments, options
  resources :categories, options
end
resources :messages, concerns: :sociable
resources :items do
  concerns :sociable, only: :create
end
```

Match routes

```
match '/items/:id/purchase', to: 'items#purchase', via: :post
match '/items/:id/purchase', to: 'items#purchase', via: :all
```

Collection form helpers

```
class Owner < ActiveRecord::Base
  has_many :items
end
class Item < ActiveRecord::Base
  belongs_to :owner
end
collection_select(:item, :owner_id, Owner.all, :id, :name)
collection_radio_buttons(:item, :owner_id, Owner.all, :id, :name)
collection_check_boxes(:item, :owner_id, Owner.all, :id, :name)
<%= f.date_select :return_date %>
```

Postgres support

To get started, first setup your database to use the hstore extension:

```
class AddHstoreExtension < ActiveRecord::Migration
 def up
  execute 'CREATE EXTENSION hstore'
 end
 def down
  execute 'DROP EXTENSION hstore'
 end
end
```

Indexes

If you are doing any queries on an hstore property, be sure to add the appropriate index. When adding an index, you will have to decide to use either GIN or GiST index types. The distinguishing factor between the two index types is that GIN index lookups are three times faster than GiST indexes, however they also take three times longer to build. Checkout the documentation, which goes into detail about the differences.

```
class AddIndexToComicsProperties < ActiveRecord::Migration
 def up
  execute 'CREATE INDEX comics_properties ON comics USING gin(properties)'
 end
 def down
  execute 'DROP INDEX comics_properties'
 end
end
```

Using the store_accessor macro style method in Active Record models, we can add read/write accessors to key/value hstore properties:

```
class Comic < ActiveRecord::Base
  store_accessor :properties, :story_arc
end
comic = Comic.create
comic.properties # => nil
comic.story_arc = 'Throne of Atlantis'
comic.save
```

To query against hstore data in Active Record, use SQL string conditions with the where query method:

```
Comic.where("properties -> 'story_arc' = 'Throne of Atlantis'")
```

Array support

```
class AddTagsToArticles < ActiveRecord::Migration
def change
  change_table :articles do |t|
   t.string :tags, array: true
 end
end
article.tags = ['rails']
article.save
```

Another example

```
def due_date_params
 params.require(:due_date).permit(
  :name { :tags => [] },
  :day_of_month, :day_of_week, :frequency
 )
end
class DueDate < ActiveRecord::Base
  store_accessor :recur, :frequency
  store_accessor :recur, :day_of_week
  store_accessor :recur, :day_of_month
end
```