Cheatography

Rails 5 ActiveRecord Cheat Sheet

by dwapi via cheatography.com/43665/cs/12981/

Migrations

```
class CreateProducts < ActiveRecord::Migration[5.0]
  def change
    create_table :products do |t|
      t.string :name
      t.text :description

      t.timestamps
    end
  end
end
class AddPartNumberToProducts <
ActiveRecord::Migration[5.0]
  def change
    add_column :products, :part_number, :string
    add_index :products, :part_number
    add_reference :products, :user, foreign_key: true
    add_column :products, :price, :decimal, precision:
5, scale: 2
  end
end
class CreateJoinTableCustomerProduct <
ActiveRecord::Migration[5.0]
  def change
    create_join_table :customers, :products do |t|
      # t.index [:customer_id, :product_id]
      # t.index [:product_id, :customer_id]
    end
  end
end
class ChangeProductsPrice <
ActiveRecord::Migration[5.0]
  def up
    change_table :products do |t|
      t.change :price, :string
      rename_column :users, :email, :email_address
    end
  end

  def down
    change_table :products do |t|
      t.change :price, :integer
      rename_column :users, :email_address, :email
```

Migrations (cont)

```
    end
  end
end
```

rails db: command

Rollback the last migration:	rails db:rollback
Revert the last 3 migrations:	rails db:rollback STEP=3
Rollback and then migrating back up:	rails db:migrate:redo STEP=3
Run specific migration	rails db:migrate:up VERSION=20080906120000

Models: Callbacks and Validations

```
class User < ApplicationRecord
  validates :name, presence: true
  validates :name, length: { maximum: 20 }
  validates :password, length: { in: 6..20, message:
"%{value} is not valid" }
  validates :email, confirmation: true # Adds tmp attr
email_confirmation
  has_many :books, :dependent => :destroy
  validates_associated :books #if has_many :books
association

  before_save :ensure_login_has_a_value, if:
:some_test?
  after_validation :set_location, on: [ :create,
:update ]

  scope :published, -> { where(published: true) }



  private
    def ensure_login_has_a_value
      if login.nil?
        self.login = email unless email.blank?
      end
    end
end
```

Routing

```
Rails.application.routes.draw do
  root to: 'main#index'
  get "/login" => "sessions#new"
  get 'photos/:id', to: 'photos#show', defaults: {
format: 'jpg' }

  post "/login" => "sessions#create"

  namespace :api do
    resources :posts, only: [:index, :create, :destroy,
:update]
  end
  defaults format: :json do
    resources :photos
  end
  #Enable websocket requests
  mount ActionCable.server => '/cable'
end
```

Concepts

Module: "Enumerable" is actually a "module", which means it is just a bunch of methods packaged together that can (and do) get "mixed in", or included, with other classes (like Array and Hash)