
RAILS

Model

```
class Model < ActiveRecord::Base
    belongs_to :associated_model
    has_many :associated_models

    after_create do |model|
    end

    before_create do |model|
    end
end
```

Association

— Assuming model have a `user_id` field

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

model = Model.first
model.user
 # => User.find(model.user_id)
 # return a user
```

— Assuming user have a `model_id` field

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

user = User.first
user.models
 # => Model.where(user_id: user.id)
 # return array of models
```

Find

```
ModelName.where(field: value)
 # => Select * from model_names where field = value
 # return empty array if no match
ModelName.order(field: :desc)
 # => Select * from model_names order by field desc
Model.where(field: value).order(field: :desc)
 # => Select * from model_names where field = value order by field desc

Model.find_by(field: value)
 # => Select * from model_names where field = value limit 1
 # return nil if not found
Model.find(123)
 # => Select * from model_names where id = 123
 # raise error if not found
```

Create

```
model = Model.create(field: value)
 # => Insert into models (field) values (value)
 # return nil if error
model = Model.create!(field: value)
 # => Insert into models (field) values (value)
 # raise error if error

model = Model.new(field: value)
model.field = new_value
 # simply create a new model WITHOUT saving it
model.save
 # => Insert into models (field) values (value)
 # return true if success
 # return false if error
model.save!
 # => Insert into models (field) values (value)
 # raise error if error
```

Update

```
model = Model.find(1)
model.field = new_value
model.save
 # => Uppdate into models set field = value where id = 1
 # return true if success
 # return false if error
model.save!
 # => Uppdate into models set field = value where id = 1
 # raise error if error

modele.update(field: value)
 # => Uppdate into models set field = value where id = 1
 # raise error if error
```

Destroy

```
model = Model.find(1)
model.destroy
 # => Delete from models where id = 1
 # return true if success
 # return false if error
model.destroy!
 # => Delete from models where id = 1
 # raise error if error
```

Lifecycle

— after_create

```
class Model < ActiveRecord::Base
after_create do |model|
  # call AFTER a model is created (from anywhere, a controller, the console, ...)
  # so the model DOES exist in the database
  # model is the newly create model
end
end
```

— before_create

```
class Model < ActiveRecord::Base
before_create do |model|
  # call BEFORE a model is created (from anywhere, a controller, the console, ...)
  # so the model DOES NOT exist in the database yet
  # model is the new model
end
end
```

Controller

```
class UsersController < ApplicationController
  before_action :set_user, only: [:show, :update, :destroy]

  # GET /users
  def index
    @users = User.all

    render json: @users
  end

  # GET /users/1
  def show
    render json: @user
  end

  # POST /users
  def create
    @user = User.new(user_params)

    if @user.save
      render json: @user, status: :created, location: @user
    else
      render json: @user.errors, status: :unprocessable_entity
    end
  end

  # PATCH/PUT /users/1
  def update
    if @user.update(user_params)
      render json: @user
    else
      render json: @user.errors, status: :unprocessable_entity
    end
  end

  # DELETE /users/1
  def destroy
    @user.destroy
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_user
      @user = User.find(params[:id])
    end

    # Only allow a trusted parameter "white list" through.
    def user_params
      params.require(:user).permit(:name, :password)
    end
end
```

Lifecycle

— inheritance

— `before_action`

— `rescue_from`

Routes

```
# config/routes.rb
Rails.application.routes.draw do
  resources :sessions, only: [:create, :destroy]
  resources :memberships
  resources :chats do
    resources :messages
  end
  resources :users
  # For details on the DSL available within this file, see http://guides.rubyonrails.org/routing.html
end
```

Scaffold

```
bin/rails generate scaffold ModelName field_name:type
field_name:type
```

- routes
  - create / update / show / list / destroy
- controller
  - create / update / show / list / destroy
- model
- migration

Migration

```
bin/rails generate migrate Model field_name:type field_name:type
bin/rails generate migrate User name:string age:integer
 # => CREATE TABLE "users" ("id" integer PRIMARY KEY AUTOINCREMENT NOT NULL, "name" varchar, "age" integer, "created_at" datetime NOT NULL, "updated_at" datetime NOT NULL)

bin/rails generate migrate AddFieldNameToTableName field_name:type
bin/rails generate migrate AddPasswordToUser password:string
 # => ALTER TABLE "users" ADD "password" varchar
```

— generate a migration file in `db/migration_name.rb`.

— does NOT update the table (yet!)

Migrate

- `bin/rake db:migrate`
  - run ALL pending migration (perform the operation of the database)
  - update `schema.rb`
- `bin/rake db:migrate:reset`
  - drop and recreate table before running migrate

Seeds

Just a regular ruby file

```
['Hugues', 'Talia', 'Naelie', 'Kian'].each do |name|
  puts "creating user #{name}"
  User.create(name: name, password: SecureRandom.hex(10))
end

bin/rake db:seed
```

Error

```
raise StandardError.new('message')
raise StandardError, 'message'
```

Logs

```
tail -f log/development.log
```

Console

```
bin/rails console
```

Server

```
bin/rails server
```

SQLite

```
bin/rails dbconsole
```