Ruby on Rails

Creando/Modificando tablas y columnas

Overview

- Como crear y borrar tablas
- Como crear y modificar columnas

Creando tablas

- Por una convención, las tablas en Rails se nombran **siempre** en **plural** (muchas filas o registros).
  - Nuestra tabla de **car** se llama **cars**
- Una columna **id** es automáticamente creada para ser utilizada como **primary key**.
- Existe un método que es agregado automáticamente a nuestro migration, se llama **timestamps**, este método crea las columnas: **created_at**, y **updated_at**. Estas columnas son actualizadas cada vez que actualizamos un registro ActiveRecord.
- **create_table** y **drop_table** son los comandos que crean y borran la tabla respectivamente.

Creando la tabla cars

20161112015033_create_cars.rb

```ruby
class CreateCars < ActiveRecord::Migration
  def change
    create_table :cars do |t|
      t.string :make
      t.string :color
      t.integer :year

      t.timestamps
    end
  end
end

```

Después de ejecutar rake db:migrate.

```
sqlite> .schema cars
CREATE TABLE "cars" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT N
ULL, "make" varchar(255), "color" varchar(255), "year" integer, "
created_at" datetime, "updated_at" datetime);
```

rake db:rollback

- Deshace el último migration (posiblemente aplica el método **down**)

```
➜  fancy_cars git:(master) ✗ rake db:rollback
== 20161112015033 CreateCars: reverting ======================================
-- drop_table(:cars)
   -> 0.0004s
== 20161112015033 CreateCars: reverted (0.0016s) =============================
```

Que pasó de nuestros datos!?

Agregando/Borrando Columnas

- `add_column :table_name, :column_name, :column_type`
- `remove_column :table_name, :column:name`

Agreguemos una columna **price** a nuestra tabla **cars**.

Agregando price a cars

```
➜  fancy_cars git:(master) ✗ rails g migration add_price_to_cars 'price:de
cimal{10,2}'
      invoke  active_record
      create    db/migrate/20161112061756_add_price_to_cars.rb
```

20161112061756_add_price_to_cars.rb

```ruby
class AddPriceToCars < ActiveRecord::Migration
  def change
    add_column :cars, :price, :decimal, precision: 10, scale: 2
  end
end
```

Agregando price a cars

```
➜  fancy_cars git:(master) ✗ rake db:migrate
== 20161112015033 CreateCars: migrating ======================================
-- create_table(:cars)
   -> 0.0021s
== 20161112015033 CreateCars: migrated (0.0022s) =============================

== 20161112061756 AddPriceToCars: migrating ==================================
-- add_column(:cars, :price, :decimal, {:precision=>10, :scale=>2})
   -> 0.0004s
== 20161112061756 AddPriceToCars: migrated (0.0004s) =========================
```

Esquema General de nuestra bd

schema.rb

```ruby
# encoding: UTF-8
# This file is auto-generated from the current state of the database. Instead
# of editing this file, please use the migrations feature of Active Record to
# incrementally modify your database, and then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your
# database schema. If you need to create the application database on another
# system, you should be using db:schema:load, not running all the migrations
# from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended that you check this file into your version control system.

ActiveRecord::Schema.define(version: 20161112061756) do

  create_table "cars", force: true do |t|
    t.string   "make"
    t.string   "color"
    t.integer  "year"
    t.datetime "created_at"
    t.datetime "updated_at"
    t.decimal  "price",      precision: 10, scale: 2
  end

end
```

Renombrando columnas

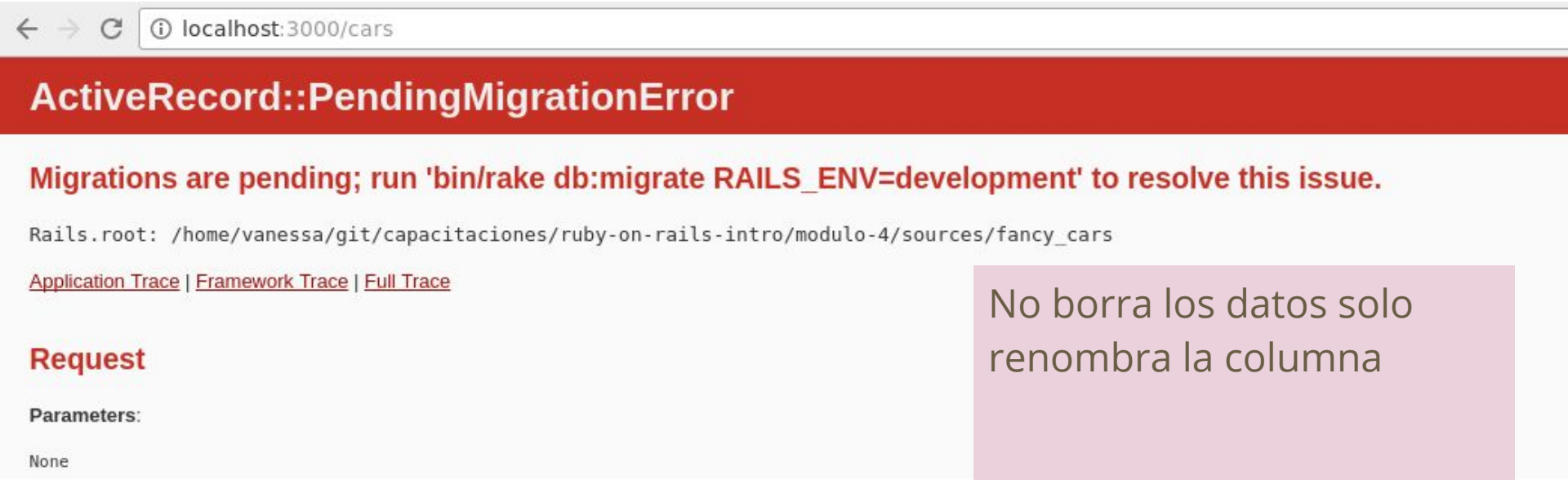
- `rename_column :table_name, :old_column_name, :new_column_name`

```
→  fancy_cars git:(master) ✗ rails g migration rename_make_to_company
      invoke  active_record
      create    db/migrate/20161112062410_rename_make_to_company.rb
```

Se debe agregar el código manualmente.

```
class RenameMakeToCompany < ActiveRecord::Migration
  def change
    rename_column :cars, :make, :company
  end
end
```

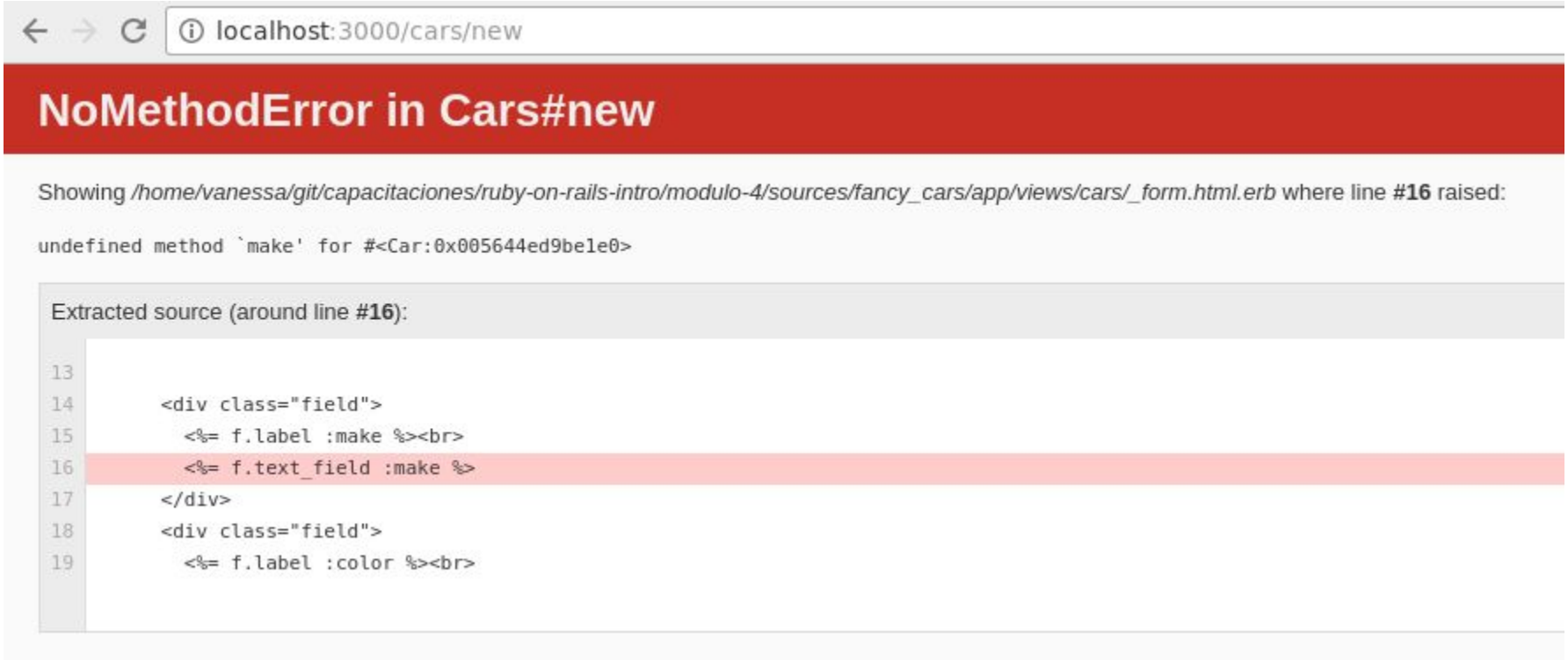
Renombrando columnas



No borra los datos solo renombra la columna

```
➜  fancy_cars git:(master) ✗ rake db:migrate
== 20161112062410 RenameMakeToCompany: migrating ==============================
-- rename_column(:cars, :make, :company)
   -> 0.0040s
== 20161112062410 RenameMakeToCompany: migrated (0.0041s) =====================
```

Renombrando columnas



Migrations irreversibles y mas

- A veces no queremos que los migrations se puedan retroceder, esto ocurre cuando la perdida de datos es inacpetable.
- http://edgeguides.rubyonrails.org/active_record_migrations.html

Entonces

- Se utilizan generadores de Rails para generar migrations.
- Se deben tener en cuenta los efectos secundarios en la aplicación debido a los migrations.

Ejercicio

- Agregar la columna **activ** a cars.
- Modificar el método de **creación** para almacenar **true**..
- Modificar el método **destroy** para almacenar **false** en la eliminación.
- Modificar el método **index**, para solamente mostrar los cars **activos**.