

Tarea PC3

Veremos los enunciados de las preguntas:

Las validaciones de modelos, al igual que las migraciones, se expresan en un mini-DSL integrado en Ruby, como muestra en el siguiente código. Escribe el código siguiente en el código dado.

```
class Movie < ActiveRecord::Base
  def self.all_ratings ; %w[G PG PG-13 R NC-17] ; end # shortcut: array of strings
  validates :title, :presence => true
  validates :release_date, :presence => true
  validate :released_1930_or_later # uses custom validator below
  validates :rating, :inclusion => { :in => Movie.all_ratings },
    :unless => :grandfathered?
  def released_1930_or_later
    errors.add(:release_date, 'must be 1930 or later') if
      release_date && release_date < Date.parse('1 Jan 1930')
  end
  @@grandfathered_date = Date.parse('1 Nov 1968')
  def grandfathered?
    release_date && release_date < @@grandfathered_date
  end
end
```

y comprueba tus resultados en la consola:

Comprobamos los resultados en consola:

```
angelic@LAPTOP-QS9DFR6S: /mnt/c/Users/ANGELIO/Desktop/DesarrolloDeSoftware/Semana7/myrottenpotatoes$ rails console
Warning: the running version of Bundler (2.2.3) is older than the version that created the lockfile (2.4.18). We suggest you to upgrade to the version that cr
eated the lockfile by running `gem install bundler:2.4.18`.
Loading development environment (Rails 7.0.8)
3.0.0 :001 > m = Movie.new(:title => '', :rating => 'RG', :release_date => '1929-01-01')
=>
#<Movie:0x00007f06108a0378
...
3.0.0 :002 > m.valid?
=> false
3.0.0 :003 > m.errors[:title]
=> ["can't be blank"]
3.0.0 :004 > m.errors[:rating]
=> []
3.0.0 :005 > m.errors[:release_date]
=> ["must be 1930 or later"]
3.0.0 :006 > m.errors.full_messages
=> ["title can't be blank", "Release date must be 1930 or later"]
3.0.0 :007 > |
```

Comprobamos que el resultado es falso, es se debido a que la clase Movie valida algunos campos del objeto Movie, instanciamos pero la consola nos alerta que **m no es valido, porque no tiene titulo, y porque la ‘release_date’ debe ser en 1930 o después;**

Explica el siguiente código:

```

1  class MoviesController < ApplicationController
2    def index
3      @movies = Movie.all
4    end
5    def show
6      id = params[:id] # retrieve movie ID from URI route
7      @movie = Movie.find(id) # look up movie by unique ID
8      # will render render app/views/movies/show.html.haml by default
9    end
10   def new
11     @movie = Movie.new
12   end
13   def create
14     if (@movie = Movie.create(movie_params))
15       redirect_to movies_path, :notice => "#{@movie.title} created."
16     else
17       flash[:alert] = "Movie #{@movie.title} could not be created: " +
18         @movie.errors.full_messages.join(", ")
19       render 'new'
20     end
21   end
22   def edit
23     @movie = Movie.find params[:id]
24   end
25   def update
26     @movie = Movie.find params[:id]
27     if (@movie.update_attributes(movie_params))
28       redirect_to movie_path(@movie), :notice => "#{@movie.title} updated."
29     else
30       flash[:alert] = "#{@movie.title} could not be updated: " +
31         @movie.errors.full_messages.join(", ")
32       render 'edit'
33     end
34   end
35   def destroy
36     @movie = Movie.find(params[:id])
37     @movie.destroy
38     redirect_to movies_path, :notice => "#{@movie.title} deleted."
39   end
40   private
41   def movie_params
42     params.require(:movie)
43     params[:movie].permit(:title, :rating, :release_date)
44   end
45 end
46
47

```

index: Obtiene todos los registros de la tabla Movie y los asigna a la variable de instancia @movies. Esto se utiliza generalmente para mostrar una lista de películas en la interfaz de usuario.

show: Recupera el ID de la película desde la ruta URI, luego busca y asigna la película correspondiente a la variable de instancia `@movie`. La vista por defecto renderizada será `app/views/movies/show.html.haml`.

new: Inicializa una nueva instancia de `Movie` y la asigna a la variable de instancia `@movie`. Esto se usa para mostrar el formulario para crear una nueva película.

create: Intenta crear una nueva película con los parámetros proporcionados (`movie_params`). Si la creación es exitosa, redirige a la ruta de películas (`movies_path`) con un mensaje de éxito. Si hay errores, muestra un mensaje de error y vuelve a renderizar la vista **'new'**.

edit: Encuentra y asigna a la variable de instancia `@movie` la película que corresponde al ID proporcionado en los parámetros. Esto se usa para mostrar el formulario de edición de una película existente.

update: Encuentra la película correspondiente al ID proporcionado y intenta actualizar sus atributos con los parámetros proporcionados (`movie_params`). Si la actualización es exitosa, redirige a la ruta de la película actualizada con un mensaje de éxito. Si hay errores, muestra un mensaje de error y vuelve a renderizar la vista **'edit'**.

destroy: Encuentra la película correspondiente al ID proporcionado y la elimina de la base de datos. Luego, redirige a la ruta de películas con un mensaje de éxito.

movie_params: Método privado utilizado para filtrar y permitir solo los parámetros específicos necesarios para crear o actualizar una película. En este caso, permite solo los parámetros **title**, **rating** y **release_date** dentro de los parámetros **movie** proporcionados en la solicitud.

Comprueba en la consola :

```
m = Movie.create!(:title => 'STAR wars', :release_date => '27-5-1977', :rating => 'PG')
m.title # => "Star Wars"
```

Comprobamos:

```

3.0.0 :001 > m = Movie.create!(title => 'STAR wars', :release_date => '27-5-1977', :rating => 'PG')
TRANSACTION (0.1ms) begin transaction
Movie Create (7.0ms) INSERT INTO "movies" ("title", "rating", "description", "release_date", "created_at", "updated_at") VALUES (?, ?, ?, ?, ?, ?) [{"title", "STAR wars"}, [{"rating", "PG"}, [{"description", nil}, [{"release_date", "1977-05-27 00:00:00"}, [{"created_at", "2023-11-15 13:36:15.954291"}, [{"updated_at", "2023-11-15 13:36:15.954291"}]]]]
TRANSACTION (8.8ms) commit transaction
=>
#<Movie:0x000056533ef5a5e0
...
3.0.0 :002 > m.title # => "Star Wars"
=> "STAR wars"
3.0.0 :003 >

```

Usamos el método **create!** para validar los datos que consignamos antes de insertar la tupla en la base de datos, por ello usa **transaction**; si la validación falla, manda una excepción.

SSO y autenticación de terceros

Generamos el modelo moviegoers y una migración

```

1 # Edit app/models/moviegoer.rb to look like this:
2 class Moviegoer < ActiveRecord::Base
3   def self.create_with_omniauth(auth)
4     Moviegoer.create!(
5       :provider => auth["provider"],
6       :uid => auth["uid"],
7       :name => auth["info"]["name"])
8   end
9 end

```

Actualizamos routes.db



```
1 Myrottenpotatoes::Application.routes.draw do
2   resources :movies
3   root :to => redirect('/movies')
4   get 'auth/:provider/callback' => 'sessions#create'
5   get 'auth/failure' => 'sessions#failure'
6   get 'auth/twitter', :as => 'login'
7   post 'logout' => 'sessions#destroy'
8 end
```

Y creamos su controlador sessions_controller:



```
1 class SessionsController < ApplicationController
2   # login & logout actions should not require user to be logged in
3   skip_before_filter :set_current_user # check you version
4   def create
5     auth = request.env["omniauth.auth"]
6     user =
7       Moviegoer.where(provider: auth["provider"], uid: auth["uid"]) ||
8       Moviegoer.create_with_omniauth(auth)
9     session[:user_id] = user.id
10    redirect_to movies_path
11  end
12  def destroy
13    session.delete(:user_id)
14    flash[:notice] = 'Logged out successfully.'
15    redirect_to movies_path
16  end
17 end
```

Claves Foráneas: