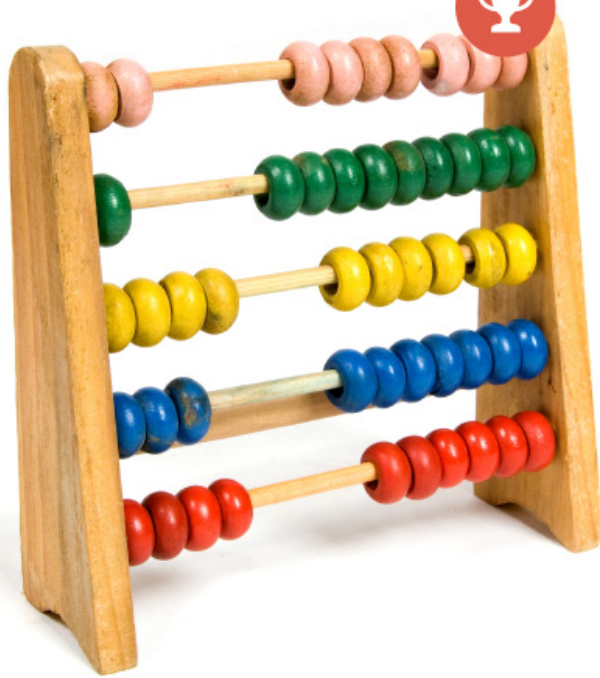


Programación con Ruby on Rails

Tienda (I)

Delio Tolivia





Indice

Creando la nueva aplicación
Creando MVC para productos
Migración
Arrancando por primera vez
Modificando el form
Resumen
Creando productos
Aplicando estilos
Resultado

Creando la nueva aplicación

- Tal y como hicimos en el tema anterior vamos a crear una nueva aplicación con el comando

rails new shop

- En nuestra nueva aplicación ya vamos a utilizar una base de datos. En este caso utilizaremos SQLite que es la que utiliza por defecto Rails.

Creando MVC para productos

- Para crear todo el “andamio” (scaffold) para los productos utilizaremos el comando

***rails generate scaffold Product title:string description:text
image_url:string price:decimal***

- Este comando nos crea el modelo **Product** (relacionado con la tabla **Products** que se creará en la base de datos), el controlador y la vista.
- Indicamos también que los productos van a tener un título, una descripción, la ruta a una imagen y un precio indicando para cada uno el tipo de datos que serán.

Migración (I)

- Uno de los ficheros que vamos a comprobar en primera instancia es el que se denomina **XXXXXXXXXXXXX_create_products.rb** que es creado en la carpeta **db/migrate**.
- Es el fichero de migración relacionado con nuestro modelo. Estos ficheros representan los cambios a realizar en la base de datos, pudiendo aplicarlos o indicar que se eliminen para volver a un estado anterior.

Migración (II)

- Vamos a modificarlo para indicar que nuestro precio va a tener una precision de 8 digits y mostrará dos detrás de la coma.

```
class CreateProducts < ActiveRecord::Migration  
  def change  
    create_table :products do |t|  
      t.string :title  
      t.text :description  
      t.string :image_url  
      t.decimal :price, precision: 8, scale: 2 #Modificamos aquí  
      t.timestamps  
    end  
  end  
end
```

Migración (III)

- Una vez modificado dicho archivo tenemos que aplicar la migración. Para ello se usa el comando rake en la consola. En este caso para aplicar la migración haremos (En Rails 5 se puede ejecutar en vez de con **rake** con **rails**):

rake db:migrate

- Con este comando buscará todas las migraciones que aún no han sido aplicadas a la base de datos. En nuestro caso creará la tabla productos tal y como está definida.

Arrancando por primera vez (I)

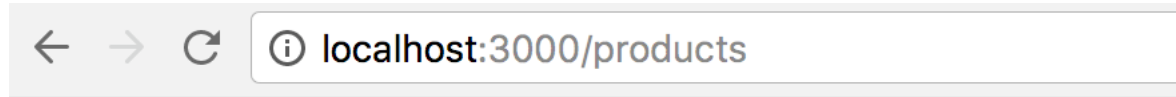
- Una vez llegados a este punto arrancamos el servidor

rails server

Para ver lo que tenemos hasta este momento podemos ir a la url

<http://localhost:3000/products>

Arrancando por primera vez (II)



Products

Title	Description	Image url	Price
-------	-------------	-----------	-------

[New Product](#)

- Nos ha creado una página con una lista de productos (de momento vacía) y un enlace para crear un nuevo producto. Si hacemos click ya nos aparece un formulario para poder crear productos.
- La plantilla del formulario se encuentra en ***views/products/_form.html.erb***

Modificando el form

- Vamos a editar la plantilla del formulario y por ejemplo indicar que el campo para la descripción tenga 10 líneas y 60 columnas. Para ello abrimos la plantilla y buscamos la línea donde se coloca dicho campos y la modificamos:

```
<%= form.text_area :description, id: :product_description, rows: 10, cols: 60 %>
```

- En nuestro ejemplo vamos a crear una librería. Rellenamos los datos de título, descripción, imagen (ponemos de momento el nombre de una imagen foo.jpg) y un precio y damos a crear.
- Esto nos lleva a una pantalla donde nos indica que el producto fue creado correctamente y un enlace a Back que nos lleva al listado inicial donde ya aparecerá junto a enlaces para mostrarlo, editarlo y borrarlo.

Resumen

- Con cuatro comandos de consola y unos pequeños cambios hemos conseguido:
 - Una página con un listado de productos.
 - Formularios para edición y creación de los mismos
 - Las tablas necesarias en la base de datos.
 - Todas las conexiones con la base de datos para que funcione.

Creando productos (I)

- Rails nos proporciona el fichero **db/seeds.rb** para poder definir una serie de elementos para crear de forma automática y rellenar nuestras tablas (está pensado para entornos de desarrollo y no tener que cada vez que lo vayamos a iniciar tener las tablas vacías).
- Cogemos el fichero **seeds.rb** (para no escribirlo se proporciona al alumno) y lo colocamos en la carpeta **db**. Si lo abrimos podemos ver:

```
Product.create!(title: 'CoffeeScript',  
  description: %{\<p>    CoffeeScript is JavaScript done right. It provides  
all of JavaScript's  functionality wrapped in a cleaner, more succinct  
syntax. In the first  book on this exciting new language, CoffeeScript guru  
Trevor Burnham  shows you how to hold onto all the power and flexibility  
of JavaScript  while writing clearer, cleaner, and safer code.    </p>},  
  image_url: 'cs.jpg',  
  price: 36.00)
```

- Utiliza el método **create!** Del modelo **Product** para crear el nuevo elemento.

Creando productos (II)

- Las imágenes que utilizaremos las colocamos en la carpeta ***app/assets/images***
- Una vez todo listo indicamos que ***rake*** rellene las tablas (hay que tener en cuenta que este proceso elimina todo el contenido anterior que existiese):

rake db:seed

- Con esto creamos nuestros nuevos productos en la aplicación.

Aplicando estilos (I)

- Los css para nuestras vistas están en la carpeta ***app/assets/stylesheets*** en nuestro caso el fichero ***products.css.scss*** (En Rails 5 solamente products.scss)
- Como vemos la extensión ***scss*** indica que es un formato especial de css (Sassy CSS) que nos permite algunas sintaxis especiales como anidar estilos, etc.
- Nos falta asignar a los elementos html los estilos que hemos creado.

Aplicando estilos (II)

- Si comprobamos los ficheros **.html.erb** podemos ver que no aparece el elemento **<head>** ni ninguna referencia a los css, etc. Rails mantiene todo esto en un fichero aparte que es usado como plantilla para todas las páginas de la aplicación es el fichero **application.html.erb** que está en la carpeta **app/views/layouts**
- Vamos a modificar nuestra etiqueta **body**:

<body class='<%= controller.controller_name %>'>

- De esta forma colocamos como clase de nuestro body el nombre del controlador de cada vista.

Aplicando estilos (III)

- Finalmente
escribimos la
vista
index.html.erb

```
<h1>Listing products</h1>
<table>
  <% @products.each do |product| %>
    <tr class="<%= cycle('list_line_odd', 'list_line_even')%>">
      <td>
        <%= image_tag(product.image_url, class:'list_image') %>
      </td>
      <td class="list_description">
        <dl>
          <dt><%= product.title %></dt>
          <dd><%= truncate(strip_tags(product.description),length:80) %></dd>
        </dl>
      </td>
      <td class="list_actions">
        <%= link_to 'Show', product %><br/>
        <%= link_to 'Edit', edit_product_path(product) %><br/>
        <%= link_to 'Destroy', product, method: :delete, data: { confirm: 'Are you sure?' } %>
      </td>
    </tr>
  <% end %>
</table>
<br>
<%= link_to 'New Product', new_product_path %>
```





Aplicando estilos (IV)

- Podemos ver distintas cosas que nos llaman la atención
 - Para conseguir colores alternos en las filas se utiliza el helper ***cycle()*** que aplica la clase ***list_line_even*** y ***list_line_odd*** automaticamente a cada fila.
 - Utilizamos en la descripción dos helpers más, ***strip_tags()*** para eliminar las etiquetas HTML de la descripción y ***truncate()*** para limitarla a los primeros 80 caracteres
 - El enlace a Borrar un elemento aparece el parámetro ***data: { confirm: 'Are you sure?'}***. Esto hace que antes de eliminar un producto aparezca una ventana para confirmarlo

Resultado

← → ↻ ⓘ localhost:3000/products

Listing products

	CoffeeScript CoffeeScript is JavaScript done right. It provides all of JavaScript...	Show Edit Destroy
	Programming Ruby 1.9 & 2.0 Ruby is the fastest growing and most exciting dynamic language ...	Show Edit Destroy
	Rails Test Prescriptions Rails Test Prescriptions is a comprehensive guide to testing ...	Show Edit Destroy

[New Product](#)