

# Programación con Ruby on Rails

Tienda (III)

*Delio Tolivia*





# Indice

Creando la tienda

Añadiendo una plantilla

Formateando el precio

# Creando la tienda (I)

- En nuestra aplicación ya podemos gestionar los productos. Ahora vamos a crear la tienda para lo que necesitaremos un controlador que llamaremos Store:

***rails generate controller Store index***

- Para acceder a este método iríamos a <http://localhost:3000/store/index> pero queremos que sea la pagina inicial de nuestra tienda. Para ello podemos definir las rutas en el fichero config/routes.rb

```
Rails.application.routes.draw do  
  get 'store/index'
```

```
resources :products  
# For details on the DSL available within this file, see http://guides.rubyonrails.org/routing.html  
root 'store#index', as: 'store'  
end
```

***Añadimos esta linea, indicamos cual es la raiz y generamos un metodo store\_path con as:'store'***

# Creando la tienda (II)

- Ahora ya podemos ver nuestro index del controlador Store en <http://localhost:3000/>
- Vamos a mostrar el listado de productos. Lo primero será obtenerlos desde nuestro controlador app/controllers/store\_controller.rb

```
class StoreController < ApplicationController  
  def index  
    @products = Product.order(:title) #Los obtenemos ordenados por título  
  end  
end
```

# Creando la tienda (III)

- Tenemos que crear nuestra vista `app/views/store/index.html.erb`

```
<% if notice %>
```

```
<p id="notice"><%= notice %> </p> Para mostrar mensajes
```

```
<% end %>
```

```
<h1>Your Pragmatic Catalog</h1>
```

```
<% @products.each do |product| %> Recorremos los productos
```

```
<div class="entry">
```

```
<%= image_tag(product.image_url)%> Método helper image_url
```

```
<h3><%= product.title %></h3>
```

```
<%= sanitize(product.description) %> Sanitize permite usar HTML
```

```
<div class="price_line">
```

```
<span class="price"><%= product.price %></span>
```

```
</div>
```

```
</div>
```

```
<% end %>
```

- Añadimos nuestro css a `app/assets/stylesheets/store.css.scss`

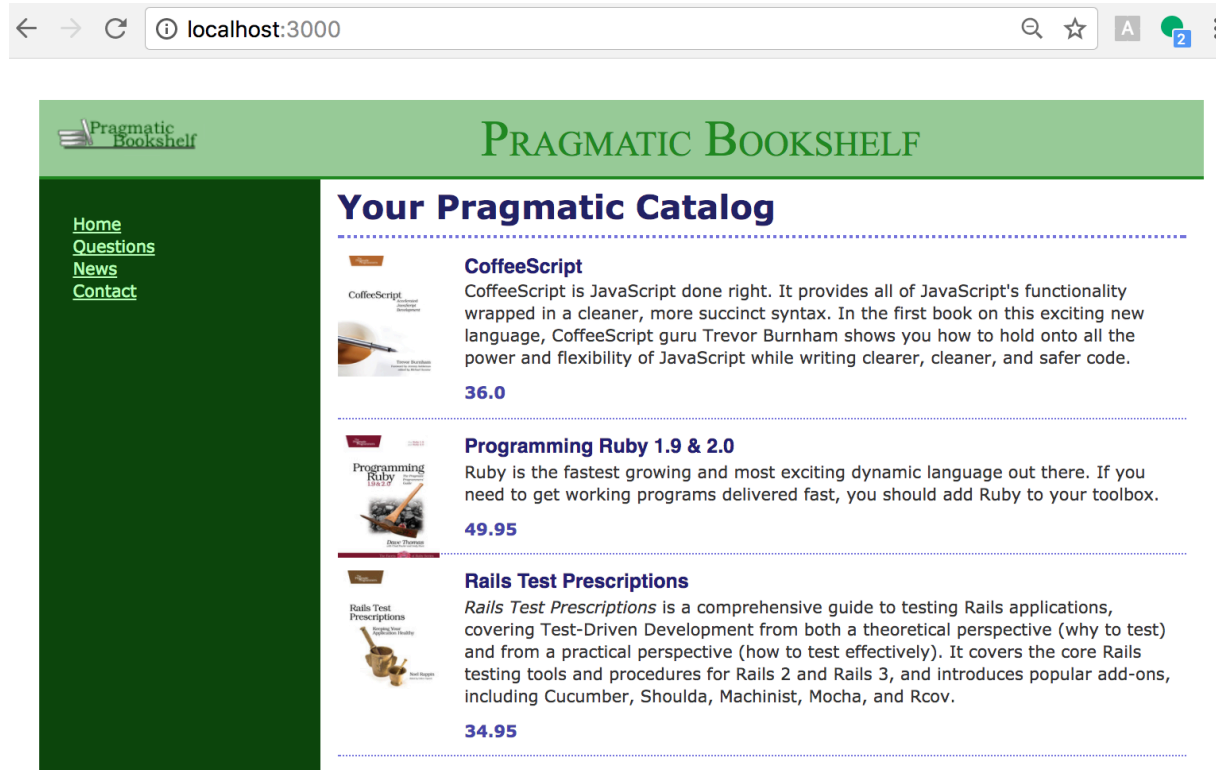
# Añadiendo una plantilla (I)

- Normalmente todas las páginas de una web comparten el mismo diseño general (plantilla). Vamos a añadir uno colocando un banner y una barra lateral. Para ello editamos nuestro `app/views/layouts/application.html.erb` (Cambiamos el contenido `<body>` por lo siguiente)

```
<div id="banner">
  <%= image_tag("logo.png") %>
  <%= @page_title || "Pragmatic Bookshelf"%>
</div>
<div id="columns">
  <div id="side">
    <ul>
      <li><a href="">Home</a></li>
      <li><a href="">Questions</a></li>
      <li><a href="">News</a></li>
      <li><a href="">Contact</a></li>
    </ul>
  </div>
  <div id="main">
    <%= yield %>
  </div>
</div>
```

# Añadiendo una plantilla (II)

- Sustituimos nuestro app/assets/stylesheets/application.css por un nuevo app/assets/stylesheets/application.css.scss




# Formateando el precio

- Para formatear el precio indicando la moneda Rails nos proporciona un método helper llamado ***number\_to\_currency()***. Es la forma correcta de hacerlo por si tenemos que internacionalizar nuestra página para que ponga el símbolo adecuado. Para ello en `app/views/store/idnex.html.erb` donde poníamos el precio colocaremos:

```
<span class="price"><%= number_to_currency(product.price) %></span>
```

---



**CoffeeScript**

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.

**\$36.00**

---