

Programación con Ruby on Rails

Tienda (VII)

Delio Tolivia





Indice

Terminando la compra
El botón de “Check Out”
Formulario de Check Out
Capturando el pedido

Terminando la compra

- Necesitamos “pedidos” (orders) los cuales tendrán varias líneas de pedido y información del cliente. Vamos a generar el scaffold de nuestro modelo “order” y actualizar la tabla de line_items para que se le añada un id del “order” asociado.

rails generate scaffold Order name adress:text email pay_type

rails generate migration add_order_to_line_item order:references

rake db:migrate

El botón de “check out” (I)

- Hay que poner en el carrito un botón que nos lleve a terminar el proceso de compra. Por ello modificamos `app/views/carts/_cart.html.erb`

.....

`</table>`

`<%= button_to 'Check Out', new_order_path, method: :get %>`

**`<%= button_to 'Empty cart', cart, method: :delete,
data: {confirm: 'Are you sure'} %>`**

.....

- Enlazamos al método `new` de nuestro nuevo modelo `order`.

El botón de “check out” (II)

- Necesitamos acceso al carrito desde nuestro nuevo controlador. Por lo que modificamos app/controllers/order_controller.rb

```
class OrdersController < ApplicationController  
  include CurrentCart  
  before_action :set_cart, only: [:new, :create]  
  before_action :ensure_cart_isnt_empty, only: :new  
  before_action :set_order, only: [:show, :edit, :update, :destroy]  
  .....
```

El botón de “check out” (III)

- Añadimos un hook al `before_action` cuando se va a ejecutar el método `new`. Tenemos que comprobar si el carrito está vacío por lo que definimos el método `ensure_cart_isnt_empty`. Modificamos `app/controllers/order_controller.rb`

```
private  
def ensure_cart_isnt_empty  
  if @cart.line_items.empty?  
    redirect_to store_url, notice: "Your cart is empty"  
  end  
end
```

Formulario de Check Out (I)

- Modificamos la vista del formulario para “order” en app/views/orders/new.html.erb

```
<div class="depot_form">  
  <fieldset>  
    <legend>Please Enter Your Details</legend>  
    <%= render 'form'%>  
  </fieldset>  
</div>
```

Formulario de Check Out (II)

- Modificamos el partial template app/views/orders/_form.html.erb

```
<%= form_for(@order) do |form| %>
  <% if @order.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(order.errors.count, "error") %> prohibited this order from being saved:</h2>

      <ul>
        <% @order.errors.full_messages.each do |message| %>
          <li><%= message %></li>
        <% end %>
      </ul>
    </div>
  <% end %>
```


Formulario de Check Out (III)

- Modificamos el partial template app/views/orders/_form.html.erb

```
<div class="field">
  <%= form.label :name %>
  <%= form.text_field :name, size:40 %>
</div>
```

```
<div class="field">
  <%= form.label :adress %>
  <%= form.text_area :adress, rows: 3, cols: 40 %>
</div>
```

```
<div class="field">
  <%= form.label :email %>
  <%= form.email_field :email, size: 40 %>
</div>
```

```
<div class="field">
  <%= form.label :pay_type %>
  <%= form.select_field :pay_type, Order::PAYMENT_TYPES, prompt: 'Select a payment method' %>
</div>
```

```
<div class="actions">
  <%= form.submit 'Place Order' %>
</div>
```

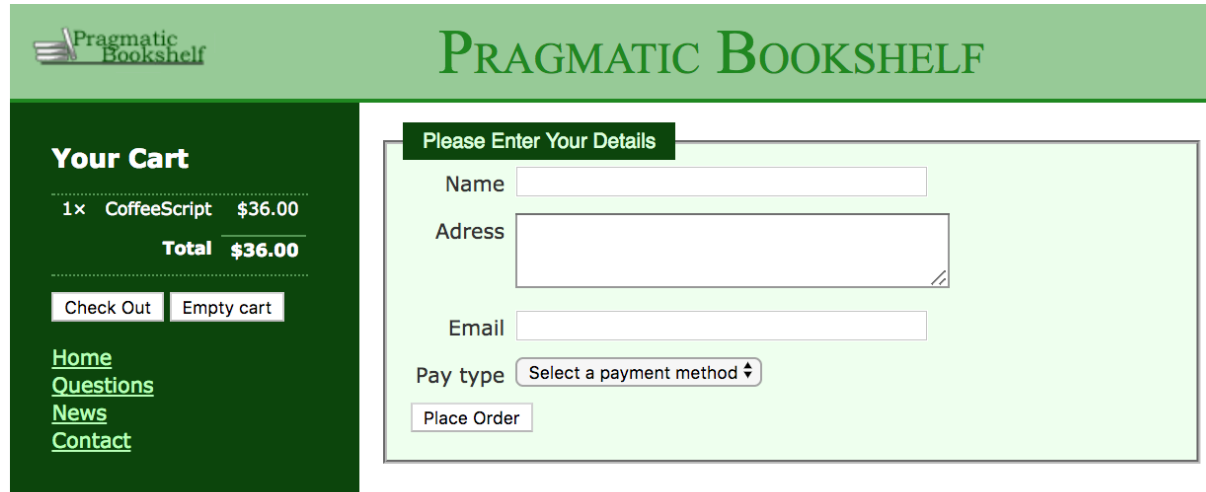
Formulario de Check Out (IV)

- Hemos indicado que los métodos de pago son un atributo del modelo Order y aún no los hemos definido. Modificamos app/models/order.rb

```
class Order < ApplicationRecord::Base  
  PAYMENT_TYPES = ["Check", "Credit card", "Purchase Order"]  
end
```

Formulario de Check Out (V)

- Modificamos nuestro app/assets/stylesheets/application.css.scss



Pragmatic Bookshelf

PRAGMATIC BOOKSHELF

Your Cart

1x CoffeeScript \$36.00

Total \$36.00

[Check Out](#) [Empty cart](#)

[Home](#)
[Questions](#)
[News](#)
[Contact](#)

Please Enter Your Details

Name

Adress

Email

Pay type

[Place Order](#)

```
.depot_form {  
  fieldset{  
    background: #efe;  
    legend{  
      color: #dfd;  
      background: #141;  
      font-family: sans-serif;  
      padding: 0.2em 1em;  
    }  
  }  
}  
form{  
  label{  
    width: 5em;  
    float: left;  
    text-align: right;  
    padding-top: 0.2em;  
    margin-right: 0.1em;  
    display: block;  
  }  
  select, textarea, input{  
    margin-left: 0.5em;  
  }  
  .submit{  
    margin-left: 4em;  
  }  
  br{  
    display: none;  
  }  
}
```

Formulario de Check Out (VI)

- Vamos a incluir validación de los campos en app/models/order.rb

```
class Order < ApplicationRecord  
  PAYMENT_TYPES = ["Check", "Credit card", "Purchase Order"]  
  validates :name, :adress, :email, presence:true  
  validates :pay_type, inclusion: PAYMENT_TYPES  
end
```

Capturando el pedido (I)

- En nuestra acción create() del nuevo controlador debemos:
 - Capturar los valores del formulario y crear un nuevo objeto de tipo Order
 - Añadir los “line items” de nuestro carrito a ese Order
 - Validar y guardar el Order, si falla mostrar los mensajes adecuados
 - Una vez guardado vaciar el carrito, volver al catalogo y mostrar un mensaje confirmando que se ha guardado el pedido

Capturando el pedido (II)

- Lo primero es definir la relación entre **Order** y **LineItem**. Modificamos `app/models/line_item.rb`

```
class LineItem < ApplicationRecord::Base  
  belongs_to :order #En Rails 5 añadir , optional:true  
  belongs_to :product #En Rails 5 añadir , optional:true  
  belongs_to :cart  
  
  .....
```

- Y `app/models/order.rb`

```
class Order < ApplicationRecord::Base  
  has_many :line_items, dependent: :destroy  
  PAYMENT_TYPES = ["Check", "Credit card", "Purchase Order"]  
  
  .....
```

Capturando el pedido (III)

- El método ***create()*** de `app/controllers/orders_controller.rb`

```
def create
  @order = Order.new(order_params)
  @order.add_line_items_from_cart(@cart)

  respond_to do |format|
    if @order.save
      Cart.destroy(session[:cart_id])
      session[:cart_id]=nil
      format.html { redirect_to store_url, notice: 'Thank you for your order.' }
      format.json { render :show, status: :created, location: @order }
    else
      format.html { render :new }
      format.json { render json: @order.errors, status: :unprocessable_entity }
    end
  end
end
```

Capturando el pedido (III)

- Nos falta el método ***add_line_items_from_cart()*** que hemos utilizado en nuestro controlador. Para ello lo añadimos a `app/models/order.rb`

```
def add_line_items_from_cart(cart)  
  cart.line_items.each do |item|  
    item.cart_id = nil  
    line_items << item  
  end  
end
```