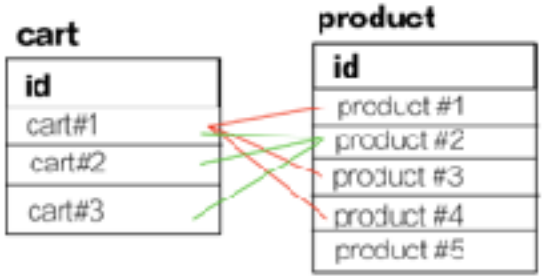
part2. Add shopping cart

yuan wang
2019 spring

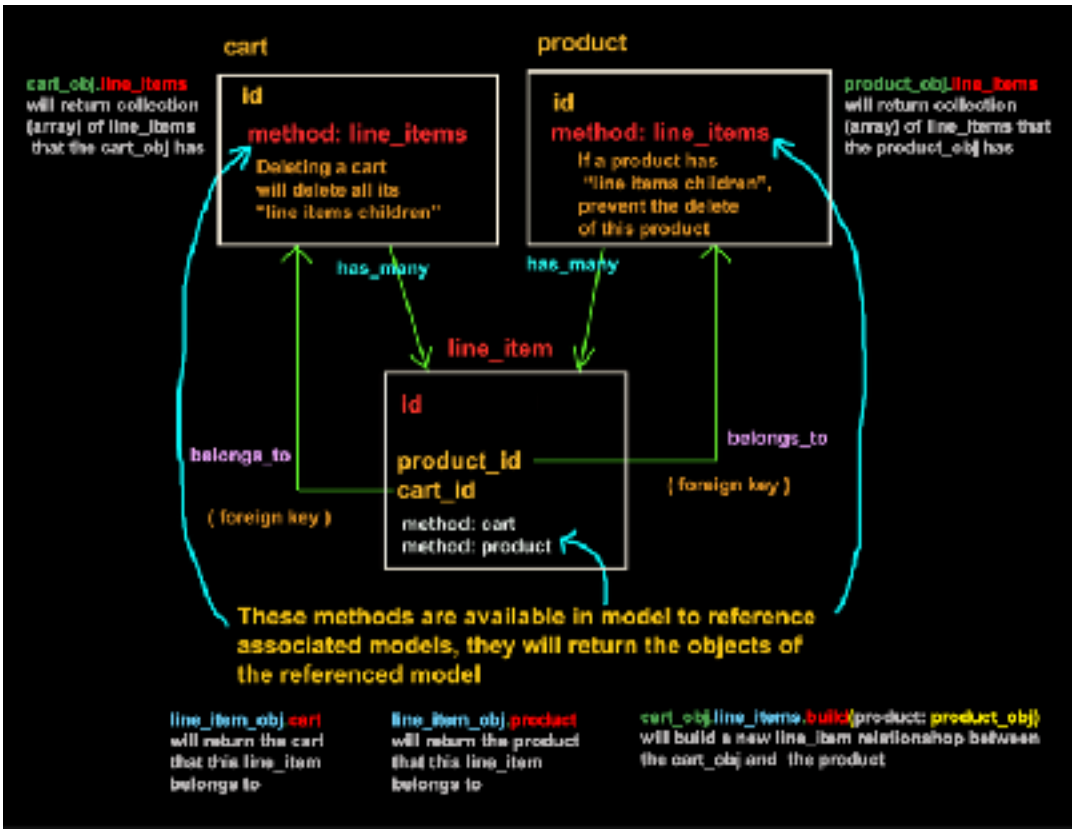
a shopping cart table is for remembering all shopping carts in the system. it just has an id.

the relationship between cart table and product table is many-to-many

1 product — belong to many carts (green)
1 carts - has many products (red)



create a third table to break many-to-many into two 1-to-many relationship like the following diagram. a third table called lineitems is created.



1 cart - has many line items
in line-item table: different line items belong to **same cart_id**
1 product - has many line items
in line-item table: different line_items belong to **same product_id**

example (description is not part of the table, it is used to show product here)

id	cart_id	product_id	product_description
1	1	8	nike red shoes
2	1	3	t-shirt white
3	2	2	t-shirt blue
4	2	9	computer
5	2	8	nike red shoes

for one user, his cart id will be saved in session

create scaffold for Cart table (just have cart_id) - reason to create scaffold is because we need models and controller actions

id

generate the scaffold for lineitem resource

```
> rails g scaffold lineitem product:references cart:belongs_to
```

id	product_id	cart_id

```
check out schema.rb to see table def, it should look like this:

 create_table "carts", force: :cascade do |t|
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
  end

  create_table "lineitems", force: :cascade do |t|
    t.integer  "product_id"
    t.integer  "cart_id"
    t.datetime "created_at",             null: false
    t.datetime "updated_at",             null: false
    t.integer  "quantity",   default: 1
  end

  add_index "lineitems", ["cart_id"], name: "index_lineitems_on_cart_id"
  add_index "lineitems", ["product_id"], name: "index_lineitems_on_product_id"

  create_table "products", force: :cascade do |t|
    t.string   "name"
    t.text     "description"
    t.string   "image"
    t.decimal  "price"
    t.datetime "created_at",  null: false
    t.datetime "updated_at",  null: false
  end
```

your lineitem model should already have belongs_to:

add has_many to cart and product model

```ruby
#if delete cart, then related lineitems are gone too
class Cart < ActiveRecord::Base
  has_many :lineitems,    dependent: :destroy
end

# if delete product, make sure no line items exist
class Product < ActiveRecord::Base
  has_many :lineitems

  before_destroy :make_sure_no_line_items

  validates :name, :description, :image, presence: true
  validates :price, numericality: {greater_than_or_equal_to: 0.01}
  validates :name, uniqueness: true
  validates :image, allow_blank: true,
  format: {with: %r{\.(gif|jpg|png)\Z}i, message: 'must be GIF, JPG, PNG images'}



 def make_sure_no_line_items
    if lineitems.empty?
      return true
    else
      errors.add(:base, 'Line Items present')
      return false
    end
  end
end
```

before_destroy: register a callback method, so that before the action (destroy) takes place, this method will be called.

The `before_destroy` callback needs a true/false value to determine whether or not to proceed.

If a before_* callback returns false, all the later callbacks and the associated action are cancelled.

add "Add to Cart" button (for each product.)
this button will lead to creation of lineitem in the lineitem table

add the following line to the view

```
<%= button_to 'Add to Cart', lineitems_path(product_id: product), class: 'add_to_cart' %>
```

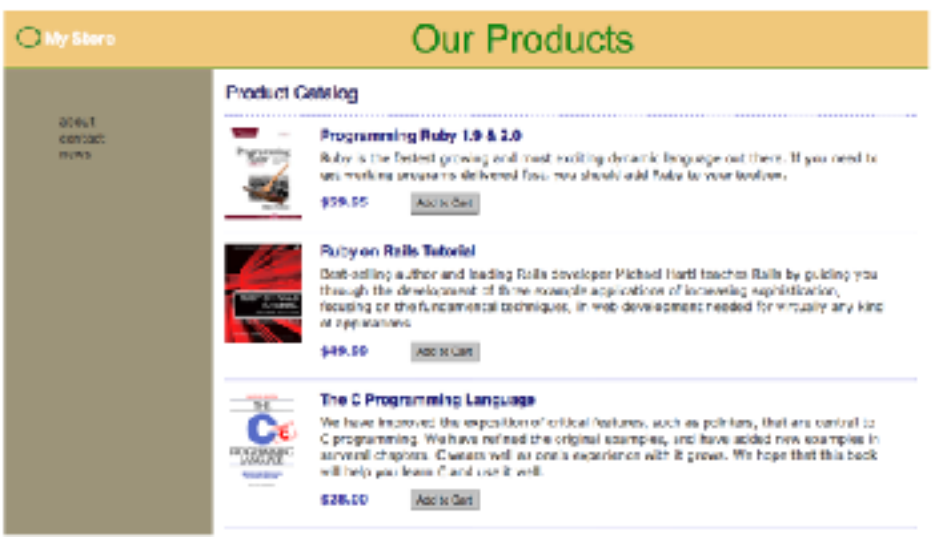
[by default, button_to is using POST]

change the style to make button in one line with price:
add style to app/assets/stylesheets/shopper.scss

```
 p, div.price_line {
  margin-left: 100px;
  margin-top: 0.5em;
  margin-bottom: 0.8em;
}
form, div {
  display: inline;
}
```

note: all .scss under app/asset/stylesheets will be loaded

now the shopper view should looks like this:



now make "Add to Cart" button work.

when click this button, a form will send a post request to lineitem(controller) to create a line item.

LineItemsController need to do these things:

- find shopping cart for the current session (or create one if it can not find one), then set instance variable @cart
- create a line item in lineitems table: link to the parent cart/product tables
- redirect to the cart view to show the content of the shopping cart

first, create set_cart method in a separate module in concerns folder

create Module CurrentCart in /app/controllers/concerns/current_cart.rb

```
module CurrentCart
  extend ActiveSupport::Concern


  def set_cart
  end

end
```

second, make sure lineitems controller call set_cart as before_action,

```
  include CurrentCart

  before_action :set_cart
```

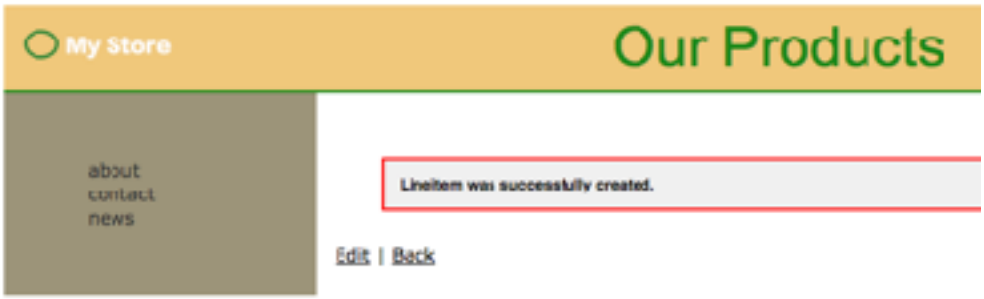
this is a callback function setup, so set_cart will be called before all actions

then modify the create action to create a lineitem

```
 def create

    product = Product.find(params[:product_id])
    @line_item = @cart.lineitems.build(product: product)

   respond_to do |format|
      if @line_item.save
        format.html { redirect_to @line_item,
                  notice: 'Line item was successfully created.' }
```

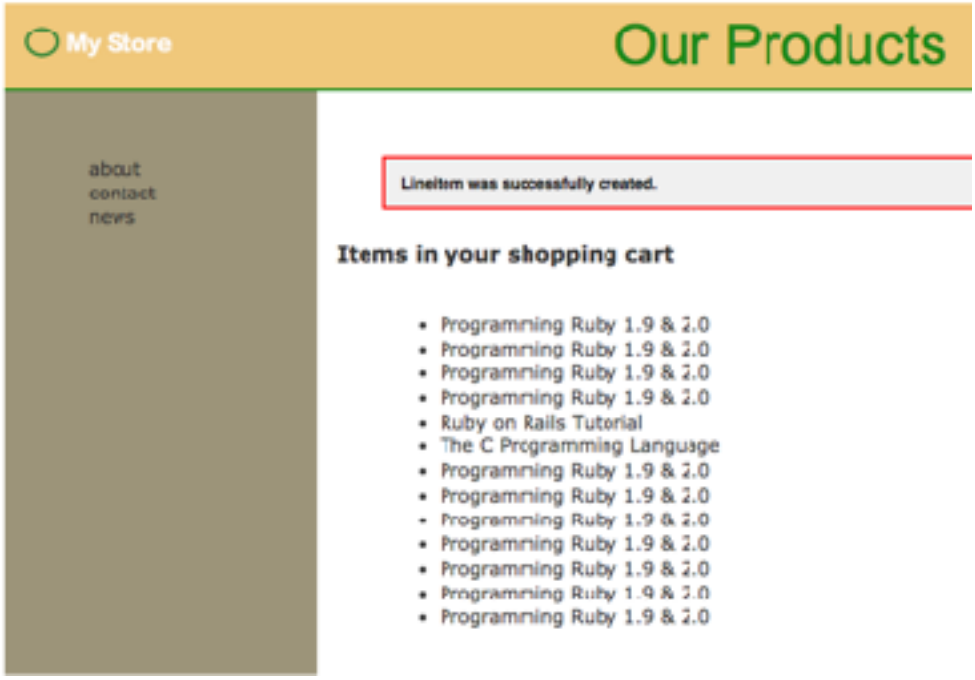
then, redirect to @line_item.cart instead of @lineitem

you should get something like this:



The above view is the default view, modify the view to display all items product name in current shopping cart

you should be able to add more products:



make changes to display the quantity of items instead of repeating it:

first, add a "quantity" field to lineitems table (by creating a migration)

add default 1 to the migration generated

```
add_column :lineitems, :quantity, :integer, default: 1
```

add a method in model to check if a lineitem exist in the lineitems table, if exist, increment the quantity, if not, create this lineitem

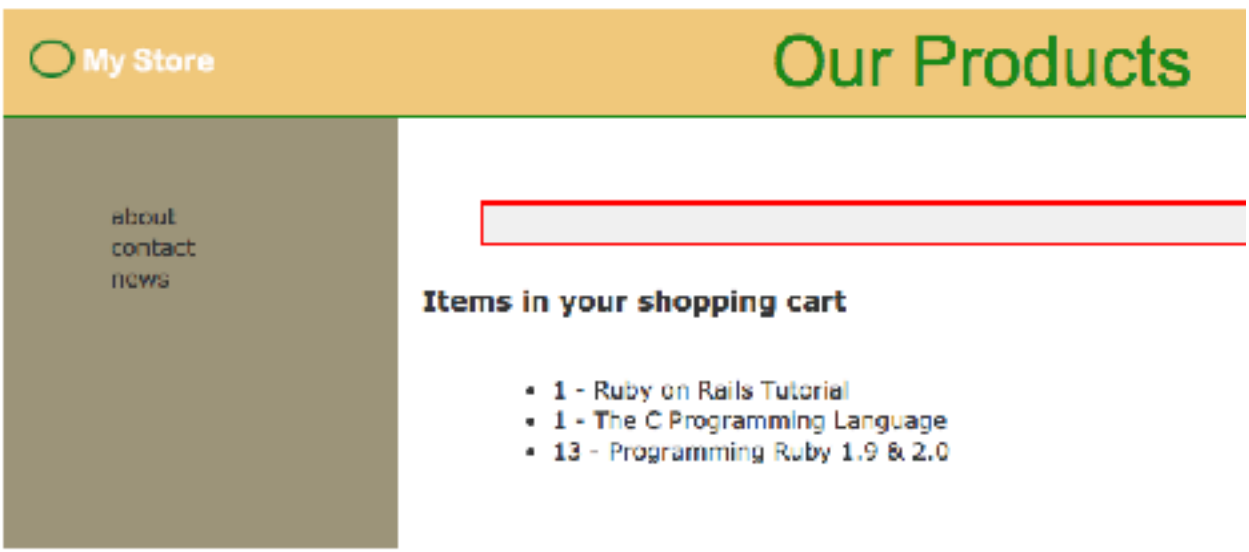
```
  def add_item(product_id)
    current_item = lineitems.find_by(product_id: product_id)

    # more code here

 end
```

change controller to use this add_item() method on @cart

```
@line_item = @cart.add_item(product.id)
```

change view to display quantity for each product in the shopping cart

you should see something like this (there might be duplicate products):



---

add "Empty cart" button in the shopping cart view:

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

modify delete action in cart controller

```
  def destroy
    @cart.destroy if @cart.id == session[:cart_id]
   session[:cart_id] = nil

    respond_to do |format|
      format.html { redirect_to cart_url, notice: 'Your shopping cart is currently empty.' }
```

```
      format.json { head :no_content }
    end
  end
```

then change redirect to catalog list

Add total price

modify the shopping cart view so that the values are displayed using <table>

quantity	name	price
Total		

The total row will display total price for the current cart

Implement total_price in Cart model

```
  def total_price
  end
```

add item_total_price to lineitem model

```
  def item_total_price
  end
```

move the cart display to the side bar

move the content of cart's show view to a partial view and render this view in the side bar:

```
     <div id='cart'>
         <% = render @cart %>
      </div>
```

in this case, @cart is an instance of a model,
it will call partial _cart.html.erb by default

create /app/views/carts/_cart.html.erb for displaying cart

now the main area is still displaying cart,

we need to display catalog

redirect to shopper_url after creating a line items.

if @cart is not available in other controller, add the following to it.

```
  include CurrentCart
  before_action    :set_cart
```

add style of the cart display
to stylesheets/application.scss

```
#columns {
  background: #141;

  #main {
    margin-left: 17em;
    padding: 1em;
    background: white;
  }

  #side {
    float: left;
    padding: 1em 2em;
    width: 13em;
    background: #141;

    form, div {display: inline;}
    input {font-size: small;}
    #cart {font-size: small; color: white;
      table {
         border-top: 1px dotted #599;
         border-bottom: 1px dotted #599;
         margin-bottom: 10px;
      }

    }



    ul {
      padding: 0;
      li {
        list-style: none;
        a {
          color: #bfb;
          font-size: small;
        }
      }
    }
  }
}
```

it will become something like this: