[Lesson 6.1: Products, Categories, and Brands](#h.c8uto9a0rby8)

[Gem of the Day](#h.z1ew4uzcr1jf)

[Our Checklist](#h.ud2i55ubgsha)

[Feature Branch](#h.yu3ezkp6q4me)

[Resources and Model](#h.oergoyidmvxo)

[Scaffolds](#h.nlghpph7bc1o)

[Backroom vs Storefront Views](#h.t18cq7hgshjp)

[Backroom](#h.y1ldwgv7fwdb)

[Add Paperclip Images to Products](#h.z1mq7sniamh1)

[Updating Product View](#h.sizl1x9p380o)

[Products: Data Entry](#h.cqw8zrqk7osk)

[Storefront](#h.adzbike5skf3)

[All Items View](#h.fwas5njb27vk)

[Bootstrap Modals](#h.ojijbfkn9u7t)

[Homework](#h.kdh81q3n01fs)

# Lesson 6.1: Products, Categories, and Brands

Let’s add e-commerce capability to our Doggy Daycare app.

## Gem of the Day

Shut the F\*ck Up <https://github.com/tpope/gem-shut-the-fuck-up>

Roll Call

Review last class - what gems? what learned about devise? what about alternatives?

## Our Checklist

* Let's start with a customer-free store (as shady as that sounds).
* We'll have products that are shown all together or by category.
* You can buy one or more of each product, and we'll add them to a cart.
* Then we'll tie a user to that cart ("check out").
* At the end we'll learn how to implement a payment process.

## Feature Branch

First, create a feature branch called “shop” that we will use as we develop this feature over the next few days:

git checkout -b shop

## Resources and Model

Product

* Name
* Price
* Quantity
* Description
* Brand?
* category\_id?

Category

* Name

We won't have a Cart resource, exactly, but two separate resources: one to store an item in as you shop, the other gather those items together:

LineItem

* product\_id
* quantity
* line\_item\_total
* order\_id

Order

* sub\_total
* sales\_tax
* grand\_total
* customer\_id\*

\* how we'll eventually tie a user (customer) to what they want to buy

### Scaffolds

* rails g scaffold Category name:string
* rails g scaffold Product name:string price:decimal quantity:integer description:text brand:string rating:integer category:references
* rake db:migrate
* associate products and categories (has many, belongs to)

## Backroom vs Storefront Views

We will use the scaffolded Views as our "backroom", a place for employees (admin users) can create and update products and categories.

We'll create new views (within a new controller) to act as our "storefront" - views for our customers to peruse products (all or by category) and purchase.

### Backroom

Add Bootstrap table class to the scaffold index table for products and categories.

Then, seed categories:

Breed.create(breed: 'Australian Shepherd')

Breed.create(breed: 'Border Collie')

Breed.create(breed: 'Jack Russell')

Breed.create(breed: 'Pit Bull')

Breed.create(breed: 'Shih-tzu')

Breed.create(breed: 'Standard Poodle')

Breed.create(breed: 'Miniature Poodle')

Breed.create(breed: 'Doberman')

Breed.create(breed: 'Bulldog')

Breed.create(breed: 'Alaskan Malamute')

Breed.create(breed: 'Eskimo')

Breed.create(breed: 'Chihuahua')

Breed.create(breed: 'Yorkie')

Breed.create(breed: 'Boxer')

Breed.create(breed: 'Mutt')

Owner.create(first\_name: "Sia", last\_name: "Karamalegos", phone: "504-555-1212", emerg\_name: "Gail Karamalegos", emerg\_phone: "979-555-1212")

Dog.create(name: "Priscilla", owner\_id: 1, breed\_id: 1, vet: "Magazine Street Animal Clinic", in\_daycare: true)

Dog.create(name: "Harry", owner\_id: 1, breed\_id: 3, vet: "Magazine Street Animal Clinic", in\_daycare: false)

categories = Category.create([{name: 'toys'}, {name: 'treats'}, {name: 'beds'}, {name: 'food'}])

Run rake db:drop, then rake db:setup (migrates and seeds).

## Add Paperclip Images to Products

rails g paperclip product image

rake db:migrate

Add validations to products model:

has\_attached\_file :image, :styles => { :medium => "300x300#", :thumb => "100x100#" }, :default\_url => "missing\_product\_:style.png"

validates\_attachment\_content\_type :image, :content\_type => /\Aimage\/.\*\Z/

Add :image to the product\_params at the bottom of the Products controller.

## Updating Product View

In the Product form, we'll remove the div for taking a rating (the customer's will be given that power later), and we'll put in a field for taking an uploaded image, and we'll tweak the field that takes the category\_id so that it's a drop-down menu to select a category.

<div class="field form-group">

<%= f.label :image %><br>

<%= f.file\_field :image %>

</div>

<div class="field form-group">

<%= f.label :category\_id %><br>

<%= f.collection\_select :category\_id, Category.all, :id, :name, {prompt: "Select a category"}, {class: "form-control"} %>

</div>

### Products: Data Entry

Now you can go through and make a couple products for us to sell!

One hint for improvement: put "number\_to\_currency" (in the Ruby tags) in front of the product.price to add a dollar sign and round to two decimal places, like so:

<%= number\_to\_currency product.price %>

## Storefront

Let's give the customers three ways to view products:

* All at once (like the Products index page)
* By Category (will use parameters to filter all items)
* By Brand (the Product attribute - will set up later maybe using JavaScript or AJAX)

$ rails g controller Storefront all\_items

### All Items View

Since customers will see this, let’s make it a bit prettier than our back-office index pages:

* The "+ quick info" link will bring up a modal with product highlights.
* Star/Rating system is commented out until we set that for the Users.

First, add to the Storefront controller, all\_items method:

@products = Product.all

Then in the all\_items view:

<h1>Welcome to Fur-Baby Fun-Time Store!</h1>

<p>Where you can spend all your money on your fur babies!</p>

<div class="row">

<% @products.each do |product| %>

<div class="col-md-4">

<div class="panel panel-default height">

<div class="panel-body">

<%= image\_tag product.image.url(:thumb), class: "img-responsive" %>

<p><a href="#">

<span class="glyphicon glyphicon-plus"></span> quick info

</a></p>

<p><strong><%= number\_to\_currency product.price %></strong></p>

<p><%= product.name %></p>

<p>Made by: <%= product.brand %></p>

<p>

<%# product.rating.times do %>

<!-- <span class="glyphicon glyphicon-star yellow"></span> -->

<%# end %>

</p>

</div>

</div>

</div>

<% end %>

</div>

Then in the storefront.scss:

.height {

height: 260px;

}

### Bootstrap Modals

Now let’s get those quick info modals working!

Let's test a demo modal on our page before fitting it into our product loop.

Go to: <http://getbootstrap.com/javascript/#modals>

Copy the code from the "Live Demo" section, and paste it in the bottom of your all\_items.html.erb file:

<!-- Button trigger modal -->

<button type="button" class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">

Launch demo modal

</button>

<!-- Modal -->

<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>

<h4 class="modal-title" id="myModalLabel">Modal title</h4>

</div>

<div class="modal-body">

...

</div>

<div class="modal-footer">

<button type="button" class="btn btn-default" data-dismiss="modal">Close</button>

<button type="button" class="btn btn-primary">Save changes</button>

</div>

</div>

</div>

</div>

Check that it works (just mostly blank modal pops up)

Let's use that modal code to create our own "quick info" modal. First, take the functionality from the demo button and move it to the anchor tags around "+ quick info". In addition, add "\_<%= product.id %>" to the data-target, this will let us have a unique element ID each time we loop through.

<!-- plus button trigger modal -->

<p><a href="#" data-toggle="modal" data-target="#myModal\_<%= product.id %>">

<span class="glyphicon glyphicon-plus"></span> quick info

</a></p>

Next move the modal code into a space between the <% end %> tag and the last </div> to close above that. At the top of this code you'll find id="myModal" add "\_<%= product.id %>" to that so it corresponds to the data-target in the anchor tag above. In addition, you will replace the modal title with the product name, and add the product price and description in the body:

</div> <!-- col-md-4 -->

<!-- Modal -->

<div class="modal fade" id="myModal\_<%= product.id %>" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>

<h4 class="modal-title" id="myModalLabel"><%= product.name %></h4>

</div>

<div class="modal-body">

<p><%= product.price %></p>

<p><%= product.description %></p>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-default" data-dismiss="modal">Close</button>

<button type="button" class="btn btn-primary">Save changes</button>

</div>

</div>

</div>

</div> <!-- modal -->

<% end %>

Check that your modals are working, and then commit your changes!

## Homework

Homework: Add the medium photo to the modal. Make your views prettier. Move sign-in/sign-out links to right on navbar. Optional: Move dog search box to dogs index page.