MARTIN BROCHHAUS

ENTREPRENEUR, SOFTWARE ENGINEER, FREELANCE WEBDESIGNER,
HOUSE DJ, PHOTOGRAPHER NOOB, WORLD TRAVELER

19TH MAY 2011

DJANGO-SHOP ADVENTURES: CREATING PRODUCTS AND ADMIN

In my last post on this series I described **how to install django-shop**. Today we will do the next most important thing, which is to create models for our products.

Create a product model

I will assume a shop that resells large LED lamps for streets or office buildings. My client gave me a huge list of articles which I will transform into a product class like this:

```
1  from django.db import models
2  from shop.models.productmodel import Product
3
4
5  class Category(models.Model):
6      """Master data: Names of categories."""
7      name = models.CharField(max_length=256)
8
9      def __unicode__(self):
10         return self.name
11
12
13 class Distributor(models.Model):
14     """Master data: Information about distributors."""
15     name = models.CharField(max_length=256)
16
17     def __unicode__(self):
18         return self.name
```

```
19
20
21 class LED(Product):
22     """Product class for LED lamps."""
23     article_number = models.IntegerField()
24     distributor = models.ForeignKey(Distributor)
25     image1 = models.ImageField(blank=True, upload_to='myshop_images
26     image2 = models.ImageField(blank=True, upload_to='myshop_images
27     # name - resides in Product base class
28     category = models.ForeignKey(Category, blank=False, null=False)
29     weight = models.FloatField(blank=True, null=True)
30
31     def __unicode__(self):
32         return '[%s] %s' % (self.article_number, self.name)
```

gistfile1.py hosted with ❤ by GitHub view raw

This is a first draft and the models will probably change later. I don't know if managing categories this way is the best approach. For once I don't plan to have categories that can be nested so this will probably be good enough. For slightly more complex categorization have a look at **django-shop-simplecategories**.

Create a product admin

Having a model is all good but it doesn't help a lot when there is no way for my customer to enter it. So next we will add an admin.py to our shop application:

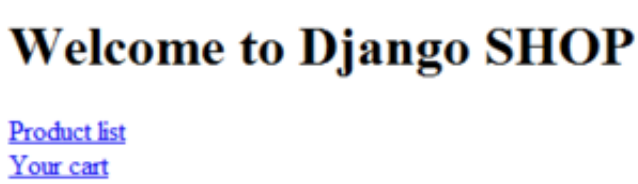
```
1  from django.contrib import admin
2
3  from myshop import models as shop_models
4
5
6  class LEDAdmin(admin.ModelAdmin):
7      prepopulated_fields = {"slug": ("name",)}
8
9  admin.site.register(shop_models.Category)
10 admin.site.register(shop_models.Distributor)
11 admin.site.register(shop_models.LED, LEDAdmin)
```

gistfile1.py hosted with ❤ by GitHub view raw

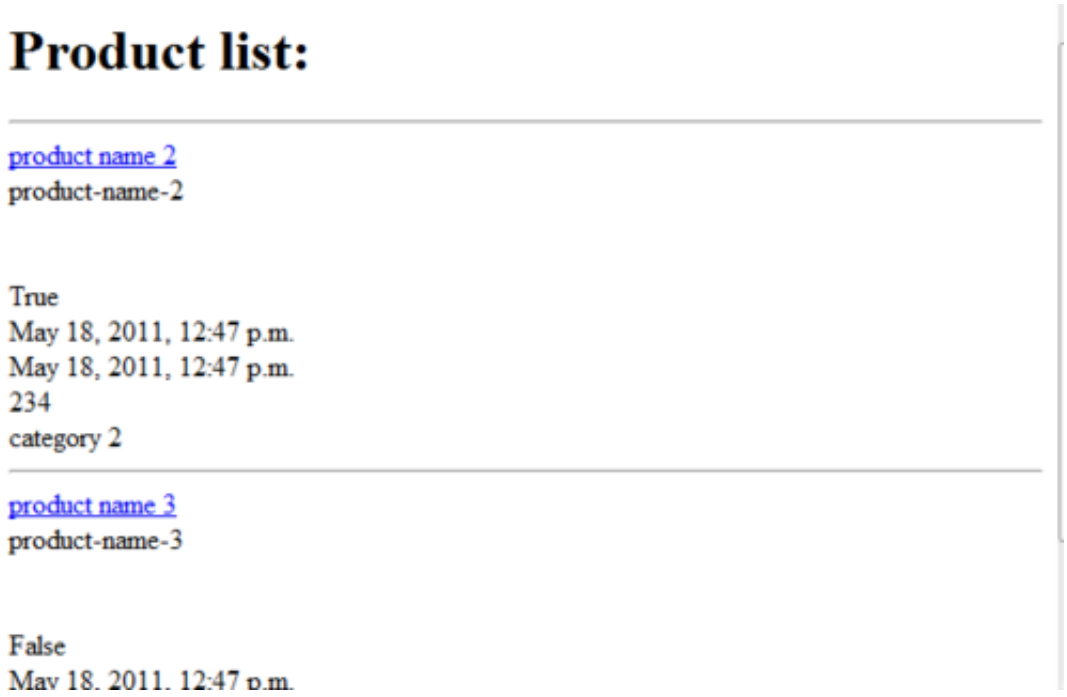
Notice line seven, where we added a **prepopulated_field** to our product's admin. As of now you have to do this manually but this will probably become a standard in the shop sooner or later. This makes sure that your slug gets generated automagically while you type in your product's name.

What do we have so far?

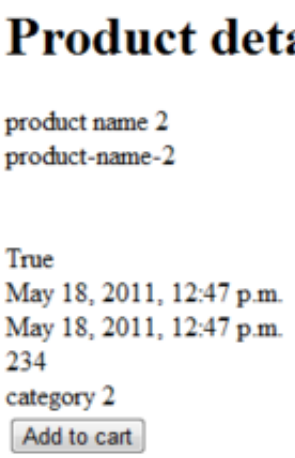
After entering some simple products we can browse our simple shop and gasp in amazement:



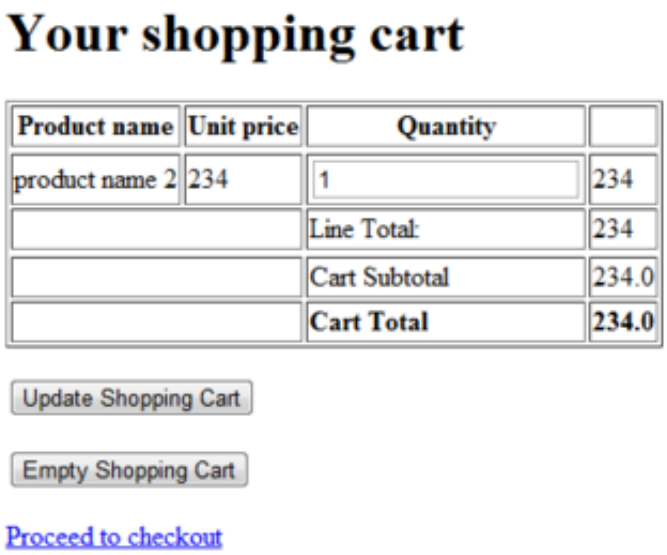
First we see the shop's main page. The simple basic template that ships with the shop just gives us two links but I can imagine having a nice welcome page here, showing best selling items, recent orders and current promotional offers.



When we click at "Product list" we get a list of all products. This is basically a class based ListView based on the `Product` model. Every model that inherits from `Product` will be shown in this list. Sweet!



We can go one step further and see the details view of a certain product. Again this is basically just a class based `DetailsView` which (theoretically) makes it very simple to create our own flavors of detail views for different kinds of products (basically linking to different templates). What we see here are standard fields that come with the **`Product` base model**. Obviously we will want to override this template later and show our own product information.



Once you add a product to your cart (which can even be done anonymously) you will see a summary of your current shopping cart. If you change your mind, you can update or even empty the cart, if not, you can proceed to checkout.

The checkout screen is where our journey ends for once. I guess we have to provide some kind of shipping method first and probably some kind of payment processor. I there is nothing else important that I have left out, I will talk about these in my next post.

The code of this project can be found on **Github (tag v0.2)**.

« Previous Next »
