Tutorial 02 to do in class – Remember to upload the repo link to Teams

A. Configuring our database

- Open an Anaconda Prompt with your environment activated and run the following command:
  - Execute in Terminal

```
conda install factory_boy
```

- In your environment, open VS Code and run the following commands in the Terminal:
  - Execute in Terminal

```
python manage.py makemigrations
python manage.py migrate
```

- The following file should appear in your project folder:

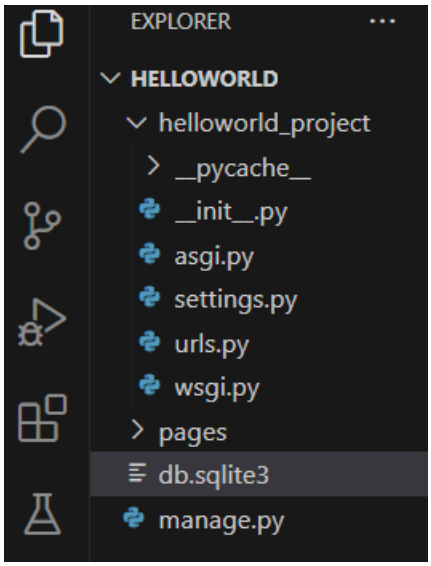


Figure 10-2. Database creation.

B. A basic product model

Model

- In your *pages* app got to *models.py* and add the following code:.

Add Code

```
class Product(models.Model):

    name = models.CharField(max_length=255)

    price = models.IntegerField()

    created_at = models.DateTimeField(auto_now_add=True)

    updated_at = models.DateTimeField(auto_now=True)
```

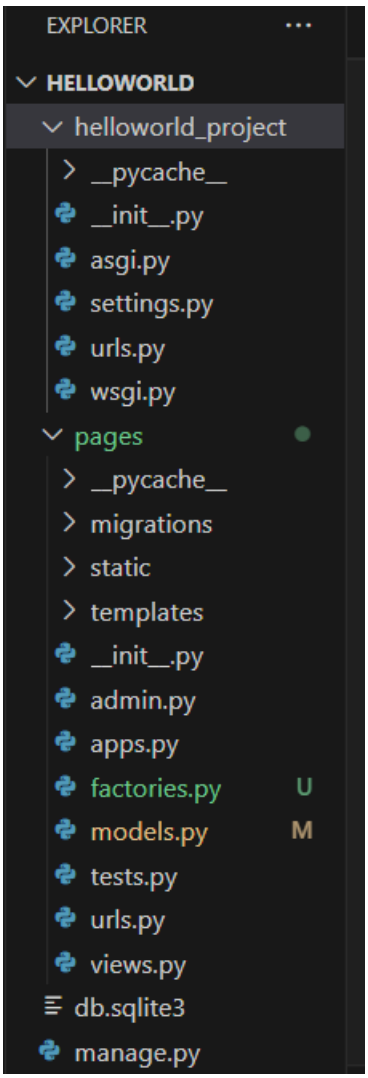
Factories

- Go to *pages* and create a file *factories.py*, with the following content:

Add Entire Code

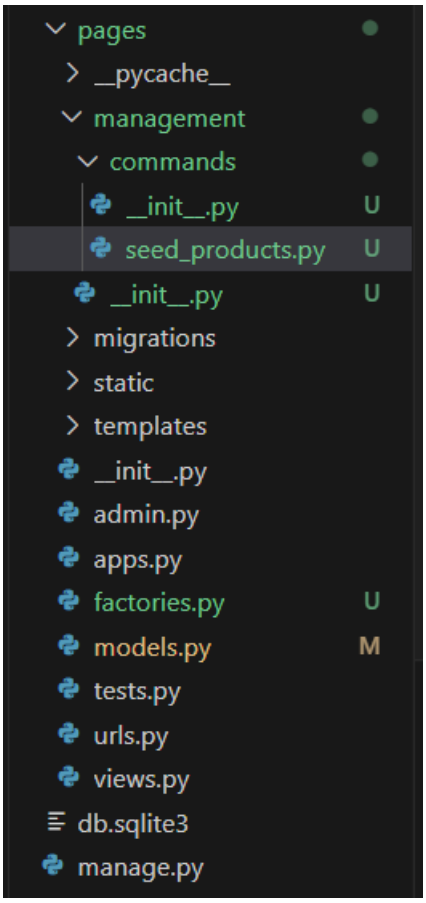
```
import factory

from .models import Product


class ProductFactory(factory.django.DjangoModelFactory):

    class Meta:

        model = Product

    name = factory.Faker('company')

    price = factory.Faker('random_int', min=200, max=9000)
```

- Your project tree should look like this:



DatabaseSeeder

- In *page* add a *management* directory. Create a file named *__init__.py* inside (leave it empty), then create a *commands* directory inside. In */management/commands* create two files named *__init__.py* and a *seed_products.py.* Your app tree should look like this:



- Go to *pages/management/commands/seed_products.py* and add the following code:

Add Entire Code

```
from django.core.management.base import BaseCommand
from pages.factories import ProductFactory

class Command(BaseCommand):
    help = 'Seed the database with products'

    def handle(self, *args, **kwargs):
        ProductFactory.create_batch(8)
        self.stdout.write(self.style.SUCCESS('Successfully seeded products'))
```

- In the Terminal, go to the project directory, and execute the following:

Execute in Terminal

```
python manage.py makemigrations
python manage.py migrate
python manage.py seed_products
```

- You should eight new products added into the database.

db.sqlite3

SELECT * FROM pages_product | Schema | Query Editor | Auto Reload | Find | Other Tools...

	id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT	name varchar(255) NOT NULL	price INTEGER NOT NULL	created_at datetime NOT NULL	updated_at datetime NOT NULL
1	1	Harmon Inc	4931	2023-08-14 13:11:3…	2023-08-14 13:11:3…
2	2	Wilson, Cisneros a…	4072	2023-08-14 13:11:3…	2023-08-14 13:11:3…
3	3	Pace Ltd	7520	2023-08-14 13:11:3…	2023-08-14 13:11:3…
4	4	Schaefer-King	3536	2023-08-14 13:11:3…	2023-08-14 13:11:3…
5	5	Brown, Hamilton an…	4671	2023-08-14 13:11:3…	2023-08-14 13:11:3…
6	6	Levy, Griffin and …	7760	2023-08-14 13:11:3…	2023-08-14 13:11:3…
7	7	Kelley, Martinez a…	5496	2023-08-14 13:11:3…	2023-08-14 13:11:3…
8	8	Le-Richard	1533	2023-08-14 13:11:3…	2023-08-14 13:11:3…

C. A basic listing products from database

View/Controller

- In *pages/views.py*, make the following changes in **bold**.

Modify Bold Code

```
from django.shortcuts import render, redirect, get_object_or_404
from django.views.generic import TemplateView, ListView
from django.views import View
from django.http import HttpResponseRedirect
from django.urls import reverse
```

```
from django import forms
from django.core.exceptions import ValidationError
from .models import Product

class Product:
    products = [
        {"id":"1", "name":"TV", "description":"Best TV", "price":50},
        {"id":"2", "name":"iPhone", "description":"Best iPhone", "price":150},
        {"id":"3", "name":"Chromecast", "description":"Best Chromecast", "price":80},
        {"id":"4", "name":"Glasses", "description":"Best Glasses", "price":30}
    ]
...
class ProductIndexView(View):
    template_name = 'products/index.html'

    def get(self, request):
        viewData = {}
        viewData["title"] = "Products - Online Store"
        viewData["subtitle"] =  "List of products"
        viewData["products"] = Product.objects.all()

        return render(request, self.template_name, viewData)



class ProductShowView(View):
    template_name = 'products/show.html'



    def get(self, request, id):

        # Check if product id is valid
        try:
            product_id = int(id)
            if product_id < 1:
                raise ValueError("Product id must be 1 or greater")
            product = get_object_or_404(Product, pk=product_id)
        except (ValueError, IndexError):
```

```
        # If the product id is not valid, redirect to the home page
        return HttpResponseRedirect(reverse('home'))

    viewData = {}
    product = get_object_or_404(Product, pk=product_id)
    viewData["title"] = product.name + " - Online Store"
    viewData["subtitle"] =  product.name + " - Product information"
    viewData["product"] = product

    return render(request, self.template_name, viewData)

...

class ProductListView(ListView):
  model = Product
  template_name = 'product_list.html'
  context_object_name = 'products'  # This will allow you to loop through 'products' in your template

  def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context['title'] = 'Products - Online Store'
    context['subtitle'] = 'List of products'
    return context
...
```

Template/view

- In *templates/products/show.html*, make the following changes in **bold**.

Modify Bold Code

```
{% extends 'pages/base.html' %}
{% block title %} {{title}} {% endblock %}
{% block header_title %} {{subtitle}} {% endblock %}

{% block content %}
<div class="card mb-3">
  <div class="row g-0">
```

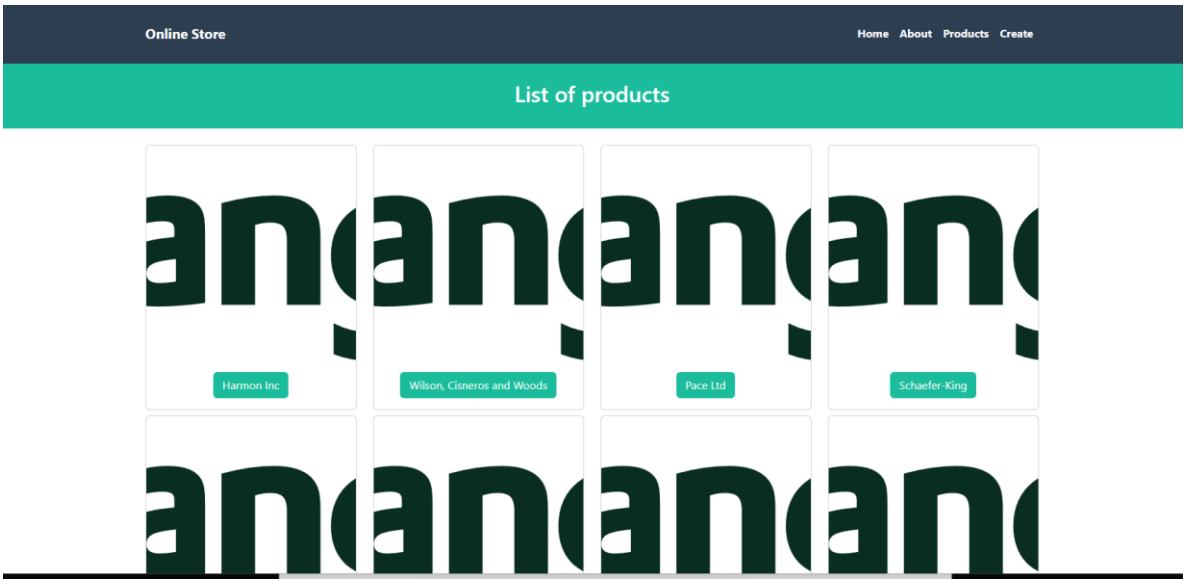
```
    <div class="col-md-4">
      <img src="https://static.djangoproject.com/img/logos/django-logo-positive.svg" class="img-fluid
rounded-start">
    </div>
    <div class="col-md-8">
      <div class="card-body">
        <h5 class="card-title">
          {{product.name}}
        </h5>
        <p class="card-text">{{product.description}}</p>
        {% if product.price > 2000 %}
            <p class="card-text" style="color: red;">{{product.price}}</p>
        {% else %}
            <p class="card-text">{{product.price}}</p>
        {% endif %}
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

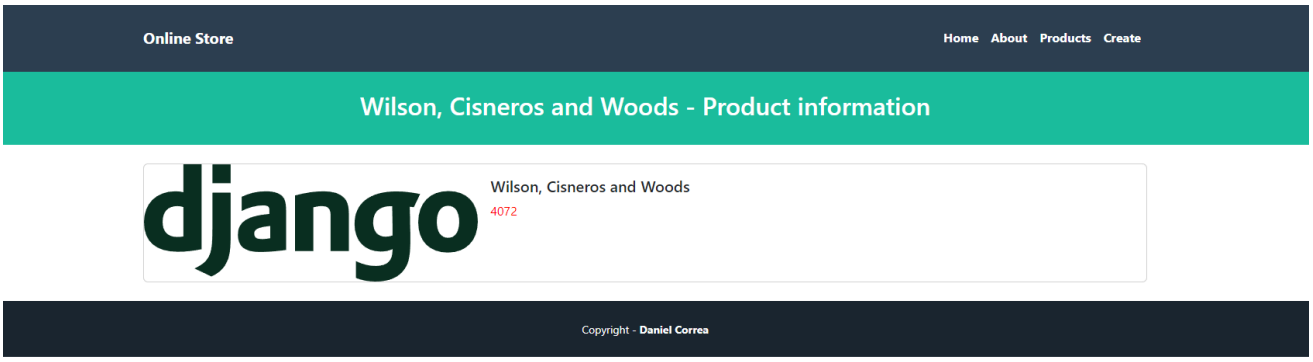
Execution

Execute in Terminal

```
python manage.py runserver
```

Go to the ("/products") route and you should see the application running with information retrieved from the database.


Online Store
Home About Products Create
List of products
Harmon Inc
Wilson, Cisneros and Woods
Pace Ltd
Schaefer-King


Online Store
Home About Products Create
Wilson, Cisneros and Woods - Product information
Wilson, Cisneros and Woods
4072
Copyright - Daniel Correa

D. A basic save product

View/Controller

- In *pages/views.py*, make the following changes in **bold**.

Modify Bold Code

```
...

class ProductForm(forms.ModelForm):
~~name = forms.CharField(required=True)~~
~~price = forms.FloatField(required=True)~~
class Meta:
    model = Product
    fields = ['name', 'price']

def clean_price(self):
    price = self.cleaned_data.get('price')
    if price is not None and price <= 0:
        raise ValidationError('Price must be greater than zero.')
    return price

class ProductCreateView(View):
    template_name = 'products/create.html'

    def get(self, request):
        form = ProductForm()
        viewData = {}
        viewData["title"] = "Create product"
        viewData["form"] = form
        return render(request, self.template_name, viewData)

    def post(self, request):
        form = ProductForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('product-created')
        else:
```

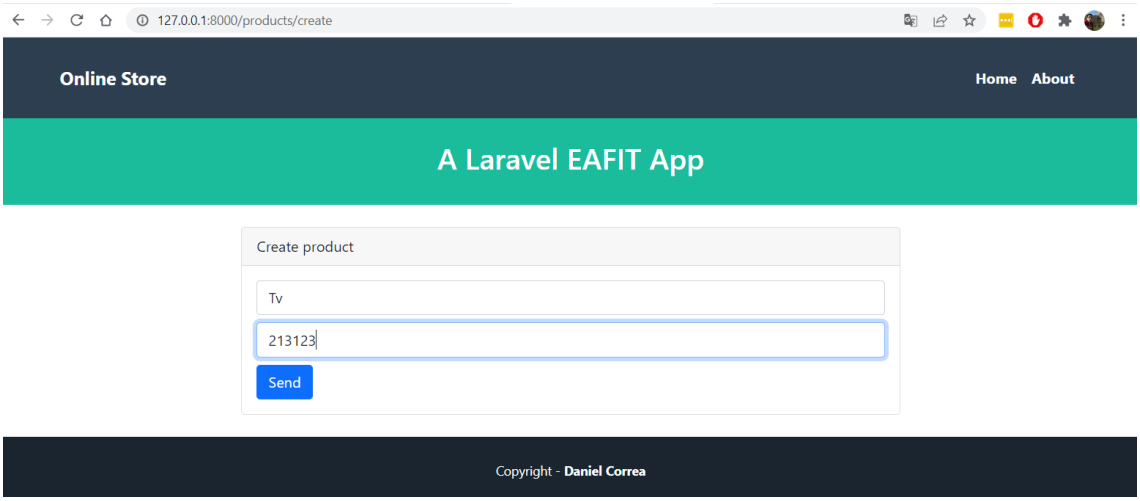
```
        viewData = {}
        viewData["title"] = "Create product"
        viewData["form"] = form
        return render(request, self.template_name, viewData)
```

Execution

Execute in Terminal

```
python manage.py runserver
```

Go to the ("/products/create") route and insert a new product. It should be added into the database.



SELECT * FROM pages_product Schema Query Editor Auto Reload Find Other Tools…

	id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT	name varchar(255) NOT NULL	price INTEGER NOT NULL	created_at datetime NOT NULL	updated_at datetime NOT NULL
1	1	Harmon Inc	4931	2023-08-14 13:11:3…	2023-08-14 13:11:3…
2	2	Wilson, Cisneros a…	4072	2023-08-14 13:11:3…	2023-08-14 13:11:3…
3	3	Pace Ltd	7520	2023-08-14 13:11:3…	2023-08-14 13:11:3…
4	4	Schaefer-King	3536	2023-08-14 13:11:3…	2023-08-14 13:11:3…
5	5	Brown, Hamilton an…	4671	2023-08-14 13:11:3…	2023-08-14 13:11:3…
6	6	Levy, Griffin and …	7760	2023-08-14 13:11:3…	2023-08-14 13:11:3…
7	7	Kelley, Martinez a…	5496	2023-08-14 13:11:3…	2023-08-14 13:11:3…
8	8	Le-Richard	1533	2023-08-14 13:11:3…	2023-08-14 13:11:3…
9	9	TV	500	2023-08-14 14:21:5…	2023-08-14 14:21:5…

E. A basic comment model and comment relationship

Model

- Go to *pages/models.py* and add the following code:

Add Entire Code

```
class Comment(models.Model):

    product = models.ForeignKey(Product, on_delete=models.CASCADE)

    description = models.TextField()
```

Executing the migrations

- To run the migrations, in the Terminal:

Execute in Terminal

```
python manage.py makemigrations
python manage.py migrate
```

Comments rows

- Open your project's *db.sqlite3* database and find the pages_comments table:

db.sqlite3

SELECT * FROM pages_product | Schema | Query Editor | Auto Reload | Find | Other Tools…

table
- auth_group
- auth_group_permissions
- auth_permission
- auth_user
- auth_user_groups
- auth_user_user_permissions
- django_admin_log
- django_content_type
- django_migrations
- django_session
- pages_comment
- pages_product

	id INTEGER PRIMAR AUTOINC		price INTEGER NOT NULL	created_at datetime NOT NULL	updated_at datetime NOT NULL
1	1		4931	2023-08-14 13:11:3…	2023-08-14 13:11:3…
2	2		4072	2023-08-14 13:11:3…	2023-08-14 13:11:3…
3	3		7520	2023-08-14 13:11:3…	2023-08-14 13:11:3…
4	4		3536	2023-08-14 13:11:3…	2023-08-14 13:11:3…
5	5		4671	2023-08-14 13:11:3…	2023-08-14 13:11:3…
6	6		7760	2023-08-14 13:11:3…	2023-08-14 13:11:3…
7	7		5496	2023-08-14 13:11:3…	2023-08-14 13:11:3…
8	8	Le-Richard	1533	2023-08-14 13:11:3…	2023-08-14 13:11:3…
9	9	TV	500	2023-08-14 14:21:5…	2023-08-14 14:21:5…

- Click on a new row add a comment with comment id, a description and link it to product id 1. Do this process three time and it should look like this in the pages_comment table:

db.sqlite3

SELECT * FROM pages_comment | Schema | Query Editor | Auto Reload | Find | Other Tools…

	id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT	description TEXT NOT NULL	product_id bigint NOT NULL REFERENCES "pages_product"("id")
1	1	Best comment	1
2	2	Bester Comment	1
3	3	Bestest comment	1

Product show view

- Go to *pages/templates/products/show.html* and make the following changes to the Template:

Modify Bold Code

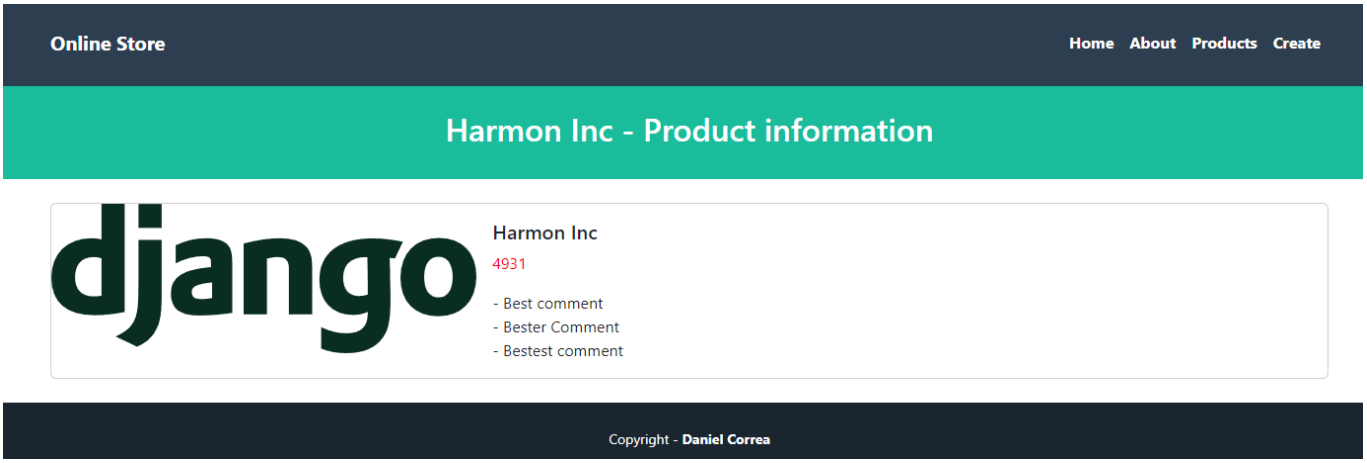
```
{% extends 'pages/base.html' %}
{% block title %} {{title}} {% endblock %}
{% block header_title %} {{subtitle}} {% endblock %}

{% block content %}
<div class="card mb-3">
  <div class="row g-0">
    <div class="col-md-4">
      <img src="https://static.djangoproject.com/img/logos/django-logo-positive.svg" class="img-
fluid rounded-start">
    </div>
    <div class="col-md-8">
      <div class="card-body">
        <h5 class="card-title">
          {{product.name}}
        </h5>
        {% if product.price > 2000 %}
            <p class="card-text" style="color: red;">{{product.price}}</p>
        {% else %}
            <p class="card-text">{{product.price}}</p>
        {% endif %}
        {% for comment in product.comment_set.all %}
        - {{ comment.description }}<br />
        {% endfor %}
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

Execution

Execute in Terminal

```
python manage.py runserver
```

Go to the ("/products/1") route. You will see the product with id 1 with its comments.



Congratulations, you have created an app which uses Django Models (including relationships), Fakers, Migrations, Seeds, and many more.

Bonus: try one of the Coding Standard Libraries in your code to clean it.