E-commerce Product Catalog using Django

Last Updated : 24 Sep, 2024

Django is an excellent choice for building a robust product catalog. Django, a high-level Python web framework, provides a solid foundation for creating feature-rich and scalable e-commerce websites. In this article, we will explore how to build an e-commerce product catalog using Django.

E-commerce Product Catalog using Django

To install Django follow these steps.

Starting the Project Folder

To start the project use this command

```
django-admin startproject ecommerce
cd ecommerce
```

To start the app use this command

```
python manage.py startapp catalog
```

Now add this app to the **'settings.py'**

Note: An e-commerce product catalog is a great Django project to work on. To master building complex web applications like this, the **Django Web Development Course** can guide you step-by-step.

Setting up the files

model.py: Here we have created a Product table with name, description, price, image, created_at, and updated_at field in the table.

Python

```python
from django.db import models

class Product(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
    image = models.ImageField(upload_to='products/')
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.name
```

admin.py: Here we are registering our table in the admin.

Python

```python
from django.contrib import admin
from .models import Product
admin.site.register(Product)
```

views.py: It includes view functions for displaying product lists and details, with routing and HTML templates. The Product model is assumed to represent products. Additional features like user authentication and shopping cart functionality need to be added for a complete e-commerce solution

Python

```python
from django.shortcuts import render
from .models import Product
def product_list(request):
    products = Product.objects.all()
    return render(request, 'myapp/index.html', {'products': products})
def product_detail(request, pk):
    product = Product.objects.get(pk=pk)
    return render(request, 'myapp/index2.html', {'product': product})
```

```python
from django.http import HttpResponse
def home(request):
    return HttpResponse('Hello, World!')
```

urls.py: Define the URL patterns in the urls.py file of the catalog app to map views to URLs.

Python

```python
# catalog/urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('/home', views.home, name='home'),
    path('', views.product_list, name='product_list'),
    path('<int:pk>/', views.product_detail, 
name='product_detail'),
]
```

Creating GUI for Project

These HTML templates use Django template language ({% ... %}) to render dynamic data from your Django models. In the product list template, it loops through the products queryset and displays each product's name, description, price, and an image. The product detail template shows detailed information about a single product and provides a link to go back to the product list.

index.html: This HTML file is used to Display all the products on the page.

HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Product Catalog</title>
    <style>
```

```
        /* Add CSS styles for flex container and items */
        .product-list {
            display: flex;
            flex-wrap: wrap; /* Allow items to wrap to the next row if necessary */
            justify-content: space-between; /* Space items evenly along the main axis */
            list-style: none; /* Remove list styles */
            padding: 0;
        }

        .product-item {
            flex: 1; /* Grow to fill available space evenly */
            max-width: 30%; /* Limit item width to avoid overcrowding */
            margin: 10px; /* Add spacing between items */
            border: 1px solid #ccc; /* Add a border for visual separation */
            padding: 10px;
            text-align: center;
        }

        /* Style the "Buy Now" button */
        .buy-now-button {
            display: block;
            margin-top: 10px;
            background-color: #007bff;
            color: #fff;
            text-decoration: none;
            padding: 5px 10px;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <h1>Product Catalog</h1>

    <ul class="product-list">
        {% for product in products %}
            <li class="product-item">
                <a href="{% url 'product_detail' product.pk %}">
```

```
                    <img src="{{ product.image.url }}"
alt="{{ product.name }}" width="100">
                </a>
                <h2>{{ product.name }}</h2>
                <p>{{ product.description }}</p>
                <p>Price: ${{ product.price }}</p>
                <a href="#" class="buy-now-button">Buy
Now</a>
            </li>
        {% endfor %}
    </ul>
</body>
</html>
```

index2.html

This HTML file is used to view the particular product on the page.

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{{ product.name }} - Product Detail</title>
</head>
<body>
    <h1>{{ product.name }} - Product Detail</h1>

    <div>
        <img src="{{ product.image.url }}" alt="{{
product.name }}" width="200">
    </div>
    <h2>{{ product.name }}</h2>
    <p>{{ product.description }}</p>
    <p>Price: ${{ product.price }}</p>

    <a href="{% url 'product_list' %}">Back to Product
List</a>
</body>
</html>
```

urls.py: Don't forget to add the necessary URL patterns for serving media files in your project's urls.py.

Python

```python
from django.contrib import admin
from django.urls import path, include
from django.conf.urls.static import static
from django.conf import settings

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('catalog.urls')),
] + static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

Setting Up Static and Media Files

Configure Django to serve static and media files during development. In your project's **settings.py** file, add the following paths:

```
# Static files (CSS, JavaScript, images)
STATIC_URL = '/static/'
STATICFILES_DIRS = [BASE_DIR / "static"]
# Media files (uploaded user files)
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / "media"
# Default primary key field type
# https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Deployement of the Project

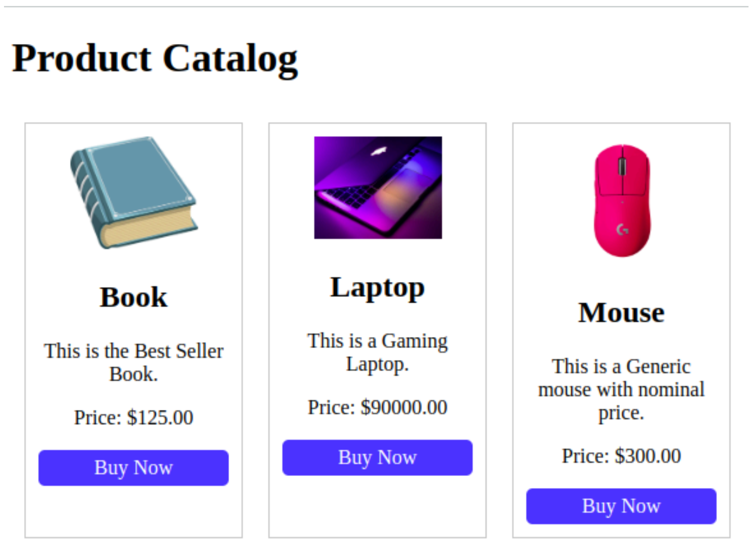
Run these commands to apply the migrations:

```
python3 manage.py makemigrations
python3 manage.py migrate
```

Run the server with the help of following command:

```
python3 manage.py runserver
```

Output



Product Catalog

Book

This is the Best Seller Book.

Price: $125.00

Buy Now

Laptop

This is a Gaming Laptop.

Price: $90000.00

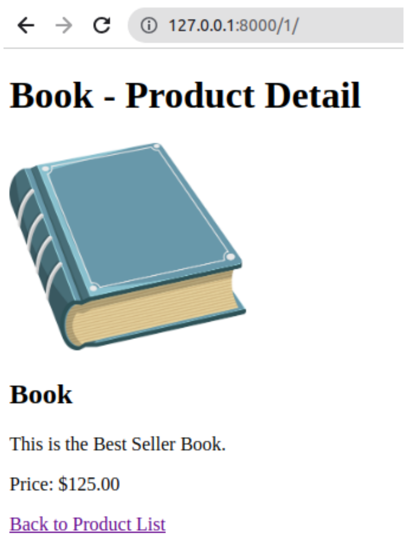
Buy Now

Mouse

This is a Generic mouse with nominal price.

Price: $300.00

Buy Now



127.0.0.1:8000/1/

Book - Product Detail

Book

This is the Best Seller Book.

Price: $125.00

Back to Product List

Are you ready to elevate your web development skills from foundational knowledge to advanced expertise? Explore our [Mastering Django Framework - Beginner to Advanced Course] on GeeksforGeeks, designed for aspiring developers and experienced programmers. This comprehensive course covers everything you need to know about Django, from the basics to advanced features. Gain practical experience through **hands-on projects** and real-world applications, mastering essential Django principles and techniques. Whether you're just starting or looking to refine your skills, this course will empower you to build sophisticated web applications efficiently. Ready to enhance your web development journey? Enroll now and unlock your potential with Django!

c chitr... 1

Previous Article **Next Article**

Bookstore Ecommerce App using MERN Stack

Similar Reads

E-commerce Website using Tailwind and React using Django

Our e-commerce website, "ECOM," aims to offer users a smooth online shopping experience. We use Tailwind CSS for styling, React for interactive user interface...

6 min read

E-commerce Website using Django

This project deals with developing a Virtual website 'E-commerce Website'. It provides the user with a list of the various products available for purchase in the...

11 min read

Django project - Creating a Basic E-commerce Website for Displaying...

Project Title - Basic Ecommerce Website using Django Django is a powerful framework based on python. Here we will see how to create a basic e-commerc...

3 min read

Adding Tags Using Django-Taggit in Django Project

Django-Taggit is a Django application which is used to add tags to blogs, articles etc. It makes very easy for us to make adding the tags functionality to our djang...

2 min read

How to customize Django forms using Django Widget Tweaks ?

Django forms are a great feature to create usable forms with just few lines of code. But Django doesn't easily let us edit the form for good designs. Here, we...

3 min read

Integrating Django with Reactjs using Django REST Framework

In this article, we will learn the process of communicating between the Django Backend and React js frontend using the Django REST Framework. For the sake...

15+ min read

E-commerce Website using MERN Stack

The project is an E-commerce website built using the MERN (MongoDB, Express.js, React, Node.js) stack. It provides a platform for users to view and...

7 min read

Fashion Store E-Commerce using MERN Stack

This E-commerce site project uses the MERN (MongoDB, Express.js, React.js, and Node.js) stack to deliver a fully functional online shopping experience. Users can...

7 min read

Create an E-commerce App using React-Native

An E-commerce app using react native is an application designed to facilitate online buying and selling of goods and services. These apps aim to provide a...

5 min read

Styling Django Forms with django-crispy-forms

Django by default doesn't provide any Django form styling method due to which it takes a lot of effort and precious time to beautifully style a form. django-crispy...

1 min read

Article Tags : Django Geeks Premier League Project Geeks Premier League 2023 +2 More

Corporate & Communications Address:- A-143, 9th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305) | Registered Address:- K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305

GET IT ON Google Play
Download on the App Store

Company
About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

Languages
Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

DSA
Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Data Science & ML
Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies
HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Python Tutorial
Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

Inteview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation
Puzzles

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved