



How to Add Cart in a Web Page using Django?

Last Updated : 24 Sep, 2024

Adding a shopping cart to a web page using Django is a crucial step when building an e-commerce website or any other application that involves online transactions. A shopping cart allows users to collect and manage items they want to purchase before proceeding to checkout. In this tutorial, we'll walk you through the process of creating a basic shopping cart using Django, including all the necessary files and HTML templates.

Steps Add Cart in a Web Page using Django

Setup and Installation

To install Django follow these [steps](#).

To start the project use this command

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

To start the app use this command.

```
python manage.py startapp cart
```

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

```

32
33     INSTALLED_APPS = [
34         'django.contrib.admin',
35         'django.contrib.auth',
36         'django.contrib.contenttypes',
37         'django.contrib.sessions',
38         'django.contrib.messages',
39         'django.contrib.staticfiles',
40         'cart'
41     ]
42

```

model.py: Here we have created a Product table with name, description, and price, image field in the table. Whereas, The CartItem model represents items added to the shopping cart by users.

Python



```

1  from django.db import models
2  from django.contrib.auth.models import User
3  class Product(models.Model):
4      name = models.CharField(max_length=100)
5      description = models.TextField(null=True)
6      price = models.DecimalField(max_digits=10,
    decimal_places=2)

```

Django Views Model Template Forms Jinja Python SQLite Flask Json Postman Interview Ques

```

9      def __str__(self):
10         return self.name
11
12     class CartItem(models.Model):
13         product = models.ForeignKey(Product,
14             on_delete=models.CASCADE)
15         quantity = models.PositiveIntegerField(default=0)
16         user = models.ForeignKey(User,
17             on_delete=models.CASCADE)
18         date_added = models.DateTimeField(auto_now_add=True)
19
20     def __str__(self):
21         return f'{self.quantity} x {self.product.name}'

```

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

Python



```
1 from django.contrib import admin
2 from .models import Product, CartItem
3
4 admin.site.register(Product)
5 admin.site.register(CartItem)
```

views.py: The **cart/views.py** code contains views that handle various aspects of the shopping cart functionality in a Django application. Here's a brief explanation of each view:

- **product_list(request):** This view fetches a list of products from the database and renders a template to display them. Each product is shown with its name, price, and an "Add to Cart" button.
- **view_cart(request):** This view displays the contents of the shopping cart. It retrieves the cart items associated with the current user, calculates the total price of the items in the cart, and then renders a template to display the cart items along with the total price.
- **add_to_cart(request, product_id):** When a user clicks the "Add to Cart" button on a product, this view is triggered. It adds the selected product to the user's cart. If the product is already in the cart, it increments the quantity. It then redirects the user to the cart view to display the updated cart contents.
- **remove_from_cart(request, item_id):** This view handles the removal of an item from the cart. It takes the item's ID as a parameter, retrieves the corresponding cart item, and deletes it from the database. After removal, it redirects the user to the cart view to reflect the updated cart.

Python



```
1 from django.shortcuts import render, redirect
2 from .models import Product, CartItem
3
4 def product_list(request):
```

```
        products = Product.objects.all()
6         return render(request, 'myapp/index.html', {'products':
products})
7
8     def view_cart(request):
9         cart_items = CartItem.objects.filter(user=request.user)
10        total_price = sum(item.product.price * item.quantity for
item in cart_items)
11        return render(request, 'myapp/cart.html', {'cart_items':
cart_items, 'total_price': total_price})
12
13    def add_to_cart(request, product_id):
14        product = Product.objects.get(id=product_id)
15        cart_item, created =
CartItem.objects.get_or_create(product=product,
16
user=request.user)
17        cart_item.quantity += 1
18        cart_item.save()
19        return redirect('cart:view_cart')
20
21    def remove_from_cart(request, item_id):
22        cart_item = CartItem.objects.get(id=item_id)
23        cart_item.delete()
24        return redirect('cart:view_cart')
25
26
27    def home(request):
28        return HttpResponse('Hello, World!')
```

Adding a cart to a Django web page is a crucial skill for e-commerce apps. If you're looking to build more complex applications, the [Django Web Development Course](#) will guide you through advanced concepts.

Creating GUI

index.html: It is used to display a list of products with "Add to Cart" buttons.

HTML



```
1  <!DOCTYPE html>
2  <html lang="en">
```

```
<head>
4     <meta charset="UTF-8">
5     <title>Product Catalog</title>
6     <style>
7         /* Add CSS styles for flex container and items */
8         .product-list {
9             display: flex;
10            flex-wrap: wrap; /* Allow items to wrap to the
next row if necessary */
11            justify-content: space-between; /* Space items
evenly along the main axis */
12            list-style: none; /* Remove list styles */
13            padding: 0;
14        }
15
16        .product-item {
17            flex: 1; /* Grow to fill available space evenly
*/
18            max-width: 30%; /* Limit item width to avoid
overcrowding */
19            margin: 10px; /* Add spacing between items */
20            border: 1px solid #ccc; /* Add a border for
visual separation */
21            padding: 10px;
22            text-align: center;
23        }
24
25        /* Style the "Buy Now" button */
26        .buy-now-button {
27            display: block;
28            margin-top: 10px;
29            background-color: #007bff;
30            color: #fff;
31            text-decoration: none;
32            padding: 5px 10px;
33            border-radius: 5px;
34        }
35    </style>
36 </head>
37 <body>
38     <h1>Product Catalog</h1>
39
40     <ul class="product-list">
41         {% for product in products %}
```

```

    <li class="product-item">
43         
44         <h2>{{ product.name }}</h2>
45         <p>{{ product.description }}</p>
46         <p>Price: ${{ product.price }}</p>
47         <a href="#" class="buy-now-button">Buy
Now</a>
48         <a class="buy-now-button" href="{% url
'cart:add_to_cart' product.id %}">Add to Cart</a>
49
50     </li>
51     {% endfor %}
52 </ul>
53 </body>
54 </html>

```

cart.html: Display the shopping cart, allow item removal, and calculate the total price.

HTML



```

1  <!-- cart/cart.html -->
2
3  <html lang="en">
4
5  <head>
6      <meta charset="UTF-8">
7      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
8      <title>Document</title>
9      <style>
10         /* Add CSS styles for flex container and items */
11         .product-list {
12             display: flex;
13             flex-wrap: wrap; /* Allow items to wrap to the
next row if necessary */
14             justify-content: space-between; /* Space items
evenly along the main axis */
15             list-style: none; /* Remove list styles */
16             padding: 0;
17         }
18

```

```
        .product-item {
20
21            flex: 1; /* Grow to fill available space evenly
22            */
23            /* Limit item width to avoid overcrowding */
24            margin: 10px; /* Add spacing between items */
25            border: 1px solid #ccc; /* Add a border for
visual separation */
26            padding: 10px;
27            text-align: center;
28        }
29
30        /* Style the "Buy Now" button */
31        .buy-now-button {
32            display: block;
33            margin-top: 10px;
34            background-color: #007bff;
35            color: #fff;
36            text-decoration: none;
37            padding: 5px 10px;
38            border-radius: 5px;
39        }
40    </style>
41</head>
42<body>
43
44    <h1>Your Shopping Cart</h1>
45
46    <div class="product-list">
47
48
49
50        {% for item in cart_items %}
51        <div class="product-item">
52            <p>{{ item.product.name }} ({{ item.quantity }})</p>
53            <p>Price: ${{ item.product.price }}</p>
54            <a href="{% url 'cart:remove_from_cart' item.id
55            %}">Remove</a>
56        </div>
57        {% empty %}
58        <p>Your cart is empty.</p>
59        {% endfor %}
```

```
        </div>
61
62     <p>Total Price: ${ total_price }</p>
63
64     <a href="{% url 'cart:product_list' %}">Continue
Shopping</a>
65
66 </body>
67
68 </html>
```

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

Python



```
1 from django.urls import path
2 from . import views
3
4 app_name = 'cart'
5
6 urlpatterns = [
7     path('', views.product_list, name='product_list'),
8     path('/home', views.home, name='home'),
9     path('cart/', views.view_cart, name='view_cart'),
10    path('add/<int:product_id>/', views.add_to_cart,
name='add_to_cart'),
11    path('remove/<int:item_id>/', views.remove_from_cart,
name='remove_from_cart'),
12 ]
```

urls.py: Add the necessary URL patterns for serving media files in your project's **urls.py**.

Python



```
1 from django.contrib import admin
2 from django.urls import path, include
3 from django.conf.urls.static import static
4 from django.conf import settings
5
```



```
urlpatterns = [  
7     path('admin/', admin.site.urls),  
8     path('', include('cart.urls')),  
9 ] + 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:

```
# 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"
```

Deployment of the Project

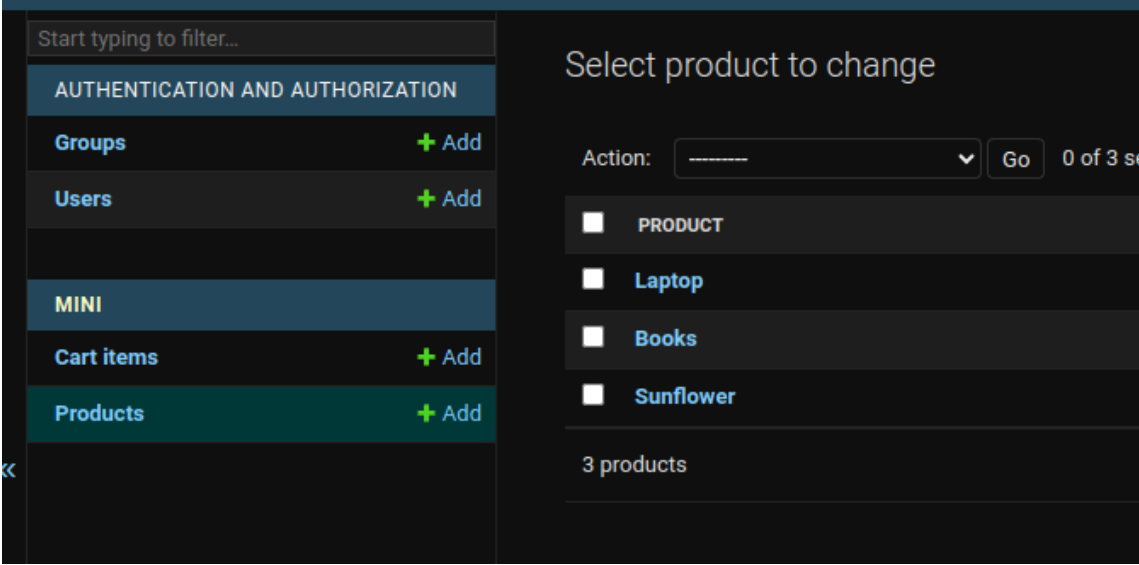
Run these commands to apply the migrations:

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

Run the server with the help of following command:


```
python manage.py runserver
```

Now, Go to the <http://127.0.0.1:8000/admin/> and add the Images, name and its description.



Output

Product Catalog




Sunflower

Helianthus is a genus comprising about 70 species of annual and perennial flowering plants in the daisy family Asteraceae commonly known as sunflowers.

Price: \$29.00

Buy Now

Add to Cart




Books

A book is a medium for recording information in the form of writing or images, typically composed of many pages bound together and protected by a cover.

Price: \$300.00

Buy Now

Add to Cart



Laptop

A laptop computer or notebook computer, also known as a laptop or notebook for short, is a small, portable

Price: \$80000.00

Buy Now

Add to Cart

Your Shopping Cart

Sunflower (3)

Price: \$29.00

Remove

Books (1)

Price: \$300.00

Remove

Laptop (1)

Price: \$80000.00

Remove

Total Price: \$80387.00

[Continue Shopping](#)

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!

A amit...



Previous Article

How to use Regex Validator in Django?

Next Article

Similar Reads

Shopping Cart Project Using C Language

Online Shopping applications are one of the favorite applications out of all online applications. It is a very much used application in everyone's life. So, Let's try to...

11 min read

Shopping Cart app using React

In this article, we will create an Interactive and Responsive Shopping Cart Project Application using the famous JavaScript FrontEnd library ReactJS. We have...

9 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 django...

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

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

How to Fix Django "ImportError: Cannot Import Name 'six' from..."

After Django 3.0, django.utils.six was removed. The "ImportError: Cannot Import Name 'six' from 'django.utils'" error occurs because of the dependency issue. Thi...

3 min read

Django Form submission without Page Reload

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it...

2 min read

Django - Creating a 404 Error Page

The 404 error is raised when the URL called upon doesn't exist or has not been defined yet. This is commonly referred to as Page does not exist error. To handle...

4 min read

Python Django Handling Custom Error Page

To handle error reporting in Django, you can utilize Django's built-in form validation mechanisms and Django's error handling capabilities. In this article, I'll...

4 min read

Article Tags :

Django

Geeks Premier League

Project

Geeks Premier League 2023

+1 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



Company

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

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

Web Technologies

Languages

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

Data Science & ML

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

Python Tutorial

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

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

System Design

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

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

Interview Preparation

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

GeeksforGeeks Videos

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

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved