Contents

This is the 102 videos playlist of CodewithHarry.

Admin credentials: Username: Tanmaya, Password: tancannon@2003

...after the 1st tutorial of Django extra documentation or my journal(what I have learned is written here)

Adding media directory

When any image is added from the admin panel it goes to a separate directory named "media"; i.e. we are saving all the images in a dedicated place within a folder inside named as the app name (e.g media/shop/images/dustbin.jpg).

-In the "settings.py" of the main project directory(here "mac") add;

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

-Then import these in the main project directory in urls.py(here "mac");

```
from django.conf import settings
from django.conf.urls.static import static
```

and add the url as below(the thing starting with + im talking about);

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

Jinga templating (reusing templates)

to copy the code of a html file to another(or in otherwors to resuse the code) we

use this jinga templating.

say in 'about.html'

{%extends 'shop/base.html' %} #this means we're reusing the contents of base.html in about.html.

also we can add variable blocks in the parent file (here base.html)

e.g;

written in base file (here base.html):

{% block title%}{% endblock %}

written in the reusing file (e.g about.html)

{% block title%} Tancannon Industries {% endblock %}

This replaces the 'Tancannon Industries' at {% block title%}{% endblock %} when reusing and can only be seen in the reusing file i.e about.html.

Adding loop to the carousal or sliders

Here im fetching the products from the database, so i need to iterate over the product objects fetched from the database.

{% for i in range %}

<li data-target="#demo" data-slide-to="{{i}}" >

{% endfor %}

This is used to to make slides whose numbers is equal to "range".

See in shop/views.py:

products = Products.objects.all() //fetching the objects from the databases.

```
    print(products)
    n = len(products)
    nSlides = n//4 + ceil((n/4)-(n//4)) //formula to get no. of slides(in carousal) needed to show images.
    params = {
        'no_of_slides': nSlides,
```

```
        'range': range(nSlides),
        'product': products
    }
```

Then;

```
<div class="container carousel-inner no-padding">
        <div class="carousel-item active">
          <div class="col-xs-3 col-sm-3 col-md-3">
            <div class="card" style="width: 18rem;">
              <!--<img src='{% static "shop/test.jpg" %}' class="card-img-top" alt="..."> -->
              <img src='/media/{{product.0.image}}' class="card-img-top" alt="...">
              <div class="card-body">
                  <h5 class="card-title">{{product.0.product_name}}</h5>
                  <p class="card-text">{{product.0.desc}}</p>
                  <a href="#" class="btn btn-primary">Add to Cart</a>
              </div>
            </div>
        </div>
```

- *The 1st product is hardcoded.*

The rest images are loaded into the slides like this using loop;

```
        {% for i in product|slice:"1:"%}
        <div class="col-xs-3 col-sm-3 col-md-3">
          <div class="card" style="width: 18rem;">
            <img src='/media/{{i.image}}' class="card-img-top" alt="...">
            <div class="card-body">
              <h5 class="card-title">{{i.product_name}}</h5>
              <p class="card-text">{{i.desc}}</p>
              <a href="#" class="btn btn-primary">Add To Cart</a>
            </div>
          </div>
```

```
        </div>
        {% if forloop.counter|divisibleby:3 and forloop.counter > 0 and
not forloop.last %}
      </div>
      <div class="carousel-item">
        {% endif %}
          {% endfor %}
     </div>
```

Adding cart button

Here we'll be able to see the number of different categories of items we need to add to the cart.

In "shop/basic.html", we have added a "popover button"

```
<!--manually added popover bottom -->
              <button type="button" class="btn btn-secondary mx-2"
id="popcart" data-container="body" data-toggle="popover" data-
placement="bottom" data-html="true" data-content="Vivamus
              sagittis lacus vel augue laoreet rutrum faucibus.">
                Cart(<span id="cart">0</span>)
              </button>
```

"gets popups" in the bottom of the cart button when is pressed.

In "index.html" in the script tag;

```
if(localStorage.getItem('cart')==null){ //we're saving the cart value in
localstoarge of the website.
  var cart = {};
}
else{
  cart = JSON.parse(localStorage.getItem('cart'));

  document.getElementById('cart').innerHTML = Object.keys(cart).length;
}
```

document.getElementById('cart').innerHTML = Object.keys(cart).length; // This is used to update the element the text or number beside the cart in the cart button text on the nav bar.

The popover is used as below within the script tag in “shop/index.html”.

```
$('#popcart').popover();
document.getElementById("popcart").setAttribute('data-content','<h5>Cart for your shopping list</h5>');
```

Creating the product page

```
<a href="productview/{{i.id}}"<button id="pr{{i.id}}" class="btn btn-primary cart">Quick view</button></a>
```

We create a “Quick view” button. It is linked to "productview/{{i.id}}".

This is link is handled in urls.py as (in urlpatterns):

```
path("productview/<int:myid>",views.productView, name="productView"),
```

Then as usual it goes to views.py:

```
def productView(requests,myid):
    product = Products.objects.filter(id=myid)
    print(product)
    params = {'product':product[0]}
    return render(requests,'shop/prodView.html',params) #product is a list of one element
    # return HttpResponse('productview page')
```

The “i.id” i.e. “product_id” is send through the link from index.html -> urls.py -> views.py

So, we fetch the product information using the id and pass it to show in the prodView.html for a specific product.

In prodView.html;

```
{% extends 'shop/basic.html' %}

{% block title%} {{product.product_name}} - My Awesome Cart{% endblock %}
{% block body %}
<div class="container my-4">
    <div class="row">
```

```
    <div class="col-md-4">
    <div class="row">
        <img src="/media/{{product.image}}" width="233px" height="385px">
    </div>
        <div class="row my-2">
            <button class="btn btn-primary mx-3">Buy Now</button>
            <button class="btn btn-primary">Add To Cart</button>
        </div>

    </div>

    <div class="col-md-8">
        <h5>{{product.product_name}}</h5>
        <p><b>Rs.{{product.price}} </b></p>
        <p>{{product.desc}}</p>


    </div>
    </div>


</div>



{% endblock %}



{% block js %}
<script>
console.log('working');
if(localStorage.getItem('cart') == null){
var cart = {};
}
else
{
cart = JSON.parse(localStorage.getItem('cart'));
document.getElementById('cart').innerHTML = Object.keys(cart).length;
}
$('.cart').click(function(){
console.log('clicked');
var idstr = this.id.toString();
console.log(idstr);
if (cart[idstr] !=undefined){
```

```
cart[idstr] = cart[idstr] + 1;
}
else
{
cart[idstr] = 1;
}
console.log(cart);
localStorage.setItem('cart', JSON.stringify(cart));
document.getElementById('cart').innerHTML = Object.keys(cart).length;
});
$('#popcart').popover();
document.getElementById("popcart").setAttribute('data-content', '<h5>Cart
for your items in my shopping cart</h5>');
</script>
{% endblock %}
```

Above the script tag is the showing of the product necessary attributes.

And in the script tag we have just copy pasted the one in index.html, so as to update the cart button on the navbar here also.

Making the contact page work

shop/contact.html:

```
{% extends 'shop/basic.html' %}
{% block title %} Contact Us {% endblock %}
{% block body %}
    <div class="container">
        <h3>Contact Us</h3>
        <form action="/shop/contact/" method="post">
            {% csrf_token %}
            <div class="form-group">
                <label for="name">Name</label>
                <input type="text" class="form-control" id="name" name="name"
placeholder="Enter Your Name">
            </div>
            <div class="form-group">
                <label for="name">Email</label>
                <input type="email" class="form-control" id="email" name="email"
placeholder="Enter Your Email">
            </div>
            <div class="form-group">
                <label for="name">Phone</label>
                <input type="tel" class="form-control" id="phone" name="phone"
placeholder="Enter Your Phone Number">
```

```
            </div>
            {% comment %} <div class="form-group"> {% endcomment %}
            <div class="form-group">
                <label for="desc">How May We Help You ?</label>
                <textarea class="form-control" id="desc" name="desc"
rows="3"></textarea>
            </div>
            <button type="submit" class="btn btn-success">Sumbit</button>
        </form>
{% endblock %}
```

Explanation:

```
<form action="/shop/contact/" method="post">
```

“action” specifies where to send the form-data when a form is submitted. This attribute overrides the form's action attribute. The form action attribute is only used for inputs/buttons with type="submit".

And set method=”post” i.e to post the form.

The keyword “name” is used to fetch it when the form is posted.

In “views.py” we are fetching the posted data using the values of keyword “name” in input fields.

In views.py:

```
from .models import Contact #we import the class Contact to use it
def contact(requests):
    # return HttpResponse('contact page')
    if (requests.method=="POST"):
        print(requests)
        #using the values of keyword “name” in “contact.html” to store data.
        name=requests.POST.get("name")
        email=requests.POST.get("email")
        phone=requests.POST.get("phone")
        desc=requests.POST.get("desc")
        print(name,email,phone,desc)
        #created a instance of class Contact named fetched_contact
        fetched_contact = Contact(name=name, email=email, phone=phone, desc=desc)
        fetched_contact.save() #to save the fetched data in the database.
    return render(requests,'shop/contact.html')
```

Explanation:

Then to save it on the database we use a class made by us in “models.py” i.e “Class Contact”.

See the above use of instance (a variable named “fetched_contacts”).

In models.py:

```
class Contact(models.Model):
    msg_id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=50)
    email = models.CharField(max_length=70,default="")
    phone = models.CharField(max_length=70, default="")
    desc = models.CharField(max_length=500, default="")

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

Explanation:

This is the class we have created in “shop/models.py”.

The model or class contact need to be registered in “shop/admin.py” before using as below.

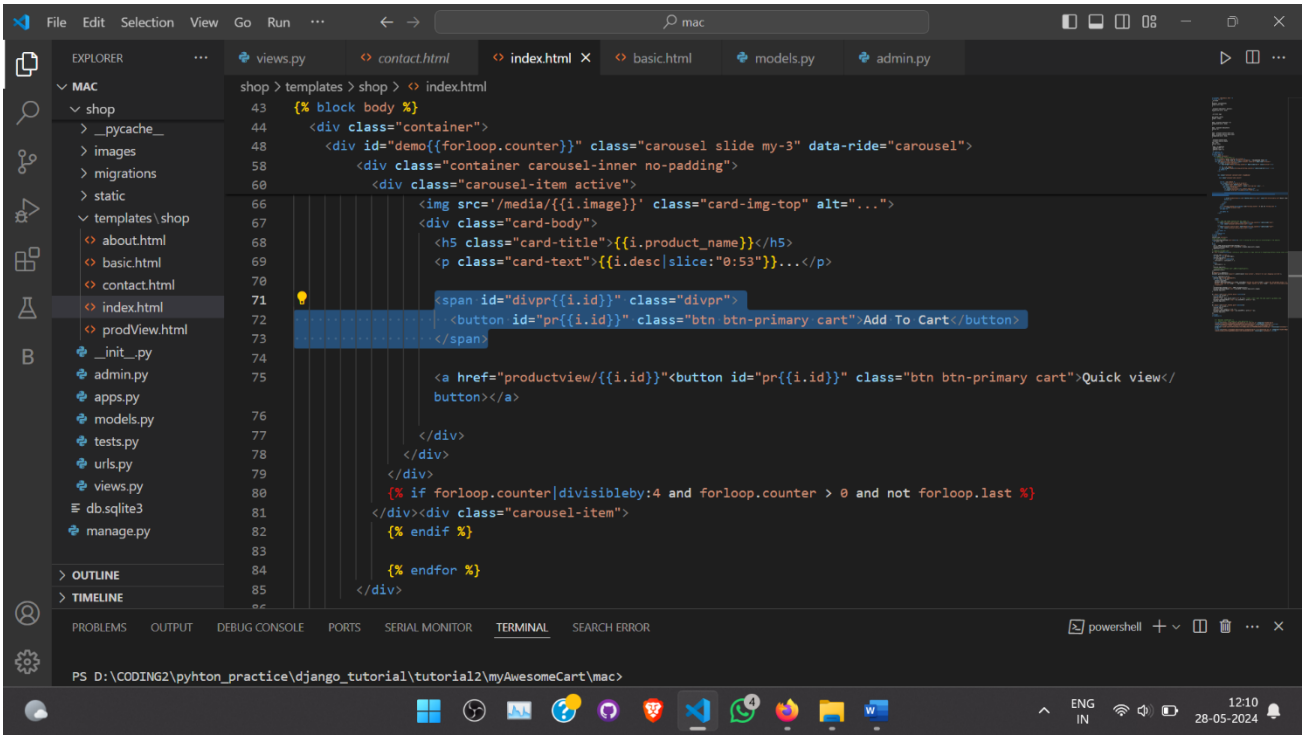
In admin.py:

```
from .models import Products, Contact
admin.site.register(Products)
admin.site.register(Contact) #this is the registration of class Contact.
```

Creating Add and Remove Buttons For Items In Cart

In this tutorial, we will create Add (+) and Remove (-) buttons for the cart of our E-commerce website. First of all, we need to make some changes to the index.html file. So, open the file and make the following changes:

1. Notice the highlighted area of the code in the image given below; we had included a tag before the buttons that we created in one of our previous tutorials. In the future, we will be using this element to target the buttons.



After doing this type the below code in the javascript section of the index.html file:

```
function updateCart(cart){
  console.log("inside updateCart");
  for (var item in cart){
    console.log(item);
    document.getElementById('div'+item).innerHTML="<button
id='minus"+item+"' class='btn btn-primary minus'>-</button> <span
id='val"+item+"'> "+cart[item]+" </span> <button id='plus"+item+"'
class='btn btn-primary plus'>+</button>";
  }
  localStorage.setItem('cart', JSON.stringify(cart));
  document.getElementById('cart').innerHTML = Object.keys(cart).length;
  console.log(cart);
}
```

In the above code, we have created a function called updateCart(), which performs the function of showing (+) and (-) button after clicking the Add To Cart button. We are using a for loop to iterate over the items of the cart. After the creation of updateCart function, we need to call this function whenever the user clicks on the "Add To Cart" button. So, type the below code inside the click function that we created earlier.

```
$('.cart').click(function(){ //setting a event listner to tags (here
we're targetting buttons) having class cart

  console.log('clicked');
```

```
  var idstr = this.id.toString();
  console.log(idstr);
  if (cart[idstr] != undefined){
    cart[idstr] = cart[idstr] + 1;
  }
  else{
    cart[idstr] = 1;
  }
  console.log(cart);
  //localStorage.setItem('cart',JSON.stringify(cart));
  updateCart(cart);
});
```

Alright! We are all set. Restart the development server and click on the "Add To Cart" button. You will notice that on click the button, (+) and (-) buttons are displayed. Now, click on (+) and (-) buttons, you will notice that nothing happens. This is because we have not written any code for incrementing and decrementing the number of products. So, type the below code in the index.html file :

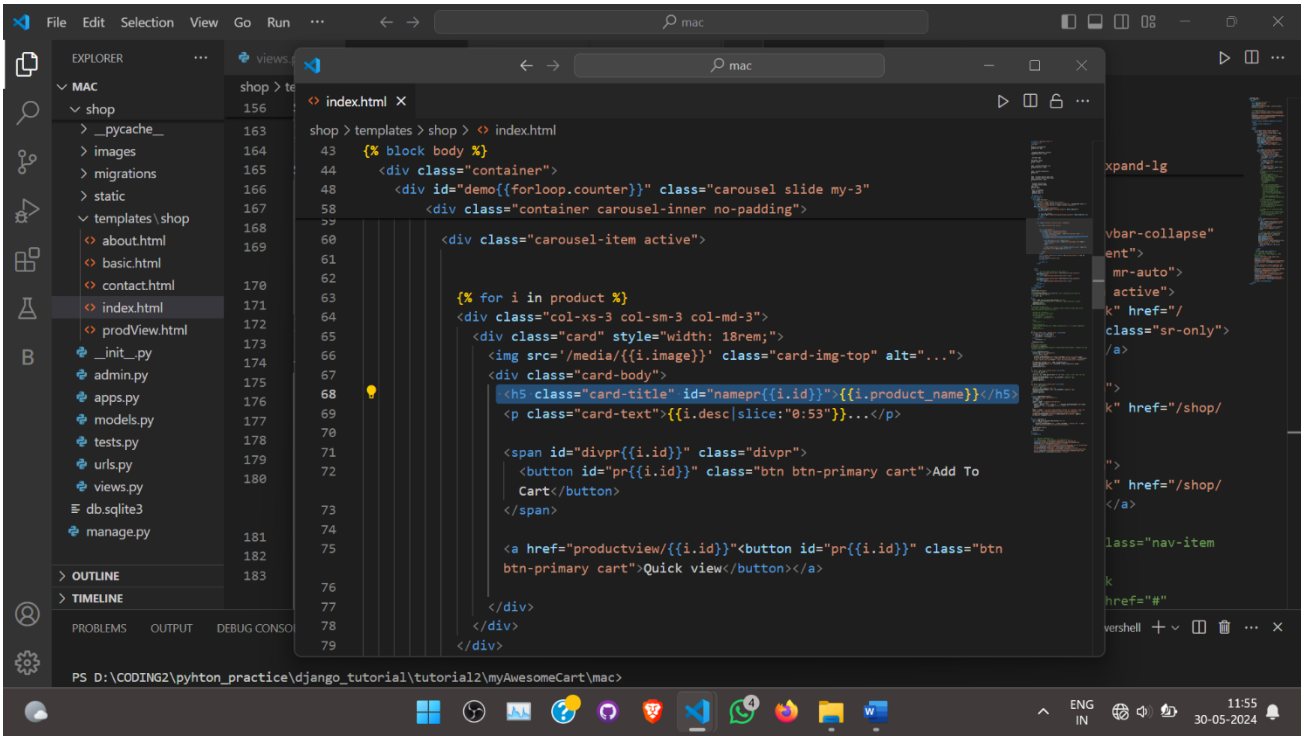
```
$('.divpr').on("click","button.minus",function(){
  a = this.id.slice(7,);
  console.log(a);
  cart['pr'+ a] = Math.max(0,(cart['pr'+ a] -1)); //cause i don't want
the item count to go below zero.
  document.getElementById('valpr'+ a).innerHTML = cart['pr'+ a];
  console.log(cart);
});

$('.divpr').on("click","button.plus",function(){
  a = this.id.slice(6,);
  console.log(a);
  cart['pr'+ a] = cart['pr'+ a] + 1;
  document.getElementById('valpr'+ a).innerHTML = cart['pr'+ a];
  console.log(cart);
});
```

In the above code, we've used jquery to select the element with div.pr class. Then, we've used the slice operation to get the id of the elements and in the end, we are changing the inner HTML. Reload the server, and if you have followed the steps correctly, then the (+) and (-) buttons will work correctly. So that's how we can easily create an add and remove button for the cart of our E-commerce website.

Showing contents in the cart in navbar

In “shop/basis.html”:

We have added a id = ”namepr{{i.id}}” to the highlighted line, so that it can be accessed and shown in the cart contents in the navbar popover.



In “shop/index.html”:

We have written this to access the select and show the title of the products who id is put in the “cart” in the popover in the navbar.

```
function updatePopover(cart){
  console.log("inside updatePopover");
  //we’re making the html into string to render it in the popover
  var popStr = "";
  var count = 1;
  popStr = popStr + "<b>Your Cart</b><br>";
  for (var item in cart){
    popStr = popStr + "<h>"+ count + ". " +
document.getElementById('name'+item).innerHTML+ " Qty:"+ cart[item] +"</h>" +
"<br>";
    count++;
  }
```

```
  popStr = popStr + "<a href='/shop/checkout/'<button id ='checkout' class='btn btn-primary my-1'>Checkout</button></a><button onclick='clearCart()' id ='clearCart' class='btn btn-primary my-1 mx-1'>Clear Cart</button>";
  document.getElementById('popcart').setAttribute('data-content', popStr);
  $('#popcart').popover('show'); }
```

Explanation:

`var popStr = "";` // we're making the html into string to render it in the popover.

Here we are sending it :

```
  document.getElementById('popcart').setAttribute('data-content', popStr);
```

`$('#popcart').popover('show');` //This just opens the popover automatically.

`document.getElementById('name'+item).innerHTML+` : this thing is used to access the heading of those item whose id is in the "cart" variable that we have made.

"item" is 'pr1' or 'pr2' ..etc. => namepr1 (i.e the id of the heading tag holding the title of product id 1 in the database. In "shop/index.html" id = "namepr{{i.id}}" that we have written above is used for accessing the content.

Adding Checkout and ClearCare Buttons in popover cart

This is we have added at last the Checkout and ClearCart Buttons in the

"updatePopover(cart)".

```
popStr = popStr + "<a href='/shop/checkout/'<button id ='checkout' class='btn btn-primary my-1'>Checkout</button></a><button onclick='clearCart()' id ='clearCart' class='btn btn-primary my-1 mx-1'>Clear Cart</button>";
```

The function declaration of ClearCart() is as follows, Checkout button

direct to checkout page shall be made functional later.

```
function clearCart() {
  //get the cart from the localstorage of the web braowser.
  cart = JSON.parse(localStorage.getItem('cart'));
```

```
  //dynamically take the “Add to cart” button to its original form (i.e
when it was not added to the “cart”)
  for (var item in cart) {
      document.getElementById('div' + item).innerHTML = '<button id="' +
item + '" class="btn btn-primary cart">Add To Cart</button>'
  }
//Clear the localstorage of the browser.
  localStorage.clear();
//make cart variable empty again.
  cart = {};
  updateCart(cart);
}
```