Blogpost

**Command#**

django-admin (lists commands you can do)

**TO create App:**

python manage.py startapp blog

**In Views:**

from django.http import HttpResponse

to get http response

create simple function as follow:

def home(request):

    return HttpResponse('<h1>Blog Home</h1>')

You created a function in the app aka “Blog” view file where you want your users to land.

You will need to create URL.py file within your Django app

In the Url following copy the following code.

from django.urls import path

from . import views

urlpatterns = [

    path('', views.home,name='blog-home'),

]

This links your urls.py in your app(“blog”) to your view file in the blog (note the second line)

Now we must link our main app url to the above app. You must import the function “include” to include your url file to your project file.

from django.contrib import admin

from django.urls import path, include

Update the url pattern

urlpatterns = [

    path('admin/', admin.site.urls),

    path('blog/', include('blog.urls')),

]

How the pathing works?

Request is sent to your projectapp url pattern file, let say you entered /blog, url pattern in your project will see that any requests to blog/ should be sent to blog.urls file which is a file call “urls.py” in our app “blog”. Now our url app file can send the users to any direction that we want to. So it user could do the following:

1. [www.yoursite/blog/about](http://www.yoursite/blog/about)
2. [www.yoursite/blog/finance](http://www.yoursite/blog/finance)
3. [www.yoursite/blog/year](http://www.yoursite/blog/year)

Our app url file has the following patterns

urlpatterns = [

    path('', views.home,name='blog-home'),

    path('about/', views.about,name='blog-about'),

]

Therefore only request# 1 is valid. Other request will result in 404 error as those webages do not exists.

**How to use templates in Django**

1. Create folder call template in your app directory
2. Create folder name using your app in our case it will be “blog”, all the templates for blog will be stored here.
3. Create html pages (home.html, about.html)
4. Add the app “blog” to installed app in project settings fig 1 below. Make sure you copy the function aka app name in your blog apps.py file correctly to installed app list
5. Import the function “render” if not already there. Use that to render the template. Change the return statements to render(request,’appname/name of html’,

Passing data into html

Create a dictionary in the file views.py with the posts. Then create another dictionary inside home function and assign the key of the dictionary to dictionary created outside function with posts. Passed the new dictionary into render function.

<body>

    {% for post in posts %}

    <h1>{{post.title}}</h1>

    <p>By {{post.author}} on {{post.date\_posted}}</p>

    <p>{{post.content}}</p>

    {% endfor %}

</body>

{% for post in posts %} is a for loop, post can be any variable, and “posts” refer to the dictionary we created in the view. We can access the value by post.key.

If statement within html as follow:

{% if title %}

    <title>Django Blog-{{title}}</title>

    {% else %}

    <title>Django Blog</title>

    {% endif %}

Make sure spaces are correct, they are very stingy about that.

**Template inheritance**

In order to save time, we can create a base.html file that can be served as template for other html.

Any different code that might be needed in other html pages can be written within

{% block content %} {% endblock %} . This goes in the base.html

**Django Database**

**We create classes in the models.py, each class is a table in the database.**

from django.db import models

from django.utils import timezone

# Create your models here.

class Post(models.Model):

    title = models.CharField(max\_length=100)

    content = models.TextField() #unrestricted text

    #date\_posted = models.DateTimeField(auto\_now\_add=True) auto\_now adds date when it was created but you cant change it later

    date\_posted = models.DateTimeField(default=timezone.now)

**make sure to use migrate commands after creating classes**

**python manage.py makemigrations**

**python manage.py migrate**

**How to show data from database in views?**

1. **from .models import Post**
   1. **this is to import the model that you will use to retrieve data**

**Install django-crispy-forms to formlize your forms and make them look good.**

**After installing add them to installed apps as crispy\_forms , also in the setting add the following setting**

**CRISPY\_TEMPLATE\_PACK = 'bootstrap4'**