**File Hierarchy:**

modivibe/ *<-* root directory of project

manage.py *<-* a command line utility that interacts with django project

modivibe/ *<-* python package (used to import anything inside it (modivibe.urls))

\_\_init\_\_.py *<-* an empty file telling python this is a python package

asgi.py *<-* entry point for ASGI-compatible web servers

settings.py *<-* settings for django project

urls.py *<-* url declarations ("table of contents" of django site)

wsgi.py *<-* entry point for WSGI-compatible web servers

webplayer/

migrations/

static/

templates/

views/

create\_html.py

pages.py

spotipy\_api.py

admin.py

apps.py

models.py

SpotifyApiObjs.py

tests.py

urls.py

**View HTML:**

*-* HttpResponse

[**python**]

# webplayer/views.py

from django.http import HttpResponse

def index(request):

return HttpResponse("Hello, World!")

[**end**]

**Map to URL:**

[**python**]

# webplayer/urls.py

from django.urls import path

from . impoort views

urlpatterns = [

path('', views.index, name='index'),

]

# modivibe/urls.py

from django.contrib import admin

from django.urls import include, path

urlpatterns = [

path('', include('webplayer.urls')),

#path('admin/', admin.site.urls),

]

[**end**]

**Path Function:**

*- route / 1st parameter:* URL pattern, django goes through all of urlpatterns to find requested URL's matched pattern

*- view / 2nd paramter:* after finding the matching pattern, calls the view function with an HttpRequest object

*- name:* lets you refer to it unambiguously from elsewhere,

**Database, Models of DBs, Admin site:**

https://docs.djangoproject.com/en/3.2/intro/tutorial02/

**Variable urls:**

[**python**]

#urls.py

urlpatterns = [

# ex: /polls/

path('', views.index, name='index'),

# ex: /polls/5/

path('<int:question\_id>/', views.detail, name='detail'),

# ex: /polls/5/results/

path('<int:question\_id>/results/', views.results, name='results'),

# ex: /polls/5/vote/

path('<int:question\_id>/vote/', views.vote, name='vote'),

]

#########

#views.py

def detail(request, question\_id):

return HttpResponse("You're looking at question %s." % question\_id)

[**end**]

**HTML code:**

[**python**]

{%if latest\_question\_list %}

{% else %}

{% endif %}

{% for question in question.choice\_set.all %} #.all = all() which grabs all the items in the dictionary

{{ question.text }}

{{forloop.counter}}

{% endfor %}

[**end**]

**Get HTML and render:**

[**python**]

from django.http import HttpResponse

from django.template import loader

def index(request):

...

#html

template = loader.get\_template('polls/index.html')

#variables

context = {

'latest\_question\_list': latest\_question\_list,

}

return HttpResponse(template.render(context, request))

OR

from django.shortcuts import render #no HttpResponse or loader necessary

def index(request):

context = {'latest\_question\_list': latest\_question\_list}

return render(request, 'polls/index.html', context)

[**end**]

**Raise 404:**

[**python**]

from django.http import Http404

from django.shortcuts import render

def detail(request, question\_id):

try:

question = Question.objects.get(pk=question\_id)

except Question.DoesNotExist:

raise Http404("Question does not exist")

return render(request, 'polls/detail.html', {'question':question})

OR

from django.shortcuts import get\_object\_or\_404, render

def detail(request, question\_id):

question = get\_object\_or\_404(Question, pk=question\_id)

[**end**]

**Reference URL:**

{% url 'detail' question.id %} = "/polls/{{question.id}}/"

*####*

*# the 'name' value as called by the {% url %} template tag*

path("<int:question\_id>/", views.detail, name='detail')

**POST Forms:**

*-* Since we’re creating a POST form (which can have the effect of modifying data),

we need to worry about Cross Site Request Forgeries. Thankfully, you don’t have to worry too hard,

because Django comes with a helpful system for protecting against it.

In short, all POST forms that are targeted at internal URLs should use the {% csrf\_token %} template tag.

*-* request.POST

let's you access submitted data by key name

**Images:**

https://docs.djangoproject.com/en/3.2/intro/tutorial06/