**Python Virtual Env Commands**

**Django Installation**

pip install django

pip install django==1.9

**Create Django Project**

django-admin startproject tictactoe

**Run Django Server**

python manage.py runserver

**Django Imp Files**

tictactoe/\_\_pycache\_\_

tictactoe/\_\_init\_\_.py

tictactoe/settings.py

tictactoe/urls.py

tictactoe/wsgi.py

db.sqllite3

manage.py

**Python Migrations Commands**

python manage.py migrate

python manage.py showmigrations

**To make migrations**

python manage.py makemigrations

python manage.py sqlmigrate gameplay 0001

python manage.py inspectdb

**urls.py**

from django.conf.urls import url,include

from django.contrib import admin

from .views import welcome

urlpatterns=[

url(r^’admin/’,admin.site.urls),

url(r’welcome’,welcome)

url(r’^starts\_with\_welcome’,welcome)

url(r’^player/’,include(‘players.url’)),

url(r’$ends\_with\_welcome’,welcome)

]

**player/urls.py**

from django.conf.urls import url

from .views import home

urlpatterns = [

url(r’home$’,home)

]

**Note**:-

To use url patterns from player/urls.py

from django.contrib.urls import url,include

urlpatterns=[

url(r’^player/’,include(‘players.urls’))

]

**views.py – helloworld**

from django.http import HttpResponse

def welcome(request):

return HttpResponse(“Hello World”)

**player/views.py**

from django.shortcuts import render

def home(request):

return render(request,”player/home.html”)

**Django App**

python manage.py startapp gameplay

python manage.py startapp player

**Add your app in settings.py**

INSTALLED\_APPS=[

‘django.contrib.admin’,

‘django.contrib.auth’,

‘django.contrib.contenttypes’,

‘django.contrib.sessions’,

‘django.contrib.messages’,

‘django.contrib.staticfiles’,

‘gameplay’

]

**Files in Django App :**

admin.py

apps.py

\_\_init\_\_.py

migrations

models.py

test.py

views.py

**models.py**

from \_\_future\_\_ import unicode\_literals

from django.utils.encoding import python\_2\_unicode\_compatible

from djangi.contrib.auth.models import User

from django.db import models

#create your models here

GAME\_STATUS\_CHOISES=(

(‘F’,’First player to move’),

(‘S’,’Second player to move’),

(‘W’,’First player wins’),

(‘L’,’Second player wins’),

(‘D’,’Draw’)

)

@python\_2\_unicode\_compatible

Class Move(models.Model):

X=models.IntegerField()

Y=models.IntergerField()

comments=models.CharField(mx\_length=300,blank=True)

by\_first\_player=models.BooleanField()

Class Game(models.Model):

first\_player=models.ForeignKey(User,related\_name=”games\_first\_player”)

second\_player=models.ForeignKey(User,related\_name=”games\_second\_player”)

start\_time=models.DateTimeField(auto\_now\_add=True)

last\_active=models.DateTimeField(auto\_now=True)

status=models.CharField(max\_length=1,default=’F’,choices=GAME\_STATUS\_CHOISES)

game=models.ForeignKey(Game,on\_delete=models.CASCADE)

def \_\_str\_\_(self):

return “{1} vs {2}”.format(self.first\_player,self.second\_player)

**Python Batteries**

ORM

URL Mapping

Templates

Forms

Admin

Packages

Architecture

**To register model to Admin site**

**admin.py**

from django.contrib import admin

from .models import Game,Move

admin.register(Game)

admin.register(Move)

admin.site.register(Game)

admin.site.register(Move)

**site** <https://127.0.0.1:8000/admin>

**Command to create superUser for admin**

python manage.py createsuperuser

**admin site customization**

**admin.py**

from django.contrib import admin

from .models import Game,Move

@admin.register(Game)

class GameAdmin(admin.ModelAdmin):

list\_display=(‘id’,’first\_player’,’second\_player’,’status’)

list\_editable=(‘status’,)

admin.site.register(Move)

Simple Example:-

from django.shortcuts import render

from gameplay.models import Game

def home(request):

return render(request,”player/home.html”,

{ ‘ngames’: Game.objects.count() })

Advance Example :

from django.shortcuts import render

from gameplay.models import Game

def home(request):

games\_first\_player = Game.objects.filter(

first\_player=request.user

status=’F’

)

games\_second\_player = Game.objects.filter(

second\_player=request.user

status=’S’

)

all\_my\_games= list(game\_first\_player) + \

list(game\_second\_player)

return render(request,”player/home.html”,

{ ‘games’: all\_my\_games})

**How to make TEMPLATES?**

**player/templates/player/home.html**

<!DOCTYPE html>

<html lang=”en”>

<head>

<meta charset=”UTF-8”>

<title>Home</title>

</head>

<body>

<h1>Welcome Home</h1>

There are currently {{ ngames }} in the database

</body>

</html>

How to fetch data of model object in a html template

<!DOCTYPE html>

<html lang=”en”>

<head>

<meta charset=”UTF-8”>

<title>Home</title>

</head>

<body>

<h1> Welcome {{ user.username }} </h1>

These are your active games

<ul>

{% for g in games %}

<li> Game {{ g.id }} : {{ g.first\_player }} vs {{ g.second\_player }} </li>

{% endfor %}

</body>

</html>

How to write customer Query using Q function?

from django.db.models import Q

class GameQuerySet(models.QuerySet):

def games\_for\_user(self,user):

return self.filter(

Q(first\_player=user) | Q (second\_player=user)

def active(self):

return self.filter(

Q(status=’F’) |Q(status=’S’)

)

Not we need to overwrite object attribute so that we call our customer query methods:

So add below line in games class

objects=GamesQuerySet.as\_manager()

~Q(status=’F’) for not condition

Now in views functions simply call

Games.objects. games\_for\_user(request.user)

Inbuild functions for quering database

Game.objects.filter(first\_player=request.user,status=”F”)

Note : Filter & exclude works only for and conditions

**How to make our text look good ?**

1] make below folder structure in your app

static/app-name/

2] Create style.css inside it

style.css

body {

color:red;

}

3] Modify your home.html file

<!DOCTYPE html>

{% load static %}

<html lang=”en”>

<head>

<meta charset=”UTF-8”>

<title> Home </title>

**<link href=”{% static player/style.css %}” rel=”stylesheet”>**

</head>

<body>

<h1> Welcome {{ user.username }} </h1>

These are your active games

<ul>

{% for g in games %}

<li> Game {{ g.id }} : {{ g.first\_player }} vs {{ g.second\_player }} </li>

{% endfor %}

</ul>

</body>

</html>

**How to make professional static page**

1] Go to [www.initializr.cpm](http://www.initializr.cpm)

2] Choose bootstrap option and download static content

3] Put this static data in static folder in your app

4] index.html is the basic html file . Move this file to template folder and rename it as base.html

5] Change Project to “Django Fundamental”

<a class =”navbar-brand” href=”/’>Django Fundamentals</a>

*this / means links points to your welcome page.*

6] Replace content with your own content.

<base.html>

<div class=”container”>

{% block content %}

{% endblock %}

</div>

7] Now your home.html should inherit base.html

{% extends “base.html” %}

8] Now we don’t need any head tags , html tags, in home.html

{% extends “base.html” %}

{% block content %}

<h1> Welcome , {{ user.username }} </h1>

There are your active games

<ul>

{% for g in games %}

<li> Game {{ g.id }} : {{ g.first\_player }} vs {{ g.second\_player }} </li>

{% endfor %}

</ul>

{% endblock %}

9] Add load static tag in base.html

{% load static %}

<script src=”stylesheet” href=”{% static ‘css/bootstrap.min.css’ %}”>

Add static folder path in settings.py

STATICFILES\_DIR = [

os.path.join(BASE\_DIR,”static”)

]

Add templates path too

‘DIRS’:[os.path.join(BASE\_DIR,”templates”)]

Similarly we can re-use our code in views.py

from django.shortcuts import render

def welcome(request):

return render(request,’tictactoe/welcome.html’)

welcome.html

{% extends base.html %}

{% block title %}

Welcome

{% endblock %}

{% block content %}

<div class=”jumbtron”>

<div class=”container”>

<h1> Hello, world</h1>

<p> Lets play tictactoe </p>

</div>

</div>

{% endblock %}