

Dharmsinh Desai University, Nadiad
Faculty of technology,
Department of Computer Engineering
Subject : Software Project
Lab Manual

Aim: Building web application with Django.

PART-I: To learn Django basics and create a simple 'hello world' application.

Requirements: Python 3.X, Django 2.X

1. Introduction: Django is the Web Framework of Python which can be used to build Web Applications. Django can be (and has been) used to build almost any type of website, from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc).

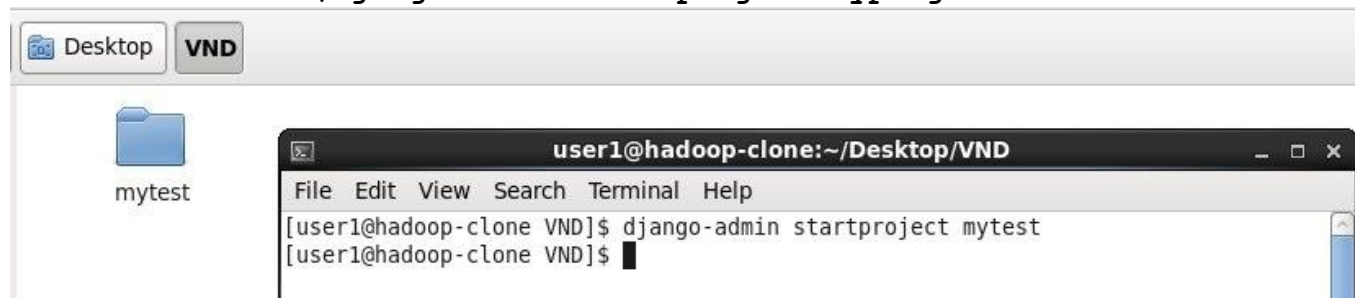
❖ Django framework is designed on **Model – View – Template(MVT)** architecture. Every django application contains models.py, views.py and urls.py file.

- **Models.py** file defines data for application. All tables descriptions are defined in model.py file.
- **Views.py** file defines programs to manipulate and access data defined in models.py file. Most of the functions (business logic) are specified in views.py. Various functionalities defined in views.py can be accessed on frontend (web pages) via templates. References to these templates (html, css, scripts etc) are stored in urls.py.

2. Creating a new project

- 1) Open terminal and go to the directory where you want to create the project.
- 2) Run following command,

\$django-admin startproject myproject



- This will create a new project directory named 'myproject' (instead of writing myproject, you may give appropriate name for your project.). The newly created files and folders inside the 'myproject' directory are as follows,
myproject

```

--> manage.py
--> myproject
    --> __init__.py
    --> settings.py
    --> urls.py
    --> wsgi.py
[user1@hadoop-clone mytest]$ tree
.
├── manage.py
└── mytest
    ├── __init__.py
    ├── __init__.pyc
    ├── settings.py
    ├── settings.pyc
    ├── urls.py
    └── wsgi.py

1 directory, 7 files
[user1@hadoop-clone mytest]$ █

```

Description of these files and directories is as follows,

- The outer myproject/ root directory is just a container for your project. (Its name doesn't matter to Django; you can rename it to anything you like.)
- **manage.py**: A command-line utility to manage the project. (We will see its use with example.)
- The inner myproject/ directory is the actual Python package for your project. Its name is the Python package name you'll need to use to import anything inside it (e.g. **demo.urls**).
- **__init__.py**: An empty file that tells Python that this directory should be considered as a Python package.
- **settings.py**: This file stores Settings/configuration for the project.
- **urls.py**: The URL declarations for the project. (a "table of contents" of your Django-powered site.)
- **wsgi.py**: An entry-point for WSGI-compatible web servers to serve your project.

3. Creating an WebApplication inside the project

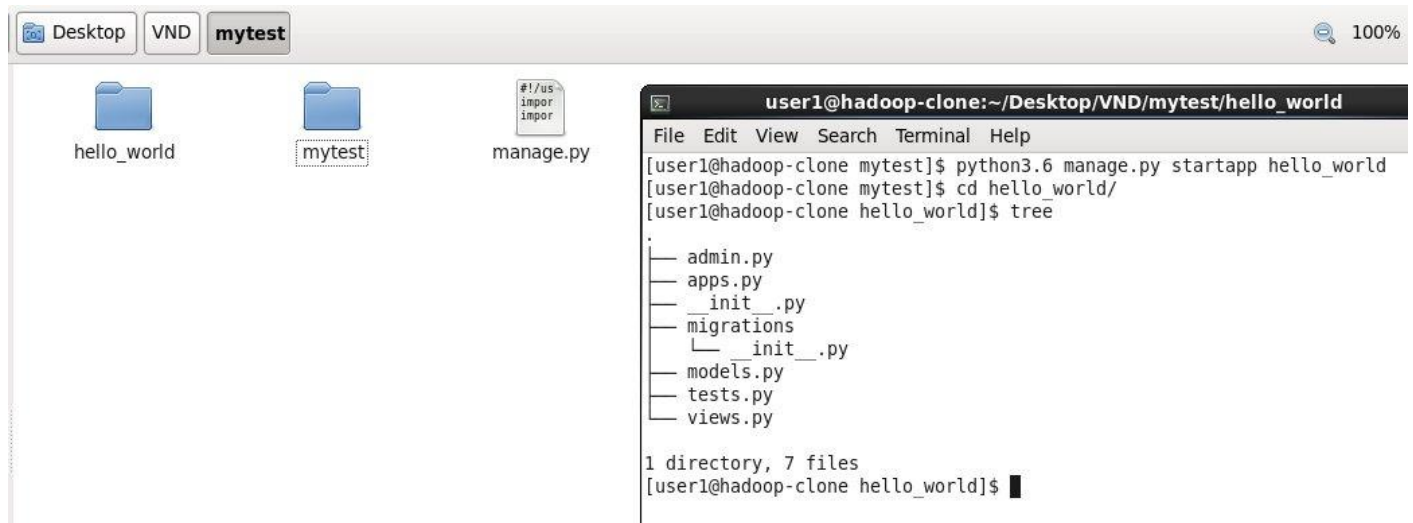
- A WebApp is a Web application that provides some service through website – e.g., a Weblog system, a database of public records or a simple poll app. A project is a collection of configuration and apps for a particular website. A project can contain **multiple** apps. **An app can be in multiple projects.**

Step 1 : To create your app, make sure you're in the same directory as **manage.py** and type this command:

```
$python3.6 manage.py startapp hello_world
```

This will create a directory for 'hello_world' app, which contains following files,

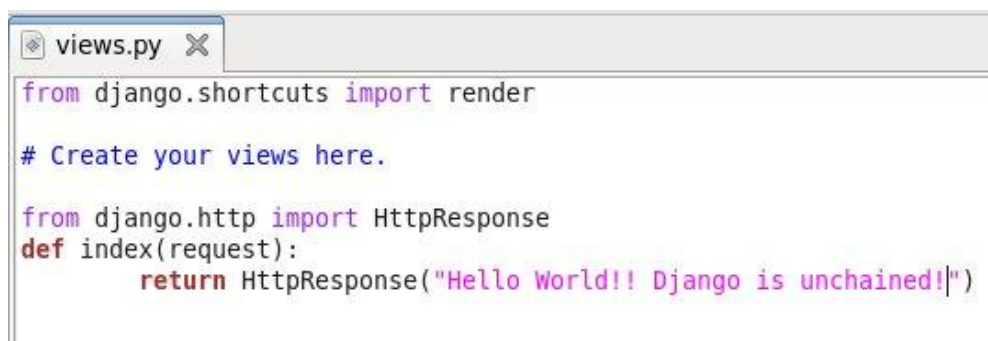
```
--> hello_world
    --> admin.py
    --> __init__.py
    --> models.py
    --> tests.py
    --> views.py
    --> migrations
```



- Your apps can live anywhere on your Python path. But we'll create our hello_world app right next to manage.py file so that it can be imported as its own top-level module, rather than a submodule of myproject
- model.py is the file which specifies database table descriptions. Most of the functions (business logic) are specified in views.py. (We will discuss use of other files later)

Step 2 : Write following code in views.py file (/hello_world/views.py) of the project.

```
from django.http import HttpResponse
def index(request):
    return HttpResponse("hello World!! Django is unchained!")
```

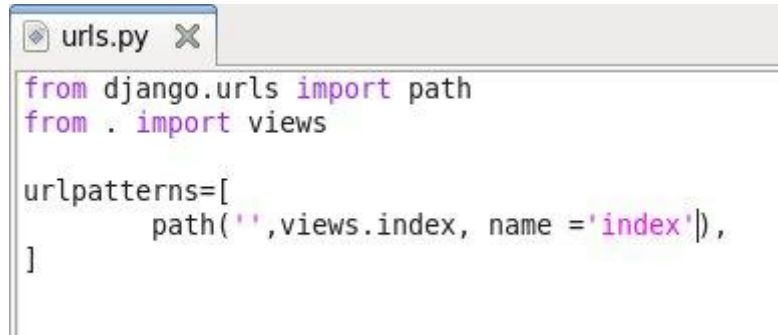


- This is the simplest view possible in Django. To call the view, we need to map it to a URL - and for this we need a URLconf.

Step 3 : To create a URLconf for hello_world app, create urls.py file in hello_world directory and write following code in it.

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```



Step 4 : The next step is to point the root URLconf at the hello_world.urls module. In myproject/urls.py, add an import for django.urls.include and insert an include() in the urlpatterns list. So your myproject/urls.py should look like this:

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('hello_world/', include('hello_world.urls')),
]
```

- The include() function allows referencing other URLconfs. Whenever Django encounters include(), it chops off whatever part of the URL matched up to that point and sends the remaining string to the included URLconf for further processing. The idea behind include() is to make it easy to plug-and-play URLs.

```

urls.py x
"""mytest URL Configuration

The 'urlpatterns' list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/2.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('hello_world/', include('hello_world.urls')),
]

```

➤ You have now wired an index view into the URLconf. Lets verify it's working.

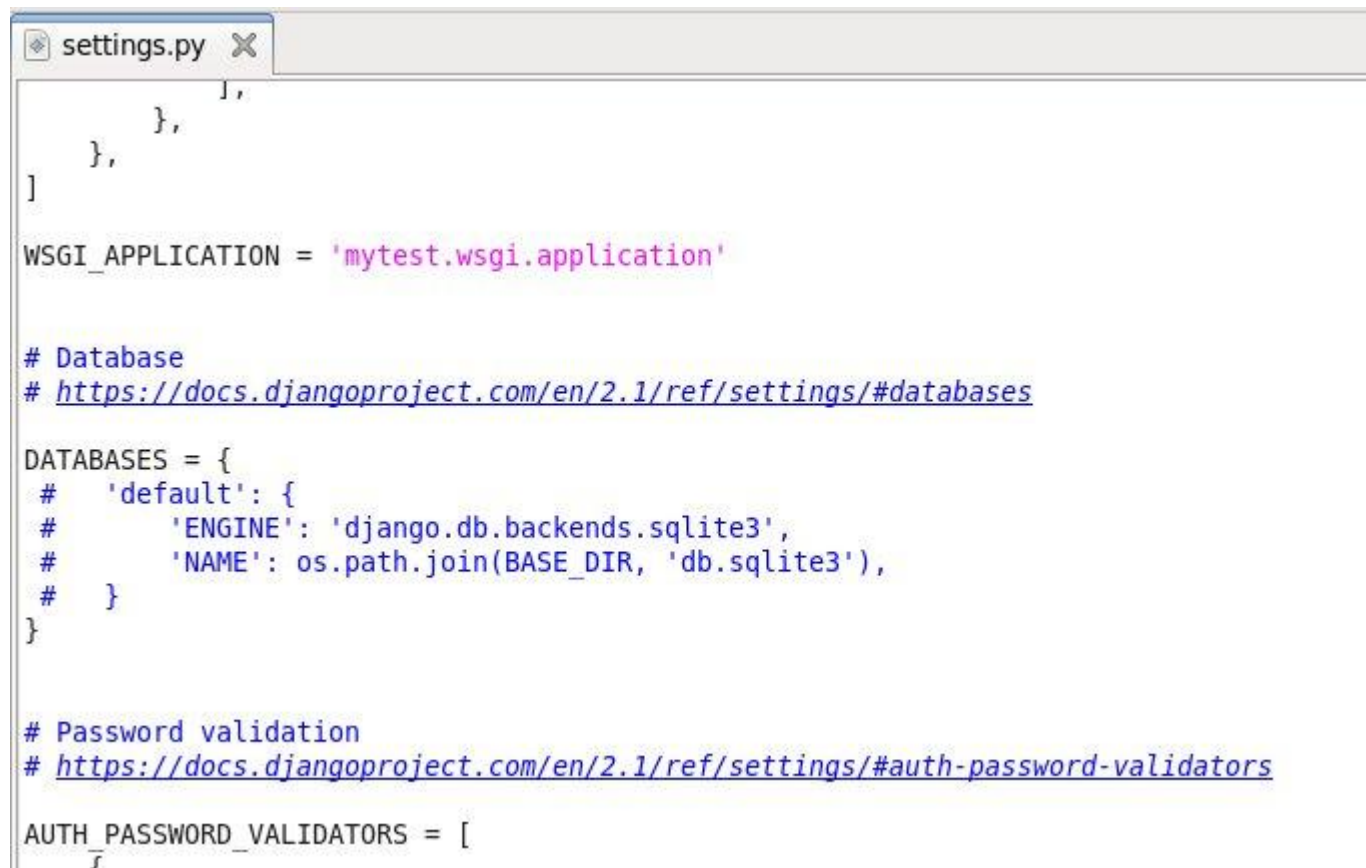
```

[user1@hadoop-clone hello_world]$ gedit views.py
[user1@hadoop-clone hello_world]$ gedit urls.py
[user1@hadoop-clone hello_world]$ cd..
bash: cd.: command not found
[user1@hadoop-clone hello_world]$ cd ..
[user1@hadoop-clone mytest]$ ls
hello_world  manage.py  mytest
[user1@hadoop-clone mytest]$ cd mytest/
[user1@hadoop-clone mytest]$ ls
__init__.py  __pycache__  settings.pyc  wsgi.py
__init__.pyc  settings.py  urls.py
[user1@hadoop-clone mytest]$ gedit urls.py
[user1@hadoop-clone mytest]$ cd ..
[user1@hadoop-clone mytest]$ cd mytest/

```

flow of execution

Step 5: Now open myproject/settings.py and in the **DATABASE** if default database have been set then comment them as we are not doing any connection with any database, otherwise it may give error.



```
settings.py
1,
},
},
]

WSGI_APPLICATION = 'mytest.wsgi.application'

# Database
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases

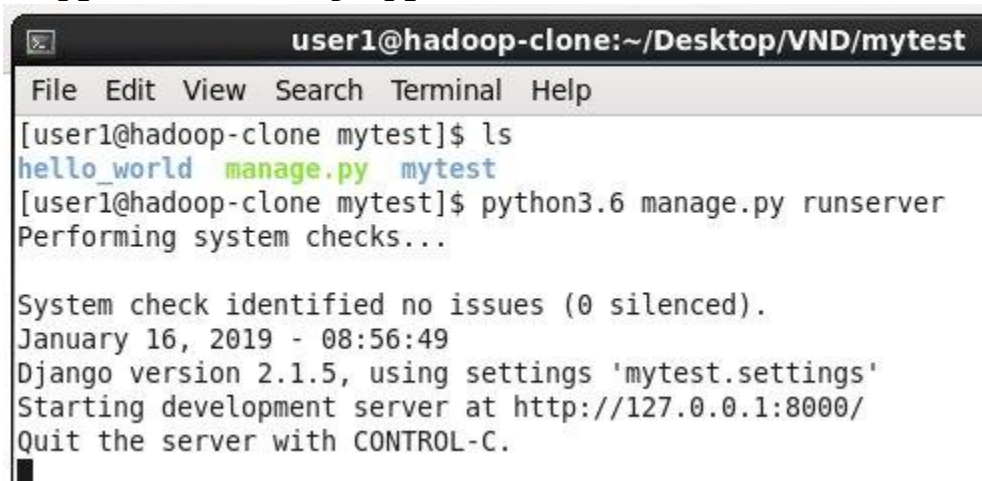
DATABASES = {
    # 'default': {
    #     'ENGINE': 'django.db.backends.sqlite3',
    #     'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    # }
}

# Password validation
# https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
```

Step 6 : Start the server: Go to project directory (where manage.py is located) and run the following command.

```
$python3.6 manage.py runserver
```



```
user1@hadoop-clone:~/Desktop/VND/mytest
File Edit View Search Terminal Help
[user1@hadoop-clone mytest]$ ls
hello_world manage.py mytest
[user1@hadoop-clone mytest]$ python3.6 manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
January 16, 2019 - 08:56:49
Django version 2.1.5, using settings 'mytest.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Step 7: Run the application

Open the web browser and enter the following url, 127.0.0.1:8000/hello_world/

PART-II: Web Application development with Python and Django 2 : Working with views and templates

Introduction:

This tutorial shows basic example to work with views.py and templates. Follow the step describe here to create a basic application which uses views and template. (Some steps are repeated from tutorial. If required, refer to build 'hello_world' web application tutorial for details.).

Step 0 : Create a django project :

```
$django-admin startproject viewdemo
```

Step 1 : Create an App:

```
$python3.6 manage.py startapp viewdemoapp
```

Step 2: Register new app in settings.py : Add an entry for viewdemoapp in settings.py (*viewdemo/settings.py*) as follows,

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'viewdemoapp'  
]
```

Step 3 : Define template: Now we need to define a template which will be rendered whenever view.py is called. Inside the application folder (*/viewdemoapp/*) create a template directory.

```
$mkdir templates
```

Go to the newly created templates directory and create an index.html page. Write following code in the index.html page.

```
<!DOCTYPE html>  
<html>  
    <head>  
        <meta charset="utf-8">  
        <title>Howdy!</title>  
    </head>  
    <body>  
        <h1>Hi! Django views and template test  
successful!</h1>
```

```

    </body>
</html>

```

Step 4: Define view: Open viewdemoapp's views.py file (*viewdemoapp/views.py*) and create following views.

```

from django.shortcuts import render
from django.views.generic import TemplateView

# Create your views here.
class HomePageView(TemplateView):
    def get(self, request, **kwargs):
        return render(request, 'index.html', context=None)

```

Step 5 : Update project and app's urls.py file : To access the webpage defined in 'viewdemoapp' we need to define url of the page in projects urls.py (*viewdemo/urls.py*). Update the existing urls.py (*viewdemo/urls.py*) file as follows,

```

from django.conf.urls import url, include
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^', include('viewdemoapp.urls')),
]

```

include('viewdemoapp.urls') defines location of urls file as */viewdemoapp/urls.py*. Now go to viewdemoapp folder and create a new urls.py file.

```

from django.conf.urls import url
from viewdemoapp import views

urlpatterns = [
    url(r'^', views.HomePageView.as_view()),
]

```

- This code imports the views from *viewdemoapp* app and expects a view called ***HomePageView*** to be defined.
- This file defines a view called ***HomePageView***. Django views take in a request and return a response. In our case, the method get expects a HTTP GET request to the url defined in our urls.py file. Once a HTTP GET request has been received, the method renders a template called index.html which is just a normal HTML file which could have special Django template tags written alongside normal HTML tags.

Step 6 : Run server


```
$python3.6 manage.py runserver
```

Step 7: Open web browser and check output at following url.

`http://127.0.0.1:8000/viewdemoapp`

Exercise: Create HTML pages (front end) for your project. Also, update models.py as per the table design of your project (If not already completed in last lab).