Chapter-2

Django & Atom Installation
And
Development of First Web Application

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

How to install django:

1. Make sure Python is already installed in our system

python --version

2. Install django by using pip

pip install django
pip install django == 1.11.9

D:\>pip install django
Collecting django
Downloading
https://files.pythonhosted.org/packages/51/1a/6153103322/Django-2.1-py3-none-any.whl (7.3MB) 100% || 7.3MB 47kB/s

Collecting pytz (from django)
Downloading https://files.pythonhosted.org/packages/30/4e/53b898779a/pytz-2018.5-py2.py3-none-any.whl (510kB)
100% || 512kB 596kB/s

Installing collected packages: pytz, django
Successfully installed django-2.1 pytz-2018.5
You are using pip version 9.0.3, however version 18.0 is ava
You should consider upgrading via the 'python -m pip install

3. To check django version:

py -m django --version

ATOM IDE/Editor:

Install ATOM IDE from the following link → https://atom.io/

Speciality of ATOM IDE:

- **It is freeware.**
- **It is open source.**
- **It supports cross platform.**
- **It provides several auto completion short-cuts for easy development etc**

How to Configure Atom for Python:

1) Terminal Installation:

File → Settings → Install → In the searchbox just type terminal → platform-ide-terminal

2) Python AutoCompletion:

File → Settings → Install → In the searchbox just type python → autocomplete-python

3) django:

File → Settings → Install → In the searchbox just type django → atom-django

4) How to Change Terminal from Powershell to Normal Command Prompt:

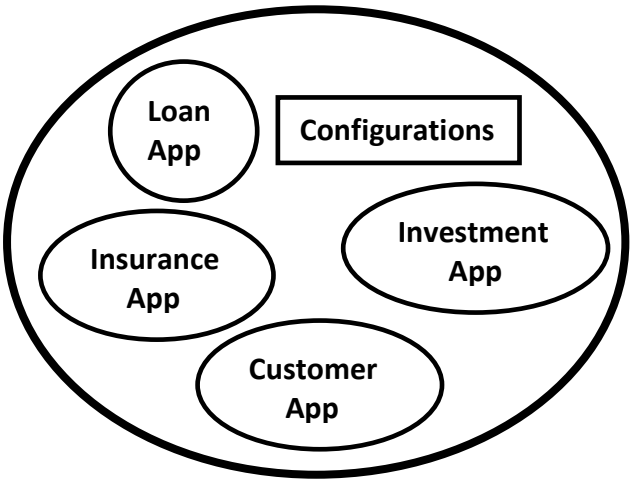
File → Settings → Install → In the searchbox just type terminal → platform-ide-terminal → settings → Shell Override

C:\Windows\System32\cmd.exe

Django Project vs Django Application:

A Django project is a collection of applications and configurations which forms a full web application.
Eg: Bank Project

A Dango Application is responsible to perform a particular task in our entire web application.
Eg: loan app
registration app
polling app etc



Bank Project

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

Project = Several Applications + Configuration Information

Note:

1) **The Django applications can be plugged into other projects.ie these are reusable. (Pluggable Django Applications)**
2) **Without existing Django project there is no chance of existing Django Application. Before creating any application first we required to create project.**

How to create Django Project:

Once we installed django in our system, we will get 'django-admin' command line tool, which can be used to create our Django project.

django-admin startproject firstProject

D:\>mkdir djangoprojects

D:\>cd djangoprojects

D:\djangoprojects>django-admin start-project firstProject

The following project structure will be created

```
D:\djangoprojects>
 |
 +---firstProject
    ¦
    ¦---manage.py
    ¦
    +---firstProject
        ¦---settings.py
        ¦---urls.py
        ¦--wsgi.py
        ¦-- __init__.py
```

__init__.py:

It is a blank python script.Because of this special file name, Django treated this folder as python package.

Note: If any folder contains __init__.py file then only that folder is treated as Python package.But this rule is applicable until Python 3.3 Version.

settings.py:

In this file we have to specify all our project settings and and configurations like installed applications, middileware configurations, database configurations etc

urls.py:

- **Here we have to store all our url-patterns of our project.**
- **For every view (web page), we have to define separate url-pattern. End user can use url-patterns to access our webpages.**

wsgi.py:

- **wsgi → Web Server Gateway Interface.**
- **We can use this file while deploying our application in production on online server.**

manage.py:

- **The most commonly used python script is manage.py**
- **It is a command line utility to interact with Django project in various ways like to run development server, run tests, create migrations etc.**

How to Run Django Development Server:

We have to move to the manage.py file location and we have to execute the following command.

```
py manage.py runserver
```

```
D:\djangoprojects\firstProject>py manage.py startserver

Performing system checks...

System check identified no issues (0 silenced).

You have 13 unapplied migration(s). Your project may not work properly until you apply
the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
August 03, 2018 - 15:38:59
Django version 1.11, using settings 'firstProject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Now the server started.

How to Send First Request:

Open browser and send request:

http://127.0.0.1:8000/

The following should be response if everything goes fine.

It worked!
Congratulations on your first Django-powered page.
Next, start your first app by running python manage.py startapp [app_label].

You're seeing this message because you have DEBUG = True in your Django settings file and you haven't configured any URLs. Get to work!

Role of Web Server:

- **Web Server provides environment to run our web applications.**
- **Web Server is responsible to receive the request and forward request to the corresponding web component based on url-pattern and to provide response to the end user.**
- **Django framework is responsible to provide development server. Even Django framework provides one inbuilt database sqlite. Special Thanks to Django.**

Note: Once we started Server a special database related file will be generated in our project folder structure.

db.sqlite3

Creation of First Web Application:

Once we creates Django project, we can create any number of applications in that project.

The following is the command to create application.

python manage.py startapp firstApp

D:\djangoprojects\firstProject>python manage.py startapp firstApp

The following is the folder structure got created.


```
D:\djangoprojects>
└──firstProject
   │  db.sqlite3
   │  manage.py
   │
   ├──firstApp
   │  │  admin.py
   │  │  apps.py
   │  │  models.py
   │  │  tests.py
   │  │  views.py
   │  │  __init__.py
   │  │
   │  └──migrations
   │        __init__.py
   │
   └──firstProject
      │  settings.py
      │  urls.py
      │  wsgi.py
      │  __init__.py
      │
```

Note: Observe that Application contains 6 files and project contains 4 files+ one special file: manage.py

1) __init__.py:

It is a blank Python script. Because of this special name,Python treated this folder as a package.

2) admin.py:

We can register our models in this file. Django will use these models with Django's admin interface.

3) apps.py:

In this file we have to specify application's specific configurations.

4) models.py:

In this file we have to store application's data models.

5) tests.py:

In this file we have to specify test functions to test our code.

6) views.py:

In this file we have to save functions that handles requests and return required responses.

7) Migrations Folder:

This directory stores database specific information related to models.

Note: The most important commonly used files in every project are views.py and models.py

Activities required for Application:

Activity-1: Add our application in settings.py,so that Django aware about our application.

In settings.py:

```
1) INSTALLED_APPS = [
2)     'django.contrib.admin',
3)     'django.contrib.auth',
4)     'django.contrib.contenttypes',
5)     'django.contrib.sessions',
6)     'django.contrib.messages',
7)     'django.contrib.staticfiles',
8)     'firstApp'
9) ]
```

Activity-2: Create a view for our application in views.py.

- **View is responsible to prepare required response to the end user. i.e view contains business logic.**
- **There are 2 types of views.**
    1) **Function Based Views**
    2) **Class Based Views**

- **In this application we are using Function based views.**

views.py:

```
1) from django.shortcuts import render
2) from django.http import HttpResponse
3)
4) # Create your views here.
5) def display(request):
```


```
6)     s='<h1>Hello Students welcome to DURGASOFT Django classes!!!</h1>'
7)     return HttpResponse(s)
```

Note:

1) Each view will be specified as one function in views.py.
2) In the above example display is the name of function which is nothing but one view.
3) Each view should take atleast one argument (request)
4) Each view should return HttpResponse object with our required response.

View can accept request as input and perform required operations and provide proper response to the end user.

Activity-3: Define url-pattern for our view in urls.py file.

- This url-pattern will be used by end-user to send request for our views.
- The 'urlpatterns' list routes URLs to views.

For functional views we have to do the following 2 activities:

1) Add an import: from firstApp import views
2) Add a URL to urlpatterns: url(r'^greeting/', views.display)

urls.py:

```
1)  from django.conf.urls import url
2)  from django.contrib import admin
3)  from firstApp import views
4)
5)  urlpatterns = [
6)      url(r'^admin/', admin.site.urls),
7)      url(r'^greetings/', views.display),
8)  ]
```

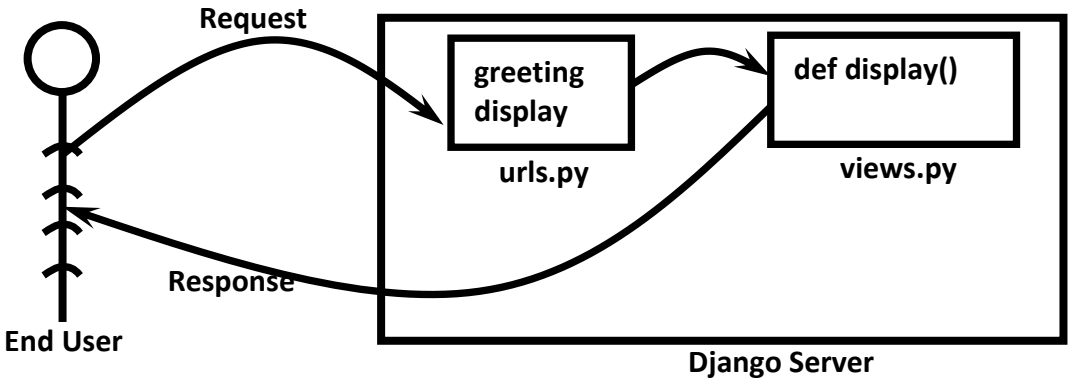
Whenever end user sending the request with urlpattern: greeting then disply() function will be executed and provide required response.

Activity-4: Start Server and Send the request

py manage.py runserver

http://127.0.0.1:8000/greetings

Http Request flow in Django Application:



1. Whenever end user sending the request first Django development server will get that request.

2. From the Request django will identify urlpattern and by using urls.py, the corresponding view will be identified.

3. The request will be forwared to the view. The corresponding function will be executed and provide required response to the end user.

Summary of Sequence of Activities related to Django Project:

1) Creation of Django project
   django-admin startproject firstProject

2) Creation of Application in that project
   py manage.py startapp firstApp

3) Add application to the Project
   (inside settings.py)

4) Define view function inside views.py
5) Define url-pattern for our view inside urls.py

6) Start Server
   py manage.py runserver

7) Send the request

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

How to change Django Server Port:

By default Django develoment server will run on port number: 8000. But we can change port number based on our requirement as follows.

py manage.py runserver 7777

Now Server running on port number: 7777
We have to send the request with this port number only

http://127.0.0.1:7777/greetings/
http://127.0.0.1:8000/time/

Various Practice Applications:

1. Write Django Application just to send Helloworld message as response.
2. Write Django application to send server time as response
3. Single application with multiple views

views.py:

```
1)  from django.shortcuts import render
2)  from django.http import HttpResponse
3)
4)  # Create your views here.
5)  def good_morning_view(request):
6)      return HttpResponse('<h1>Hello Friend Good Morning!!!</h1>')
7)
8)  def good_evening_view(request):
9)      return HttpResponse('<h1>Hello Friend Good Evening !!!</h1>')
10)
11) def good_afternoon_view(request):
12)     return HttpResponse('<h1>Hello Friend Good Afternoon!!!</h1>')
```

urls.py

```
1)  from django.conf.urls import url
2)  from django.contrib import admin
3)  from testapp import views
4)
5)  urlpatterns = [
6)      url(r'^admin/', admin.site.urls),
7)      url(r'^morning/', views.good_morning_view),
8)      url(r'^afternoon/', views.good_afternoon_view),
```

```
    url(r'^evening/', views.good_evening_view),
]
```

4. Single project with multiple applications:

greetingapp: views.py

```
from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.
def greetings_view(request):
    #total html code 1000 lines of code
    return HttpResponse('<h1>Hello Friends Good Morning...Have a Nice Day</h1')
```

timeapp:views.py

```
from django.shortcuts import render
from django.http import HttpResponse
import datetime

# Create your views here.
def time_info_view(request):
    time=datetime.datetime.now()
    s='<h1>Hello Current Date and Time is :'+str(time)+'</h1>'
    return HttpResponse(s)
```

Issue with urls.py

```
from greetingapp import views
from timeapp import views
```

Take special care while defining url patterns.we will get error b'z only one views.py is available. But we can solve this problem with any of the following 2 ways

First Approach

```
from greetingapp.views import greetings_view
from timeapp.views import time_info_view

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^greetings/',greetings_view),
    url(r'^time/',time_info_view),
```

```
8) ]
```

Second Approach

```
1) from greetingapp import views as v1
2) from timeapp import views as v2
3)
4) urlpatterns = [
5)     url(r'^admin/', admin.site.urls),
6)     url(r'^greetings/',v1.greetings_view),
7)     url(r'^time/',v2.time_info_view),
8) ]
```

Q. Is it possible to define mutlitple url patterns for the same view function?
Answer: Yes, by using the following approach

```
1) urlpatterns = [
2)     url(r'^admin/', admin.site.urls),
3)     url(r'^$', views.wish),
4)     url(r'^test/', views.wish)
5) ]
```

http://127.0.0.1:8000/
http://127.0.0.1:8000/test

Defining URL Patterns at Application Level instead of Project Level:

A Django project can contain multiple applications and each application can contain multiple views.Defining url-patterns for all views of all applications inside urls.py file of project creates maintenance problems and reduces reusability of applications.

We can solve this problem by defining url-patterns at application level instead of project level. For every application we have to create a separate urls.py file and we have to define all that application specific urls in that file. We have to link this application level urls.py file to project level urls.py file by using include() method.

Demo Application:

1. Creation of Project
django-admin startproject urlProject

2. Creation of Application
py manage.py startapp testapp

3. Add our application to the Project inside settings.py file

```
INSTALLED_APPS=[
        .........
            'testapp'
            ]
```

4. Define View Function in views.py

views.py

```
1)  from django.shortcuts import render
2)  from django.http import HttpResponse
3)
4)  # Create your views here.
5)  def first_view(request):
6)      return HttpResponse('<h1>Response from First View</h1>')
7)
8)  def second_view(request):
9)      return HttpResponse('<h1>Response from Second View</h1>')
10)
11) def third_view(request):
12)     return HttpResponse('<h1>Response from third View</h1>')
13)
14) def fourth_view(request):
15)     return HttpResponse('<h1>Response from Fourth View</h1>')
16)
17) def fifth_view(request):
18)     return HttpResponse('<h1>Response from Fifth View</h1>')
```

5. Create a seperate urls.py file inside application

testapp/urls.py

```
1)  from django.conf.urls import url
2)  from testapp import views
```

```
3)
4)  urlpatterns = [
5)      url(r'^first/', views.first_view),
6)      url(r'^second/', views.second_view),
7)      url(r'^third/', views.third_view),
8)      url(r'^fourth/', views.fourth_view),
9)      url(r'^fifth/', views.fifth_view),
10) ]
```

6. Include this application level urls.py inside project level urls.py file.
from django.conf.urls. import include

project level:urls.py

```
1)  from django.conf.urls import url,include
2)  from django.contrib import admin
3)
4)  urlpatterns = [
5)      url(r'^admin/', admin.site.urls),
6)      url(r'^testapp/', include('testapp.urls')),
7)  ]
```

6. Run Server
py manage.py runserver

7. Send Request
http://127.0.0.1:8000/urlApp/test

Note: We can see reusability of application in other project just with only 2 lines addition

1. settings.py -->add application name
2. urls.py --->just add: url(r'^urlApp/',include('urlApp.urls'),

Advantages:

The main advantages of defining urlpatterns at application level instead of project level are

1) **It promotes reusability of Django Applications across multiple projects**
2) **Project level urls.py file will be clean and more readable**