Internet and Web Technologies

BE(CSE) IV-Semester

Prepared by
S Durga Devi
Assistant Professor
Department of Computer Science and Engineering
Chaitanya Bharathi Institute of Technology

Unit-IV

- **Django:** Introduction
- Models
- Templates
- supported data bases
- URL configuration.
- Templates
- Modifying and Improving the Templates
- Creating a Form
- Connecting Django with databases
- Enable Django sessions.

S. Durga Devi, CSE,CBIT

Introduction to Django

- Django is web framework to build web applications
- Free and open source frame work and python related framework.
- Google, yahoo maps, Mozilla, dropbox, instagram, NASA, Netflix, spotify, youtube developed by Django.
- Django follows MVT(Model, View, Template) design pattern.
- Django maintained by DSF(Django software Foundation.
- Django official website : https://www.djangoproject.com/
- Django authors: Adrian Holovaty, Simon williams.

- Other python related frame works: flask, pyramid, Bottle, Torando,web2py,cherrypy etc.

Features of DJango framework

1. Fast
2. Fully loaded
3. Security
4. Scalability
5. versatile

1. Fast

95% of project work can be done by Django and remaining work will be completed developer due to its rich set of libraries

2. Fully loaded: for every web development the common things are authentication, security, session management, administrative acitivies will be taken care by django by providing lakhs of libraries.

3. security: provides strong security. Avoids SQL injection, cross – site scripting, cross- site request forgery etc.

4. scalability: handle large no of requests to meet heavy traffic demands.

5. versatile: Django used to develop small, large, scientific applications, companies, colleges, governments etc.

Installation of django in windows

1. Ensure python3 installed on your system.
2. Go to visual studio and open terminal
3. Upgrade pip install

   python –m pip install --upgrade pip

4. Install django

   pip install django

5. Check whether django installed properly or not

   5.1. type python in terminal

   5.2. import django

   django.VERSION

#create virtual environment

1. pip install virtualenv
2. pirtualenv enironmentname - demo
3. Go to demo
4. Goto folder Scripts
5. Activate environment using command activate.
6. Come back to demo and install django

S. Durga Devi, CSE,CBIT

Installing django in ubuntu

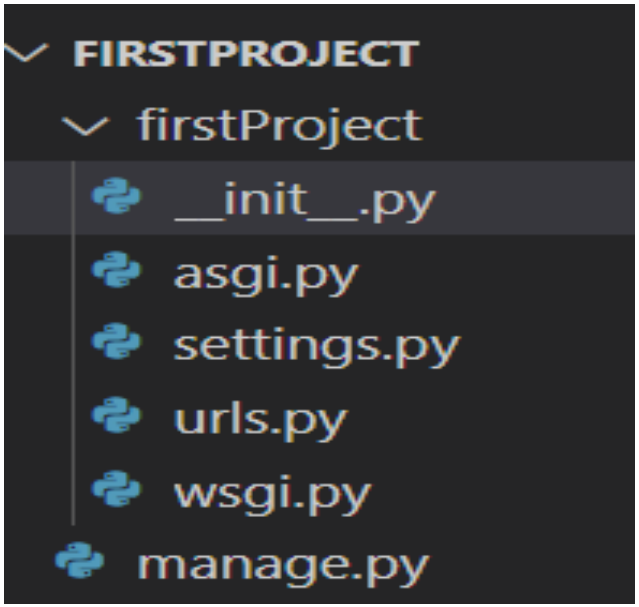
1. python3 –version
2. sudo apt install python3-django
3. Check for django version

   django-admin --version

Create django project

1. Go to specified foloder in command prompt.
2. django projects created by using ‘django-admin’ command line tool

   django-admin startproject firstProject



1. *__ init__.py:* if any folder contains this file will be considered as a python package.
2. *settings.py:* all project related settings and configurations will be specified like database configurations, installed apps and middleware configurations.
3. *urls.py:* specifies what are the urls are required to access particular page in web applications
4. *wsgi.py:* web server gateway interface. Used when application is deployed in online web servers.
5. *asgi.py:* Asynchronous server gateway interface. Communicates with web server and web application
6. *manage.py:* to create, start app, run servers.

How to run django server?

- Django provides inbuilt web server.
- Goto location where manage.py file is stored
- python manage.py runserver

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until y
ou apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
June 01, 2022 - 23:39:15
Django version 4.0.4, using settings 'firstProject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

- Copy url in browser http://127.0.0.1:8000/

- Stop server ctrl+c

Note: we can also specify port no : python manage.py runserver 8888

Role of web server

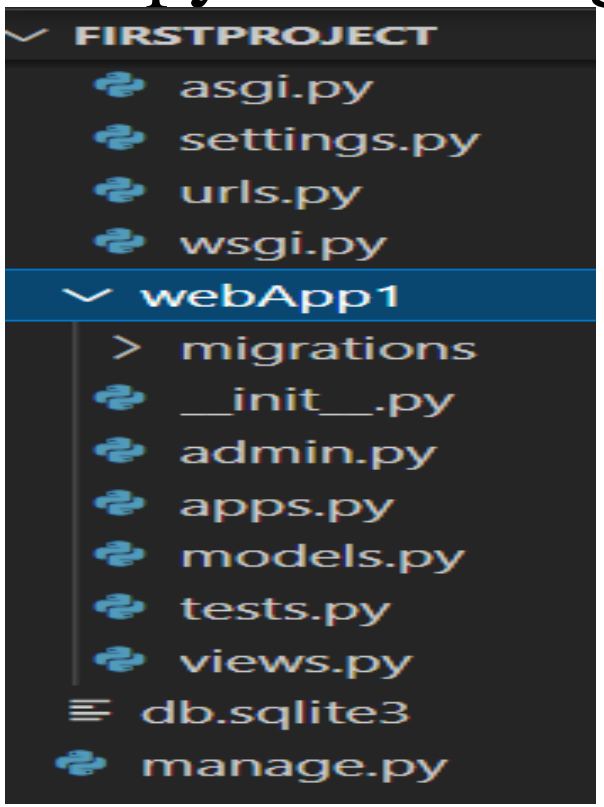
1. Web server provides environment to run web applications
2. Web server process request send by user.
3. Django framework is responsible to provide development server. Even Django framework provides one builtin database sqlite.

Note: Once we started Server a special database file will be created in our project folder structure named db.sqlite3

We can configure our own web servers (tomcat server, lighttpd) and databases also.

Create application in project

1. Goto project folder like myproject
2. python manage.py startapp webApp1



Most important file is views.py

Execute your first Django application

Activity1- settings.py

Add your app name

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles','webApp1'
]
```

Activity2- views.py

Add below code

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

# Create your views here.
def display(request):
    hello='<h1>Thank you DJango</h1>'
    return HttpResponse(hello)

```

Activity3- urls.py

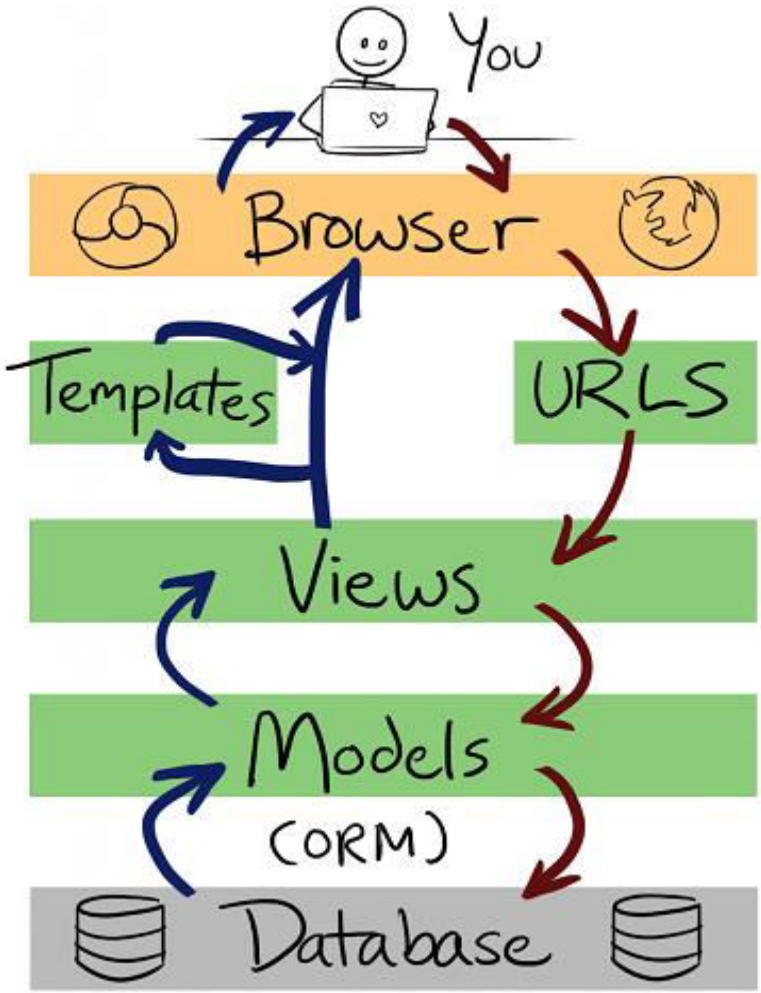
```
from django.contrib import admin
from django.urls import path
from webApp1 import views
from django.urls import re_path as url

urlpatterns = [url(r'^admin/', admin.site.urls),url(r'^greetings/', views.display), ]
```

Activity4

1. Start server: python manage.py runserver
2. Send request: http://127.0.0.1:8000/greetings

Http request flow in Django application



http://127.0.0.1:8000/greetings

S. Durga Devi, CSE,CBIT

https://medium.com/@humble_bee/django-basics-for-a-beginner-5d864e6aa084

Templates

- Not recommended to write html code inside of python views.py because

1. Reduces readability
2. Python developer should aware of both python and html
3. Does not support reusability of code.

Creating templates

1. Create project and application
2. Add application name in settings.py under INSTALLED APP
3. Create ‘templates’ folder inside main project folder
4. In settings.py add following code

```
from pathlib import Path
import os



# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
#print(BASE_DIR)
x=os.path.join(BASE_DIR,'templates')
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [x,],
        'APP_DIRS': True,
```

5. Inside template folder create results.html

appname/results.html

templates > firstTemplate > result.html > h1

```
<!DOCTYPE html>
<h1> Welcome to Templates</h1>

```

6. Open views.py

```
from django.shortcuts import render
# Create your views here.
def hello(request):
  return render(request,'firstTemplate/result.html')
```

7. Open urls.py

```
from django.contrib import admin
from django.urls import path
from firstTemplate import views as v1


urlpatterns = [
    path('admin/', admin.site.urls),path('hello',v1.hello),path('date',v2.da
]
```

8. Start server
9. Type url in browser

Template tags

- we can include dynamic content to the template file by using template tag or template variable.

1. Create new app - firstapp
2. Go to firstapp/views.py

```
firstTemplate > views.py
1  from django.shortcuts import render
2  # Create your views here.
3  def hello(request):
4      return render(request,'firstTemplate/result.html')
```

3. Configure ursl.py in project level

Template tags

Template tags are used to insert dynamic content into the template are called template tags or template variables.

1. Create app - dateapp
2. dateapp/views.py

Write an application to wish your friend based on the time like good morning, good afternoon and good evening.

views.py

```
from django.shortcuts import render
import datetime

# Create your views here.
def date_view(request):
  date=datetime.datetime.now()
  #send date object to template using dictionary object
  #my_dict={'current_date':date}
  msg="Hi "
  hours=int(date.strftime('%H'))
  if hours<12:
    msg+='Good Morning'
  elif hours<16:
    msg+='Good Afternoon'
  elif hours<21:
    msg+='Good Evening'
  else:
    msg+='Good Night have  a nice sleep'
  my_dict={'current_date':date,'send_msg':msg}
  return render(request,'dateapp/date.html',context=my_dict)
```

templat/dateapp/date.html

```
<!DOCTYPE html>


<body bgcolor="red">
  <h1>{{send_msg}}</h1>
  <h1>current date and time is : {{current_date}}</h1>
</body>
```

Adding static files in template like image/css files

1. create a folder ‘static’ in main project folder

```
STATIC_DIR=os.path.join(BASE_DIR,'static')
----
----

Goto
STATIC_DIR=os.path.join(BASE_DIR,’static’)

STATICFILES_DIR=[STATIC_DIR,]
```

2. Create folder image/css
3. Save your static files in above folder
4. Goto settings.py

Adding static files in template like image/css files

5. Use Template Tags to insert image At the beginning of HTML just after we have to include the following template tag

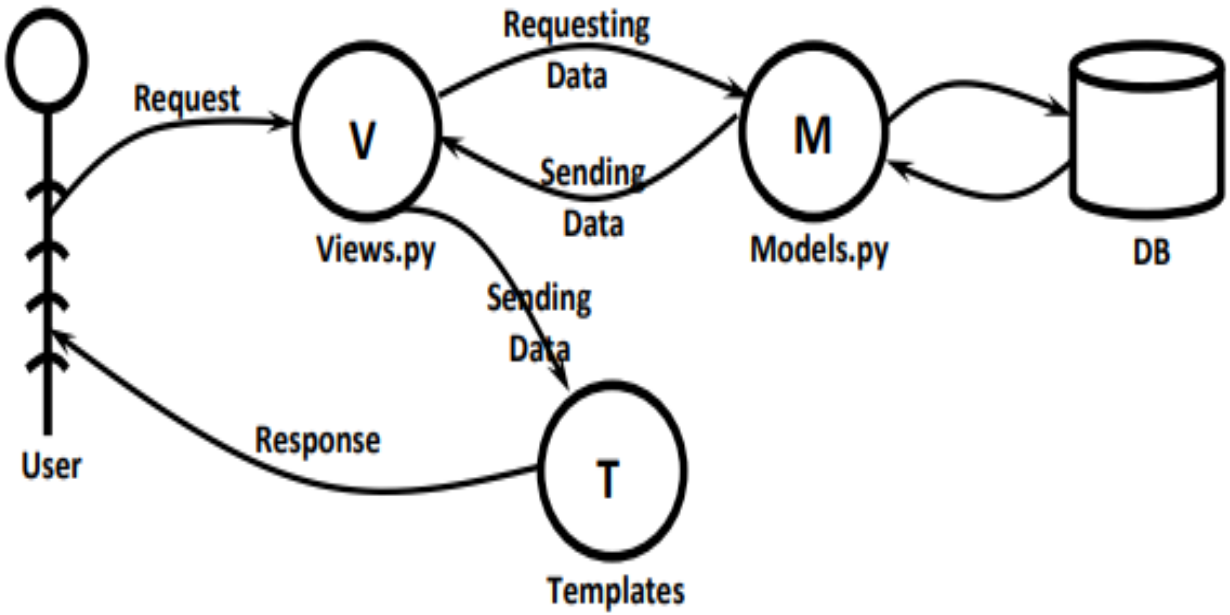
{% load staticfiles %}

include image

include css file

<link rel="stylesheet" href="{%static "css/demo.css"%}">

MVT


Requesting
Data
Request
V
M
Sending
Data
Views.py
Models.py
DB
Sending
Data
Response
User
T
Templates

Working with models and databases

- Django supports sqllite3, oracle, mysql, postgresql etc.
- Default database is sqllite3 provided by django
- How to configure database?

Settings.py

```
# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

check whether database configuration done properly or not

1. Goto command prompt

   python manage.py shell

2. from django.db import connection
3. c=connection.cursor()

if no errors then your database is configured properly.

Model class

- model class is python class that contains database information each model maps to database table.
- In django database tables need not be created.
- Model class contains fields and its behaviour information.
- Model class is child class of django.db.models.Model
- Model class is to be defined in models.py file

How to write model classes in models.py?

1. Create project and app
2. Projectapp/settings.py add your app name
3. Open models.py file

```
from django.db import models

# Create your models here.
class Student(models.Model):
    sno=models.IntegerField()
    sname=models.CharField(max_length=40)
    smarks=models.FloatField()
```

4. Convert this model class to sql code.

Convert model class into database specific SQL code

1. makemigrations : used to convert model class to SQL code.

   python manage.py makemigrations

2. # check SQL code

   python manage.py sqlmigrate testapp 0001

3. migrate: execute SQL code

   python manage.py migrate

How to check created database table in Django admin interface?

- # register model class in ‘admin.py’ file

```
from django.contrib import admin
from testapp.models import Student

# Register your models here.
admin.site.register(Student)
```

create superuser to login to admin interface

python manage.py createsuperuser

create username and password

run server

open browser: http://127.0.0.1:8000/admin

Django model class create database table with following fields

```python
from django.db import models

# Create your models here.
class Student(models.Model):
    sno=models.IntegerField()
    sname=models.CharField(max_length=40)
    smarks=models.FloatField()
```

sno
sname
smarks

Model class fields

ID
sno
sname
smarks

Data base table

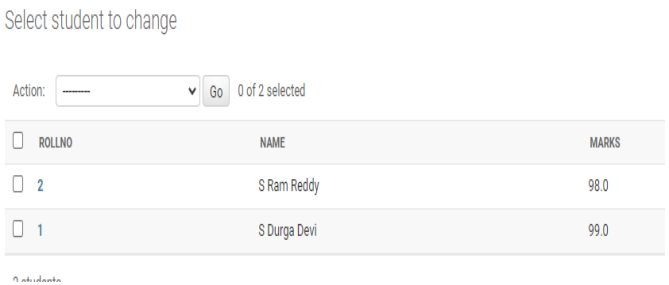
S. Durga Devi, CSE,CBIT

Display data in admin Interface in browser

- Following changes to be done

```
createDB > models.py
from django.db import models

# Create your models here.
class student(models.Model):
    rollno=models.IntegerField()
    name=models.CharField(max_length=30)
    marks=models.FloatField()
def __str__(self):
    return self.name
```




```
createDB > admin.py
from django.contrib import admin
from createDB.models import student

# Register your models here.
class studentAdmin(admin.ModelAdmin):
    list_display=['rollno','name','marks']
admin.site.register(student,studentAdmin)
```

Display database tables to end user

1. Create project and app
2. Add your app in settings.py
3. Create model class in models.py (student)
4. Convert model class into sql code using make migrations command
5. Execute sql code using migrate
6. Create folder templates and configure in settings.py
7. Go to views.py
8. Create html file in folder templates/app/results.html

7. views.py

Employees > views.py

```python
from django.shortcuts import render
from Employees.models import Employee

# Create your views here.
def employeedata(request):
    emp_list=Employee.objects.all()
    my_dict={'emp_list':emp_list}
    return render(request,'Employees/results.html',context=my_dict)

```

8. templates/Employee/results.html

templates > Employees > results.html > html >

```
8   <body>
9     <h1>Employees details</h1>
10    {% if emp_list %}
11    <table border="2">
12      <thead>
13        <th>empno</th>
14        <th>empname</th>
15        <th>empsalary</th>
16      </thead>
17      {% for emp in emp_list %}
18      <tr>
19        <td>{{emp.empid}}</td>
20        <td>{{emp.empname}}</td>
21        <td>{{emp.empsalary}}</td>
22      </tr>
23      {% endfor %}
24    </table>
25    {% else %}
26    <p>no records</p>
27    {% endif %}
28  </body>
```

9. Add url in urls.py

```
from django.contrib import admin
from django.urls import path
from Employees import views

urlpatterns = [
    path('admin/', admin.site.urls),path('details',views.employeedata)
,]
```

10. Start server

Working with MySql database

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'employeedb',
        'USERNAME':'root'
        'PASSWORD':'root'
    }
}
```

Django Forms

S. Durga Devi, CSE,CBIT

references

https://buildmedia.readthedocs.org/media/pdf/django/4.0.x/django.pdf

Download django documentation.