**Django Framework**

1. It is a back end web framework.
2. Freeware and open source
3. Completely developed by using python
4. It is maintained by Django Software Foundation(DSF)

Official website: <https://www.djangoproject.com/>

Documentation:<https://buildmedia.readthedocs.org/media/pdf/django/3.1.x/django.pdf>

Generally every web application is developed using MVC (model view controller), Django follows MVT(Model View Template) design pattern.

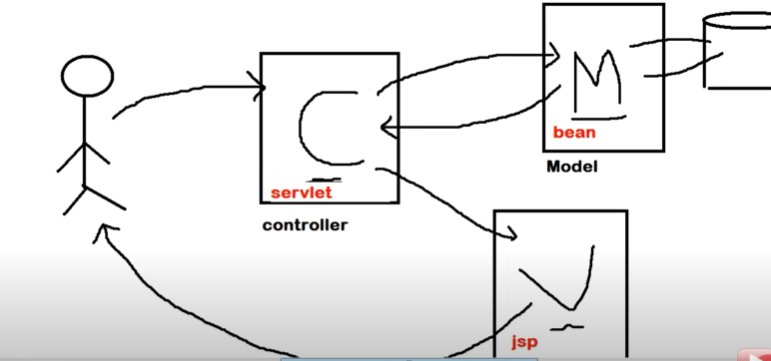
**Difference between MVC and MVT approach**:

**MVC** :

M:Model(Business logic)

V:View(Presentation logic)

C:Controller(Controlling activities)

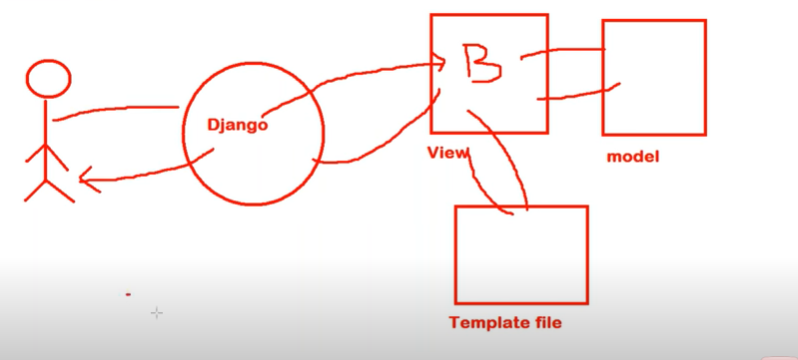


**MVT :**

M:Model(Database)

V:View(Business logic)

T:Template(Presentation logic)



**Top 5 features of Django:**

* Fast
* Fully loaded(Authentication, security, session management)
* Security(CSRF attack-Cross site request forgery)
* Scalability
* Versatile It can be used in any type of application.

**Django Project v/s Application:**

---Application is used to perform certain action

---Collection of applications is known as Project

Project=several applications+project configuration

**How to create django project**:

1. Open command prompt
2. Enter into working directory
3. To start project type this command :: django-admin startproject projectname
4. Project will be launched in IDE.

**Whenever we create a project this files will be automatically created:**

Firstproject

manage.py

firstproject

\_\_init\_\_.py

settings.py

urls.py

wsgi.py

**Steps to create an application**:

1. Enter into project
2. py manage.py startapp testapp

**Whenever we create an application this files will be automatically created:**

testapp

migrations

\_\_init\_\_.py

views.py

models.py

admin.py

apps.py

test.py

**Command to start an application:**

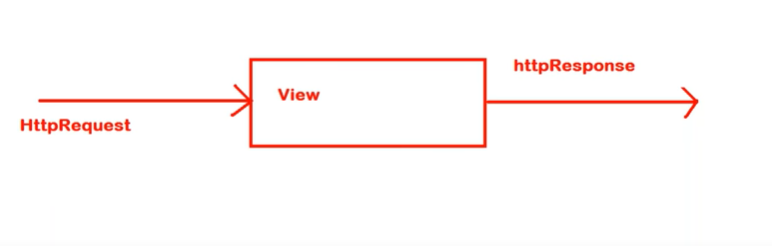
py manage.py startapp testapp

py manage.py runserver 8999

**Procedure:**

When a client sends a request to a server, the server sends a request to the application. In the application view is responsible for providing required functionality.

View is a function that takes httpRequest and provides httpResponse.



**View:** Used for business logic.

**Types of views:**

1. **Function based views:**

1. **Class based views:**

**Steps to execute an application**:

1. Start project.
2. Start application.
3. Add this application to project in settings.py file
4. Define view function in views.py.
5. Define url pattern for our view function inside urls.py
6. Run server.
7. Send a request.

In single file we can have any number of applications and urls

urls---Project level

**How to define urls at application level instead of project level:**

1. Create a separate urls.py file at application level.
2. Include this application level urls.py to project level urls.py.

**How to copy one application to another?**

1. Select the application and click on copy
2. Create new application
3. Paste in new application
4. Add application in settings.py.
5. Include url in urls.py

**Template:** Template is an html file, used for presentation.

**Steps to add templates in application:**

1. Start project.
2. Start application.
3. Create a template folder in our main project folder.

In that template folder create a separate folder with application name in that folder we can create html files(template files).

1. Add this application to project in settings.py file

Add templates folder in settings.py

1. Define view function in views.py.
2. Define url pattern for our view function inside urls.py
3. Run server.
4. Send a request.

Firstproject

templates

testapp1

\*. html files

testapp2

\*. html files

manage.py

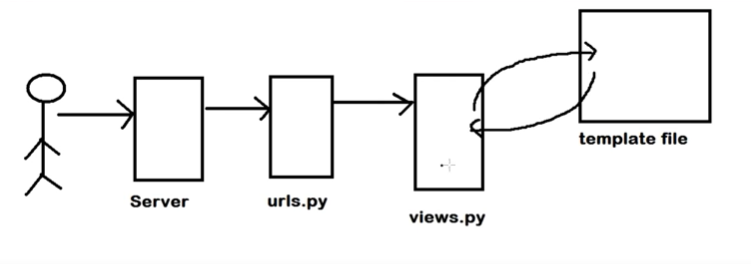
firstproject

\_\_init\_\_.py

settings.py

urls.py

wsgi.py



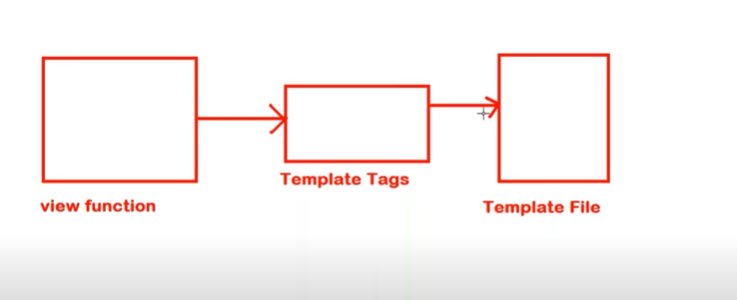
**How to inject Dynamic content from view function to template file:**

This can be done by using template tags or template variables.

1. Import required libraries in views.py
2. Add dynamic content in the form of a dictionary.
3. Add context as a parameter while returning a response.
4. Retrieve dynamic data from views.py using key name and access it using double curly braces.

{{templateTag}}-->Basic tag

{% %}--->Advanced template tag(These are known as jinja2 template tags)



**How to use css/image/javascript files in template files:**

These all files are known as static files.

**Working with Static files:**

1. Create a static folder.

Create another folder named with images/css inside the static folder.

1. Add application in settings.py.

Add template file path.

--- TEMPLATE\_DIR=PurePosixPath(BASE\_DIR).joinpath('templates')

Add a static file path.

--- STATIC\_DIR=PurePosixPath(BASE\_DIR).joinpath('static')

Add below code at bottom of settings.py file.

--- STATICFILES\_DIRS=[

STATIC\_DIR,

]

1. Add {%load static %} before html tag.
2. Access files using <img src="{%static "images/12.png" %}">

**Models:**

Django provides an in-built database.

Sqlite3 is the default database, it supports small to medium scale applications.

**Configurations of database:**

Database configurations are not required for the default database.

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

**How to check whether a database is configured correctly or not?**

1. Py manage.py shell
2. In interactive console

from django.db import connection

c=connection.cursor()

If these three commands are working correctly then the database is said to be configured correctly, if not errors will be triggered.

**How to create a table in python?**

For creation of table sql queries are not required. Class should be written in models.py then that class will be converted into table.

For each table one model class should be created.

**How to create a class in models.py?**

class Employee(models.Model):

enum=models.IntegerField()

ename=models.CharField(max\_length=64)

esal=models.FloatField()

eaddress=models.CharField(max\_length=64)

table\_name=appname\_classname(Ex: modelapp\_employee)

fields=enum,ename,esal,eaddress

behavior= enum(Int)

ename(Strings)

esal(Float)

eaddress(String)

There are 2 activities performed to execute python code:

1. makemigrations-->Used to convert python code to sql.

py manage.py makemigrations

1. migrate-->Used to execute sql code.

python manage.py sqlmigrate appname createdfilename

ex: python manage.py sqlmigrate modelapp 0001

Python manage.py migrate is used to migrate all the applications in a project.

**In admin.py:**

from django.contrib import admin

from modelapp.models import Employee

admin.site.register(Employee)

**How to launch admin interface:**

In the admin interface we can check our created table.

We can perform CRUD operations on table.

1. 127.0.0.1/admin

2. We require a username and password.

3. Create superuser:

py manage.py createsuperuser

4. Import model class in admin.py

from modelapp.models import Employee

admin.site.register(Employee)

**How to generate fake data?**

Fake data can be generated by using faker module

Code:

pip install faker

