

DEVELOPING A SIMPLE, FAST AND POWERFUL WEB DEVELOPMENT USING DJANGO

Dr Krishnakumar A, Assistant Professor, Department of Computer Science,
Nallamuthu Gounder Mahalingam College, Pollachi, Tamil Nadu, India. krishna2c@gmail.com
Mr.E.Vengatesh, Student, Department of Computer Science, Nallamuthu Gounder Mahalingam
College, Pollachi, Tamil Nadu, India.vengatesheswaran26@gmail.com

ABSTRACT

This Research paper studies about a python-based web framework known as Django. An open source framework follows the basic Model View Template structure with some modifications. These modifications are explained in this paper and it helps to know the use Django over other web frameworks that are available in the industry. In addition, how to install and create a basic project using this framework also detailed in this paper. This paper also studies about the different modules that are available in this framework, which helps to understand about the MVT structure of this framework very briefly. This paper helps to know how to create apps inside the framework and add them to our main project and how to create forms in views to interact with databases and how it performs operations on databases. It discusses about how to make dynamic websites and tasks easily through centralizing access to the apps. This paper covers everything required for a person or a student to get started with the Django framework and to learn the basics to create some simple projects related to the Django web framework and make learning interactive and easy even for a lame person.

KEYWORDS :Django, Web Development, Python Framework.

Introduction

Django is a web application framework which is an open source and written in Python language. It uses MVT design structure (MVT stands for Model View Template). Due to its rapid development feature, Django is very demanding in the current software market. It takes less time to build any kind of application and this Model View Template framework will work based upon the model as a database and view as a controlling functionality. Template will work as a user side for communication interaction. The Django model will work as database management and two main commands like *python manage.py make migrations* deduct the changes in models.py file and ready to send data into the sqlite3 (default database). Through the command *python manage.py migrate* the Django system will save all changes in its database system and through one more command *Python manage.py runserver* at the end to start our project and gives us the localhost address for the project to run locally. views.py file will handle the request for the project to the API's call to template management in requests.

History

In 2003 by Lawrence, Django was designed, developed, and released to the open public under BSD license in 2005. Currently, Django Software Foundation takes care of maintenance and new releases.

Django is widely accepted and used by various well-known sites such as:

Spotify

1. Youtube
2. Dropbox
3. Pinterest
4. NASA

Installation & Creation of New Project

1. To install Django in our local environment meaning python & pip need to be installed if it not installed already. After python & pip is installed run pip3 install Django in the terminal to install Django.
2. After installing Django, now move to the next step i.e. creating a new project.
 1. Run `django-admin startproject projectName` to create a number of starter files for our project.
 2. Run `cd projectName` to navigate into the new project folder.
 3. To start the server we issue this command in terminal `python manage.py runserver` to start a local server in your system.
 4. In your browser Visit the url `http://localhost:8000/` to see the default page.

Why Django

1. Open-source means Free
2. Faster Development
3. Completely Scalable
4. Security is priority
5. Built in Administration portal

MVT Structure of Django

To know the MVT Structure of Django firstly we need to know what is MVT structure. The full form of MVT is Model View Template. MVT Structure has three parts 1. Model 2. View 3. Template.

Model: This part of the MVC structure acts as a medium for storing data from the user into the database. This is responsible for handling the logical part of the web application as well as how the data is stored in the database.

Views: This is a user interface. It is responsible for displaying data from databases and storing information provided by the user. In Django views are not the same as they are in basic MVC structure.

Controller: This part in MVC is responsible for the whole logic and workings behind the web application. When a user raises an HTTP request, the controller receives the request and sends back the appropriate response.

Hence Django implements a different kind of MVT architecture.

Creating App in Django

Method 1:

To create an app, we need to move inside the project directory through the terminal and type and execute this command:

`python manage.py startapp <APPNAME>`

Method 2:

To create an app, we will move inside the project directory through terminal and type and execute the following command:

`django-admin startapp app_name`

Now it creates directory structure of the app folder.

To make that app run mention it under `INSTALLED_APPS` which is located inside `setting.py` inside the `proj1` directory.

After completing the above steps, finally create the app and to render it specify the URL to the app so that the app is inside the main project and the URL provided will be redirected to that app. To do so follow the given steps:

Move to `project_directory` -> `project_directory` -> `urls.py` and add this code inside the header of the file.

from django.urls import include

Now inside the code of URL patterns specify the name of the app for URL of the app created. Below is the sample code to do so:

```
urlpatterns = [  
    path('main/', main.site.urls)  path("", include("proj1_name.urls")),  
]
```

Now to use the basic MVT model to create models, views, templates, and URLs inside the app and they will be added to the main project automatically.

That is the end of this part and now move to the next part called Django model. The model is defined as `Models.py` file. The file may contain multiple models.

Django Model

Django Model Provides a database-abstraction API that allows to create, retrieve, update and delete records from a map. Contains important fields and behavior for data you store. Typically, each model maps in a data table.

Django Model is a single SQL Database used with Django. Models make the task easier and organize tables into models. Generally, the maps for each model are in the same data table and Django models provide compatibility, simplicity, version control, and advanced metadata management.

```
from django.db import models  
class c1 (models.Model):  
    n1 =  
    models.CharField(max_length=30) =  
    n2  
    models.CharField(max_length=30)
```

Django CRUD

CRUD means Create, Read, Update, and Delete

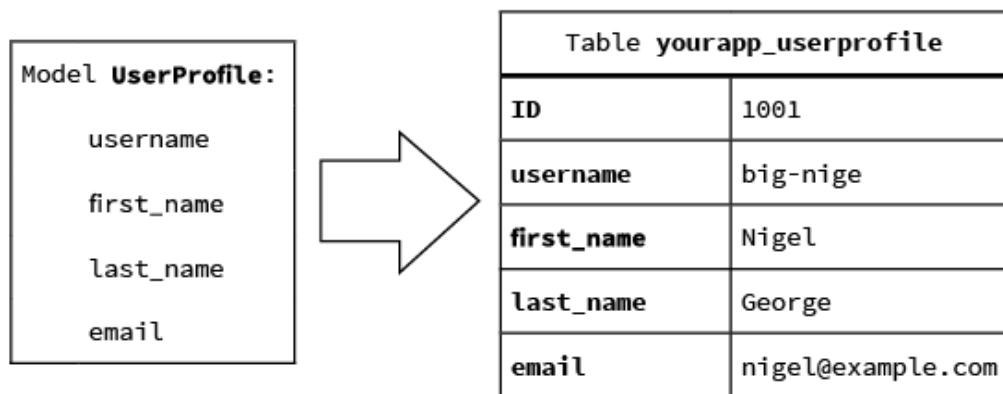
Django allows sharing its previous data types: - add, delete, modify and query items, using a database-generating database called ORM (Object Relational Mapper).

One can access Django ORM by using the following command within the project guide. Most general information is organized by some type of SQL, but each database uses SQL in its own way. SQL can also be difficult and difficult to learn. The ORM tool simplifies the database system by providing a

simple map between an object ('O' in ORM) and a basic database. This means that the producer does not need to know the data structure, nor does it require complex SQL to manipulate and retrieve data.

ORM Allows Easy Data Usage without Writing Complex SQL

In Django, the model is something that is specified in the database. When making a model, Django uses SQL to create a corresponding table in the database without having to write a single line of SQL. Django names the table and the name of your Django request. The model also links related information to a database.



Creating a Model

Following is a real model created as an example.

This category has 4 attributes (3 CharField and 1 Integer), which will be table fields.

```
from django.db import
models class
real(models.Model):
    web=
        models.CharField(max_length =
50)    n1 =
        models.CharField(max_length
= 50)    n2 =
        models.CharField(max_length
= 50)    n3 =
        models.IntegerField()
class Meta:
    db_table = "example1"
```

Django Views

Django views are a set of functions classes. Everything is contained inside the views.py enter in the app directory. A view is a user interface that is seen in a browser when the site is rendered.

Definition as per Django Documentation: “A view function is a python function that takes a web request and returns a web response”. The forms of responses can be HTML web pages, XML documents, images or a 404 error. Functions and logic proceed each time when a different URL is visited.

A simple view function:

```
from django.http import HttpResponse

import library_name

def function_name(request):

n1=library_name.library_name.request_name(:

    html="<html><body>Some    Random
text according to need
    %s.</body></html>"%n1
return HttpResponse(html)
```

Let's go through the steps:

1. Line1 is for importing the class from the Django http module.
2. Line2 is for importing the python datetime library.
3. Line 3 is for defining a function called cur_datetime. This is exactly how a view function looks like.
4. Line 5 to 7 is for the structure of a web page and the view is returning a response as an HttpResponse.

Django has two types of views

1. Function Based Views
2. Class Based Views

Function-based views: A function-based view, is a python function that takes a web request and returns a web response. The form of response can be a HTML content, XML document, 404 error, etc. All view functions takes a HttpRequest object as its first parameter.

```
def function_name(request):

    return
    HttpResponse('html/variable/text')
```

In Django, the user can work on many files at the time so it is important to notice that in which file the user is writing the code.

view.py file of the application is to write functions that contain the business logic of the application, later it is required to define the URL name for the functions in the *urls.py* file of the project.

Class-based views: It provides a different way to execute views as python objects rather than the functions. They do not replace function-based views but it's had certain differences and advantages.

A code that handles HTTP GET in a class- based view:

```
from django.http import HttpResponse
```

```
from django.views import View
```

```
class new_view(View):
```

```
def get(self,request):  
  
    return HttpResponseRedirect('res')
```

Django Templates

Django provides an easy way to create powerful HTML using its template program. The Django templates are commonly created using HTML, CSS, and Javascript. Django template manages well and produces HTML pages that are visible to the end user. Django works a lot with the background in order to give the structure of any website. For these purpose templates are generally used. Template function takes three parameters -

1. **Request-** Initial Request.
2. **The Path to create templates** - There is the `TEMPLATE_DIRS` option related to the *project.py* variables that are changing.
3. **Parameters Dictionary** - A dictionary that contains each element, which is required for the template. We can use `local people ()` to transfer all local variables announced in the view or one can create own variable as per needs.

Django Template Language (DTL)

It provides a small language to define the front-end part of the program that a user faces generally. A Django template is a Python thread used with the language of the Django template and there are some constructions known and translated by a template engine.variable variables and tags. The template is provided in context.offers that are flexible in their values, looked up in context, and then tagged.

Django is a framework that enable to separate python, HTML, the python part goes inside the view, and the HTML part goes inside templates. Linking the two, Django depends on dedicated performance and the language of the Django template.

Tags:

Tags allow working on following tasks: if status, loop, template asset, and many more can performed later.

Tag for:

Like 'if', we have the 'for' tag, which works in the same way as Python. Let us change our mindset so we can move the list to our template.

```
def hello (request):  
    n1 = library_name.library_name.calling(). Internal  
    function ()  
    samplearray =  
        ['1', '2', '3', '4', '5', '6', '7']  
    roll back (  
        request, "m1.html",  
        {"n1": n1,  
         "calling_array":samplearray})
```

HTML is a static markup language and due to this reason, one cannot write python code inside the HTML because a browser cannot understand the Python code.

Using Django Templates

Templates not only display static data but also display data from different information linked to the dictionary.

```
<!DOCTYPE html>
<html lang = "en">
<head>
    <meta charset = "UTF-8">
    <meta name = "viewport">
    <title> sample template </title>
</head>
<body>
    <h1> Some title</h1>
    <p> Data is here {{data}} </p>

</body>
</html>
```

Django Forms

HTML forms are the basic component of modern websites. It is the primary source of collecting information from website visitors and users. Django comes with a Form class that is used to create HTML forms. We can do all the work from Django forms with the advanced HTML, but Django makes it easier and more efficient for the end user, especially in form validation part. Once you enjoy working with Django forms, you will just forget about HTML forms.

Django completes three distinct parts in the work related to forms. It is preparing and reorganizing data so that it is ready for rendering. It creates HTML forms of data. It also receives and process submitted forms and customer data.

Create Django Forms using Form Class

Let's create a Django form using form class, to create a new file inside the application folder using the file name forms.py. Now we can write the below code inside forms.py to create a django form.

```
from django import forms

class sample_form(forms.Form):

    id1=forms.CharField()

    #here length is not required

    id2=forms.CharField()
```

After that, when this form is rendering its look like this

```
<tr>
  <th>
    <label for="id1">label:
  </label>
  </th>

  <td>
    <input          type="text"
name="id1" id="id1">
    </td>
</tr>                                     required
```

```
<tr>
  <th>
    <label for="id2"> id2 :</label>
  </th>

  <td>
    <input type="text" name="id2"          required id="id2" >
    </td>
</tr>
```

Display Form to User

To display the form to the user, create an object of the Form class in `views.py` then pass the object to template files and use the Form object in the template file.

```
from django.shortcuts import render

from .forms import sample_form

def showformdata(request):

    fm=sample_form()
    return

    render(request,"enroll/formdata.html",{ 'form':fm})
```

After that, passing the context form into formdata template and its look like the below code:
`//templates/enroll/formdata.html`

```
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">    <meta name="viewport"
content="width=device-width,
  initial-scale=1.0">
  <title>FormData</title>
</head>

<body>
<form action="" method="get">
  <table>
```

```
{{form}}  
</table>  
<input type="submit" value="Submit"/>  
</body> </html>
```

Now, provide the URL in urls.py file.

```
from django.urls import  
path from . import views  
urlpatterns=[  
    path('new/',views.showformdata),  
  
]
```

Now, Include the above url in the main urls.py file in the inner project folder.

```
from django.contrib import admin  
  
Form django.urls import path, include  
  
urlpatterns=[    path(.,main/.,admin.site.urls),  
  
    path(.,new/.,include(enroll.urls))  
  
]
```

Finally, It's time to save the all code and run the server and access the form to achieve this type

Command: *python manage.py runserver*

It will produce the following output-



The screenshot shows a web browser displaying a form. It has two input fields: one labeled 'Name:' and another labeled 'Email:'. Below these fields is a button labeled 'Submit'.

Conclusion:

Django is a Python-based web application framework that is free and open source. A framework is simply a collection of modules that facilitate development. They're grouped together and allow you to build apps or websites from scratch rather than starting from scratch."Rapid development and clean, pragmatic design" are key benefits of Django. When installed on a web server, the Django web framework can assist developers in quickly creating a feature-rich, secure, and scalable web frontend.^[ka1]

Django's architecture is based on the MVT framework, which stands for MODEL, VIEW, and TEMPLATE. MVT is an alternative to the MVC framework, consisting of three components: Model, View, and Controller.

The main difference between MVT and MVC is that the Django architecture handles the responsibilities that the MVC architecture's controller part handles. All of the controller-related tasks are handled by Django's templates. To put it another way, the template's contents are a mix of Django Template Language (also known as DTL) and HTML (HyperText Markup Language). Django is an excellent choice for projects that must deal with large amounts of content (e.g., media files), user interactions, high traffic, and complex functions or technology (e.g., machine learning). It is, however,

simple enough for smaller projects or if you plan to scale your project to a much larger scale. That is why many businesses use Django with varying sizes and objectives.

If you are interested in making your career as a full-stack developer, you can explore our Python Certification Course. The course will cover the basics of Python along with the Django framework.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

- [1] Shyam, A., & Mukesh, N. (2020). A django based educational resource sharing website: shreic. *Journal of scientific research*, 64(1), 138-152.
- [2] Juneau, J., Baker, J., Wierzbicki, F., Muoz, L. S., Ng, V., Ng, A., & Baker, D. L. (2010). *The definitive guide to Jython: Python for the Java platform*. Apress.
- [3] Chou, J., Chen, L., Ding, H., Tu, J., & Xu, B. (2013, November). A method of optimizing Django based on greedy strategy. In *2013 10th web information system and application conference* (pp. 176-179). IEEE.
- [4] Rosenbloom, A. (2018, May). A simple MVC framework for web development courses. In *Proceedings of the 23rd western canadian conference on computing education* (pp. 1-3).
- [5] Gore, H., Singh, R. K., Singh, A., Singh, A. P., Shabaz, M., Singh, B. K., & Jagota, V. (2021). Django: Web development simple & fast. *Annals of the Romanian Society for Cell Biology*, 25(6), 4576-4585.
- [6] Aniche, M., Bavota, G., Treude, C., Gerosa, M. A., & van Deursen, A. (2018). Code smells for model-view-controller architectures. *Empirical Software Engineering*, 23, 2121-2157.
- [7] Forcier, J., Bissex, P., & Chun, W. J. (2008). *Python web development with Django*. Addison-Wesley Professional.
- [8] Vincent, W. S. (2021). *Django for Beginners: Build websites with Python and Django*. WelcomeToCode.
- [9] George, N. (2020). *Build a Website With Django 3: A complete introduction to Django 3*. GNW Independent Publishing.
- [10] Vincent, W. S. (2022). *Django for APIs: Build web APIs with Python and Django*. WelcomeToCode.
- [11] Shabaz, M., & Kumar, A. (2019). SA sorting: a novel sorting technique for large-scale data. *Journal of Computer Networks and Communications*, 2019.
- [12] Rakhra, M., Singh, R., Lohani, T. K., & Shabaz, M. (2021). Metaheuristic and machine learning-based smart engine for renting and sharing of agriculture equipment. *Mathematical Problems in Engineering*, 2021, 1-13.
- [13] Rattan, P., Arora, M., Rakhra, M., & Goel, V. (2021). A neoteric approach of prioritizing regression test suites using hybrid ESDG models. *Annals of the Romanian Society for Cell Biology*, 2965-2973.
- [14] Arora, M., Chopra, S., Rakhra, M., Minhas, V., Walia, R., Aggarwal, R., ... & Shabaz, M. (2021). Agile Umbrella Methodologies and its Global Impact. *Annals of the Romanian Society for Cell Biology*, 2990-3003.
- [15] Mitra, D., Arora, M., Rakhra, M., Kumar, C. R., Reddy, M. L., Reddy, S. P. K., ... & Shabaz, M. (2021). A Hybrid Framework to Control Software Architecture Erosion for Addressing Maintenance Issues. *Annals of the Romanian Society for Cell Biology*, 2974-2989.
- [16] Panesar, G. S., Venkatesh, D., Rakhra, M., Jairath, K., & Shabaz, M. (2021). Agile software and business development using artificial intelligence. *Annals of the Romanian Society for Cell Biology*, 1851-1857.