IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Django Web Development Simple & Fast

[1]Rakesh Kumar Singh, [2]Himanshu Gore, [3]Ashutosh Singh, [4]Arnav Pratap Singh
[1]Student, [2]Student, [3]Student, [4]Student
[1]Lovely Professional University ,
[2]Lovely Professional University ,
[3]Lovely Professional University ,
[4]Lovely Professional University

Abstract

This Research paper studies about a python based web framework known as Django. It is an open source framework which follows the basic Model View Template structure with some modifications which are explained inside the paper and it also help us to know why we use Django over other web frameworks that are available in the industry and how we can install it on our system and create a basic project using this framework following this our paper also studies about the different modules that are available to us this also helps us understand the MVT structure of this framework very briefly. When we go deep inside the paper we also get to know how we can create apps inside the framework and add them to our main project and how we can create form inside views how it interacts with databases and how it performs operations on databases how it make easy making dynamic websites and making task easy by centralizing access to the apps. Basically this paper covers everything required for a person or a student to get started with the django framework and 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.

Introduction

Django is a web application framework which is open source and written in the 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 Market. It takes less time to build any kind of application. Why we say this Model View Template because this framework will work based upon the model as a database and view as a controlling functionality and template will work as a user side for communication interaction.

The Django model will work as database management, we use two main commands like:- python manage.py makemigrations Django will deduct the changes in models.py file and ready to send data into the sqlite3 (choose any database). Then we make python manage.py migrate. then the Django system will save all changes in his database system.

Then we make one more command Python manage.py run server at the end this will start our project and gives us the localhost address for the project running locally. And views.py file will handle the request for the project to the API's call to template management in requests. we can write the views in the form of python functions.

History

In 2003 by Lawrence, Django was designed and 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:

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

Installation & Creation of new project

1. We need to install Django in our local environment meaning python & pip need to be installed if you haven't already. After python & pip is installed run pip3 install Django in the terminal to install Django.
2. After installing Django we 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 we will get the directory structure of the app folder.

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

After completing the above steps we have finally created our app but to render it we need to specify the URL to our app so that the app is inside our main project and the URL we provided will be redirected to that app. To do so we need to follow the given steps :

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

from django.urls import include

Now inside the code of URL patterns we need to specify the name of app for URL of the app we created below is the sample code to do so :

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

Now we can 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 we will move to the next part.

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 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. 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

Create, Read, Updating, and Deleting Data.

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

We can access Django ORM by using the following command within our 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.

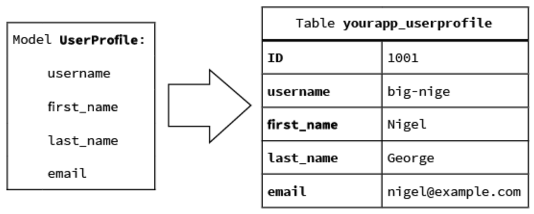


Table yourapp_userprofile	
ID	1001
username	big-nige
first_name	Nigel
last_name	George
email	nigel@example.com

Creating a Model

Following is a real model created as an example.

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

```
    models.IntegerField()

     class Meta:

     db_table = "examplel"
```

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 which we see in a browser when we are actually rendering a website.

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_n
ame(:

        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 take an HttpRequest object as its first parameter.

```
def function_name(request):

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

In Django, we are working on many files at the time so it is important to notice that in which file we write the code.

We use the view.py file of the application 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 execution of views as python objects rather than the functions. They do not replace function-based views but it's have certain differences and advantages.

A code that handle 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  HttpResponse('res')
```

Django Templates

Django provides an easy way to create powerful HTML using its template program. The Django templates are commanly 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 endings, And in order to give the structure of any website, For these purposes we use templates.
Template function basically 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 which contains each and every element which is required for the template. We can use local people () to transfer all local variables announced in the view or we can create our own variable.

Django Template(DTL) Language

It provides a small language to define the front end part of the program which a user faces.

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 are flexible in their values, looked up in context, and then tagged.

Django is a framework which enables us to separate python and 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 us to work on following tasks: if status, loop, template asset, and many more are performed.

Tag for:

Like 'if', we have the 'for' tag, which works in the same way as Python. Let's 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})
```

As we know HTML is a static markup language and due to this reason we cannot write python code inside our HTML because a browser cannot understand that code because Python is a programming language.

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 more easier and efficient for you, especially the 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 data and reorganizing data so that it is ready for rendering. It creates HTML forms of data. It also receives and processes submitted forms and customer data.

Create Django Forms using Form Class

Let's create a Django form using form class, to achieve this we have to create a new file inside the application folder let's say the file name is 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"
required id="id1">
    </td>
</tr>
```

```
<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 we have to 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=id1()
    return
render(request,"enroll/formdata.html",{'for
m':fm})
```

After that, passing the context form into formdata template its look like this:

//templates/enroll/formdata.htm

```
<!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">
```

Now, provide the URL in urls.py file.

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

  ]
```

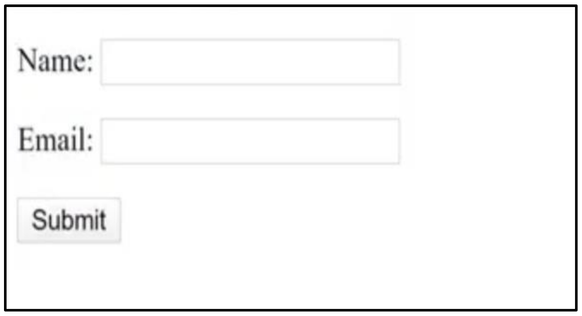
Now, Include the above url in the main urls.py file in the inner project folder.

```
from django.contrib import admin

Form django.urls import path, include

urlpatterns=[
       path('main/',admin.site.urls),

       path('new/',include(enroll.urls))
  ]
```

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

Command: `python mange.py runserver`

It will produce the following output-



Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

1. Adamya Shyam , Nitin Mukesh A Django Based Educational Resource Sharing Website: Shreic,Volume 64, Issue 1, 2020.
2. Josh JuneauJim BakerVictor NgLeo SotoFrank Wierzbicki,The Definitive Guide To Jython pp 281-325, Web Applications With Django
3. Jian Chou, Lin Chen Hui Ding; Jingxuan Tu, Baowen Xu,A Method of Optimizing Django Based on Greedy Strategy,2013 10th Web Information System and Application Conference
4. Arnold Rosenbloom,A Simple MVC Framework for Web Development Course,the 23rd Western Canadian Conference
5. Abhishek Bera,Shubham Darda, Ashish Gaikwad,Shivam Chanage,Prof. Suresh Rathod,Volume 6, Issue 5, May 2016, Endorseme - Professional Networking Using Django.
6. M. Aniche, G. Bavota, C. Treude, M.A. Gerosa and A. van Deursen, "Code smells for Model-View-Controller architectures", *Empir. Softw. Eng.*, vol. 23, no. 4, pp. 2121-2157, 2018.
7. Jeff Forcier, Paul Bissex, Wesley J Chun Python Web Development with Django
8. William S. Vincent Django for Beginners: Build websites with Python and Django Book
9. Nigel George Build a Website With Django 3: A complete introduction to Django
10. William S. Vincent Django for APIs: Build Web APIs with Python and Django