Dwarang - A Web based Python-Django Framework for Official Purposes

Srihari.S under the guidance of Dr.G.V.V.Sharma

1 Introduction

This project has been carried out under the guidance of Dr.G.V.V.Sharma (IIT-Hyderabad) during the Summer Research Internship undertaken in 2020(May-July), which is sponsored by Indian Academy Of Sciences. Dwarang is a software where we can display data onto a pdf forms of IITH where data is entered through html forms.

2 Installation

First clone the repository using git clone https://github.com/Srihari123456/Dwarang.git

2.1 Windows

1) Install python3.8
2) pip install virtualenv

2.2 Linux

1) sudo apt-get install python-pip
2) sudo pip install virtualenv

Then do pip install -r requirements.txt (Note: This installs Django and the required software.)

3 Running the project on Django local server

*3.1 **Type the below commands on console window***

1) cd dwarang_root

2) virtualenv env_dwarang (sets the virtual environment)

3) To activate the virtual environment:
   In windows: env_dwarang\scripts\activate
   In linux: source env_dwarang/bin/activate

4) python manage.py migrate (sets all the database tables)

5) python manage.py runserver (starts the local Django server)

6) Type the url 127.0.0.1:8080 in your browser

4 Structuring The Application Directory

--dwarang_root (folder)
- --db.sqlite3
- --manage.py
- --login (folder)
- --dashboard (folder)
- --reimbursement (folder)
- --ta_bill (folder)
- --screenshots (folder)
- --requirements.txt
- --contingent_exp (folder)
- --cert_a (folder)
- --telephone (folder)
  - --__pycache__ (folder)
  - --templates (folder)
  - --migrations (folder)
  - --__init__.py
  - --admin.py
  - --apps.py
  - --forms.py
  - --models.py
  - --tests.py
  - --urls.py
  - --views.py
- --dwarang_site (folder)
  - --__pycache__ (folder)
  - --templates (folder)
  - --static (folder)
  - --__init__.py
  - --asgi.py
  - --settings.py
  - --urls.py
  - --utils.py
  - --views.py

--wsgi.py

5 Uses of all files

1) __init__.py : Indicates that this directory is a python package.

2) models.py : Contains all the classes which will be converted into database tables.

3) forms.py : Used to render the forms to the user.

4) views.py : Here, all the objects from forms.py and models.py are imported here, and the actual functions are written here. Its the processing unit.

5) templates : All the templates are stored in this folder, which are required for the project.

6) db.sqlite3 : The database created when we ran the migrate command.

7) app.py : Runs the project on a django local server. It is the configuration file common to all the django apps.

8) manage.py : A command line utility for executing Django commands from within the project.

9) settings.py : Contains the configuration information for the project.

10) urls.py : Contains project level url declarations.

11) wsgi.py : Enables WSGI compatible webservers to serve the project.

12) migrations : The folder where django stores the migrations or changes to the database.

13) admin.py : Used to register the models with the django admin application.

14) tests.py : contains the test procedures which'll be run when testing the app.

15) static : Contains all the css,js and images required for the project.

16) utils.py : Contains the utility function required to render the html as pdf.

5.1 Understanding Code

We will understand the code for Telephone-Reimbursement form. The similar code is used for other forms.

*5.1.1 **models.py**:*

```
class Telephone_Reim(models.Model):
    name = models.CharField(max_length=120)
    amount = models . DecimalField
(decimal_places=2,max_digits=1000)
    Bank_name_branch = models.TextField()
```

Here,we create our model which is a data object that maps the apps data to the database. We define the fields for the model. These will have a corresponding field in the database table that django creates for the model. DecimalField() indicates that only a decimal value is valid for this field. CharField() and TextField() enables us to enter both characters and numbers.

*5.1.2 **admin.py**:*

```
from .models import Telephone_Reim
admin.site.register(Telephone_Reim)
```

Here,our model is registered with the admin. Try python manage.py createsuperuser to view inside the admin page.

*5.1.3 **apps.py**:*

```
class TelephoneReimConfig(AppConfig):
name = 'telephone_reim'
```

This contains a configuration class named after the app.

*5.1.4 **forms.py**:*

```
from .models import Telephone_Reim
class TelephoneForm(forms.ModelForm):
        name = forms.CharField ( label =
'Name' , widget = forms.TextInput ( attrs =
"placeholder" : "Your Name" ))
        Bank_name_branch = forms.CharField
( widget = forms.Textarea ( attrs =  "placeholder"
: "Bank's Name and Branch" , "rows" : 5 , 'cols' :
19 ))
        amount = forms.DecimalField ( initial =
0.00 )
        class Meta:
            model = Telephone_Reim
```

```
fields = ['name' ,
'Bank_name_branch' , 'amount' , ]
```

This contains a form which is to be rendered to the user based on the model created. Any alterations required can be specified.The fields which are to be displayed in the form are specified as a python list.

5.1.5 *urls.py:*

```
from .views import telephone_create_view ,
telephone_update_view , telephone_delete_view
, GeneratePDF
app_name = 'telephone'
urlpatterns = [
    path(' pdfsec/< str : tit >/< str : emp >/ ' ,
GeneratePDF , name = 'telephone-pdf' ),
    path(' create/ ' , telephone_create_view ,
name = 'telephone-create' ),
    path(' delete/< str : tit >/< str : emp >/ ' ,
telephone_delete_view , name =
'telephone-delete' ),
    path(' update/< str : tit >/< str : emp >/ ' ,
telephone_update_view , name =
'telephone-update' ),
]
```

This contains the url-configuration required for the app. We import the required views which are associated with each url. An optional name parameter can be used for quick reference to the url. Here < *str* : *tit* > is a path converter which converts the captured data into a string parameter and is passed to the corresponding view.

5.1.6 *views.py:*

```
@login_required(login_url="/login/create_acc/")
def telephone_create_view(request):
    form = TelephoneForm(request.POST or
None)
    if form.is_valid():
        form.save()
        form = TelephoneForm()
    context =
        'form': form

return render ( request ,
"telephone/telephone_create.html" , context )
```

The decorator @login_required(login_url="/login/create_acc/") ensures that without a successful login the user can't view the specified page, there-by ensuring authenticity of users. In this file, the functions are executed. Functions such as viewing the form, create a new form, update and delete the forms are executed here.
First, we import all the forms and models. We check whether the user is trying the get or post data into the html. If the request method is GET, then the form is passed on to the template and the template is viewed.
If the user is trying to post information into database, we check whether its a POST request, if it is, then we take the form data, and create a object for the corresponding class. We store that data into the object. Now, the object 'form' is saved in the database using the command form.save(), and is passed as a dictionary to the required html template.
For the views related to deletion, update and pdf-generation if a valid username and employee-id isn't provided, the user is redirected to login page using Django's redirect() function.

5.1.7 *telephone_create_form.html (Template):*

This html template consists of the form that browser submits the data to the database.

The sole purpose of these templates is to accept data from the browser and submit it to the database. After pressing the submit button the corresponding function in views.py file is executed, and the code checks for errors. If no error, data is stored as an entry into the database table.

5.1.8 *telephone_reim_form.html (Template):*

This html template consists of the form that is converted to pdf when required. The values to the corresponding form fields are inserted using django template tags, which are obtained using the views.py file.