**Django Tutorial Report**

**What is Django?**

It is a free and open-source web application framework, written in Python. It follows model-view- template (MVT) architectural pattern. Django's primary goal is to ease the creation of complex, database-driven websites. Some well-known sites that use Django include the Public Broadcasting Service, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket, and Nextdoor.

**Why Django?**

* It is fast and simple.
* It’s secure. It helps developers avoid common security issues, including clickjacking, cross-site scripting and SQL injection.
* It’s fully loaded with extras and scalable, so we can make applications that handle heavy traffic and large volumes of information.
* It works with most major databases and allows using a database that is more suitable in a particular project, or even multiple databases at the same time.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Installation (Setup)**

* Download Python from its website (<https://www.python.org>) according to your system and install it, select **Add to Path Variables** while installing. (Pip will be installed with Python by default)
* If we do not want to directly install Django in our system, we can use Virtual Environment.
* To install Virtual Environment, use this command in Command Prompt/Powershell **“pip install virtualenvwrapper-win”** to install it.
* To make a virtual environment we can use **“mkvirtualenv test”** command. (Here test is the environment name and it can be anything)
* Now to enter virtual environment we can use **“workon test”** command
* Now we can install Django in that environment by using **“pip install Django”** command.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Creation of Project and App**

* Open Command Prompt or PowerShell.
* Create a folder by using **“mkdir projects”** command, (projects is the folder name and it can be anything) and use **“cd projects”** to enter that folder in cmd/powershell.
* Now to create a project use **“django-admin startproject firstproject”** command. (Here firstproject is the project name and it can be anything)
* To create an app, we can use **“python manage.py startapp firstapp”** command where firstapp is the name of app.
* Now create **urls.py** file in app folder.

1. **Printing Hello World**

* Open urls.py file of app folder in that we have to write this code.

***from* django.urls *import* path**

***from* . *import* views**

**urlpatterns = [**

**path('', views.home, *name*='home')**

**]**

* Now open views.py and to print Hello World, we can use HttpResponse to return Hello World string. (Code should be like this)

***from* django.http.response *import* HttpResponse**

***from* django.shortcuts *import* render**

***# Create your views here.***

**def home(*request*):**

***return* HttpResponse("Hello World")**

* To make it work we have to include our apps urls.py file to projects urls.py to do that open project urls.py and add path to apps urls.py. (Code should be like this)

***from* django.contrib *import* admin**

***from* django.urls *import* path, include**

**urlpatterns = [**

**path('', include('firstapp.urls')),**

**path('admin/', admin.site.urls),**

**]**

* Now we included firstapp.urls file, if we load our site, it will display Hello World in left upper corner.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Template**

To use the template, we have to create templates folder and put html and css etc. files there to use it. Before we can use those files, we have to declare the path in settings.py dir section.

* Let’s print Hello Sandeep (a name) from html page.
* Create a **templates** folder outside app and project directory. (Name can be anything)
* Then create a html file **“home.html”** (home is the name of file) and put html code in there.
* After that open settings.py of project and then put the location of templates folder there. (It should look like this)

**'DIRS': [os.path.join(BASE\_DIR, 'templates')],**

* Now open views.py of projects and return render of that home.html (Code should look like this)

***from* django.shortcuts *import* render**

***# Create your views here.***

**def home(*request*):**

***return* render(*request*, 'home.html', {'name':'Sandeep'})**

* And write this code in html file

<h1>Hello {{name}} !!!</h1>

* Now if we refresh the page, we will get the name printed with Hello.

**Using the code of two or more html files**

* Lets create a base.html file in templates folder and we have our previous home.html file
* We can write the code of basic structure of site in base.html and we can use home.html for dynamic content. For example
* This is our base.html code

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta http-equiv="X-UA-Compatible" content="IE=edge">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>First Site</title>**

**</head>**

**<body bgcolor="cyan">**

**{% block content %}**

**{% endblock %}**

**</body>**

**</html>**

* We added the block content code and this is our home.html

**{% extends 'base.html' %}**

**{% block content %}**

**<h1>Hello {{name}} !!!</h1>**

**{% endblock %}**

* In our home.html we extended the home.html to base.html so base.html can use this code now whatever the code we will write in between block content block it will get extended to base.html block content block and we will be able to see the output of that code in our site page.

**Addition of two numbers**

**GET and POST**

* GET – When we are fetching a resource from server (for example image), we use GET method.
* POST – When we are sending the data, we use POST.
* To add two number, we will create a form in home.html so we can let user enter those two numbers. Form code will look like this

**<form action="add" method="POST">**

**{% csrf\_token%}**

**Enter 1st number: <input type="number" name="num1"><br>**

**Enter 2nd number: <input type="number" name="num2"><br>**

**<input type="submit">**

**</form>**

* Then we have to go urls.py of app and write the path for add function. We will have to define this function in views.py.

**path('add', views.add, *name*='add')**

* Open views.py and create the add function which will do this addition. It will look like this

**def add(*request*):**

**val1 = int(*request*.POST['num1'])**

**val2 = int(*request*.POST['num2'])**

**res = val1 + val2**

***return* render(*request*, 'result.html', {'result':res})**

* Now create a result.html file in templates folder and extend the base.html to it and print result. The code should look like this.

**{% extends 'base.html' %}**

**{% block content %}**

**Result: {{result}}**

**{% endblock %}**

**MVT Framework**

The Model View Template is a software design pattern. It is a collection of three important components Model View and Template.

* The Model helps to handle database. It is a data access layer which handles data.
* The Template is a presentation layer which handles User Interface part.
* The View is used to execute the business logic and interact with a model to carry data and render a template.

**Using HTML/CSS Template**

We will create a new app by using **“py manage.py startapp travello”** command for our new app, where we will be using html template.

* Download the travello template from this link <https://bit.ly/3niZizr>.
* Now lets copy the index.html of travello folder to templates folder of our project.
* To load that index.html we have to create urls.py in travello app after that we have to write this code.

***from* django.urls *import* path**

***from* django.urls.resolvers *import* URLPattern**

***from* . *import* views**

**urlpatterns = [**

**path("", views.index, *name*="index")**

**]**

* After that in views.py of travello app we have to fetch index.html file to do that we can write this code.

***from* django.shortcuts *import* render**

**def index(*request*):**

***return* render(*request*, 'index.html')**

* And we have to add travello.urls to projects urls.py and it will now load our travello app. (Project urls.py should look like this)

***from* django.contrib *import* admin**

***from* django.urls *import* path, include**

**urlpatterns = [**

**path('', include('travello.urls')),**

**path('admin/', admin.site.urls),**

**]**

* Now we can see after loading the site we can only see the executed html code, we are not getting images, css and javascript.

1. **Static Files**
   1. First of all create a static folder in your project directory (outside of app and project).
   2. Now copy the css, javascript, plugins and images folder in that static folder (Copy all the static files you need in your site there).
   3. And now we have to add the path to settings.py of project. Code should look like this

**STATIC\_URL = 'static/'**

**STATICFILES\_DIRS = [**

**os.path.join(BASE\_DIR, 'static')**

**]**

**STATIC\_ROOT = os.path.join(BASE\_DIR, 'assets')**

* 1. After this run **“py manage.py collectstatic”** command to get static files in assets folder.
  2. Now we can use these files in our index.html but we have to write the href url like this {% static ‘url’ %} and to make this work we have to use “{% load static %}” code in index.html file.

1. **Dynamic Files**
   1. To pass the dynamic data we have to create class in models.py of app, for example we are creating Destination class in models.py and in that class we will create some variable like id, name, dec, img and price. (Code should look like this)

***from* django.db *import* models**

**class Destination:**

**id : int**

**name : str**

**desc : str**

**img : str**

**price : int**

**offer : bool**

* 1. After that we will create the objects of this class in views.py of app and pass those objects to index.html. (Code will look like this)

***from* django.shortcuts *import* render**

***from* travello.models *import* Destination**

**def index(*request*):**

**dest1 = Destination()**

**dest1.name = "Mumbai"**

**dest1.img = 'destination\_1.jpg'**

**dest1.desc = "City That Never Sleeps"**

**dest1.price = 700**

**dest1.offer = False**

**dest2 = Destination()**

**dest2.name = "Delhi"**

**dest2.img = 'destination\_2.jpg'**

**dest2.desc = "Delhi Description"**

**dest2.price = 655**

**dest2.offer = True**

**dest3 = Destination()**

**dest3.name = "Kolkata"**

**dest3.img = 'destination\_3.jpg'**

**dest3.desc = "Kolkata Description"**

**dest3.price = 900**

**dest3.offer = False**

**dests = [dest1, dest2, dest3]**

***return* render(*request*, 'index.html', {'dests' : dests})**

* 1. Now we can use these objects in our index.html file of templates. Instead of passing different objects we passed one list object and used a for loop for html code so we can change the objects inside that code. We use conditional statement if to display Special Offer on certain location. (Code should look like this)

**{% for dest in dests %}**

***<!-- Destination -->***

**<div class="destination item">**

**<div class="destination\_image">**

**<img src="/static/images/{{dest.img}}" alt="">**

**{% if dest.offer %}**

**<div class="spec\_offer text-center"><a href="#">Special Offer</a></div>**

**{% endif %}**

**</div>**

**<div class="destination\_content">**

**<div class="destination\_title"><a href="destinations.html">{{dest.name}}</a></div>**

**<div class="destination\_subtitle"><p>{{dest.desc}}</p></div>**

**<div class="destination\_price">From ${{dest.price}}</div>**

**</div>**

**</div>**

**{% endfor %}**

**Working with Database**

* First of all, download and install Postgresql and Pgadmin.
* Open Pgadmin and select server Postgresql it will ask you for password enter the password you created while installing Postgresql and after that we will be connected with Postgres.
* Now to create new database click on Postgres and then right click on Databases and click create then database, we can name it whatever we want let’s name it mydatabase for now and click save, it will create the new database for us.
* Let’s connect our Django project with database, to do that we will open our project (which is firstproject) settings.py fille scroll down and find DATABASES and then we have to mention the name, credentials and host etc. (Code will look like this)

**DATABASES = {**

**'default': {**

**'ENGINE': 'django.db.backends.postgresql',**

**'NAME': 'mydatabase',**

**'USER': 'postgres',**

**'PASSWORD': 'root',**

**'HOST': 'localhost'**

**}**

**}**

* Now to connect Django with Postgresql we need some software/adapter and we will be using Psycopg2 and to install it we can run **“pip install psycopg2”** command.
* We have not mentioned our app in Django project settings.py, to do that open settings.py and scroll down and find INSTALLED\_APPS and mention the travello app there the code should look like this.

**INSTALLED\_APPS = [**

**'travello.apps.TravelloConfig',**

**]**

* Now we will open models.py of our app which is travello and we created a destination class there but that will not work now, so we need a model now which can create table in the database. (Code should look like this)

***from* django.db *import* models**

**class Destination(models.Model):**

**name = models.CharField(*max\_length*=100)**

**img = models.ImageField(*upload\_to*='pics')**

**desc = models.TextField()**

**price = models.IntegerField()**

**offer = models.BooleanField(*default*=False)**

* Now we will migrate this, but before that we have to make migration file, we can do that by using **“py manage.py makemigrations”** command. If our system don’t have Pillow library we will get a error to fix that install Pillow by using **“pip install Pillow”** command and run the makemigration command again.
* Now we will have a 0001\_initial.py migration file and to migrate it we will use “py manage.py sqlmigrate travello 0001” command and after that use “py manage.py migrate” command.
* Now this migration will create the table in database.

**Admin Panel**

* We can access admin panel using /admin after our site address in this case that’s localhost. We need username and password to sign in and to create that we have to use **“py manage.py createsuperuser**” command.
* After that command it will ask you to enter username, email and password, enter all the details and the superuser will be created, now we can use this username and password to login to Django admin panel.
* Now to add destination page there we have to register our destination page in admin.py of app, the code should look like this.

***from* django.contrib *import* admin**

***from* .models *import* Destination**

**admin.site.register(Destination)**

* After this if we go-to Django admin panel, we can see the destination there and now we can add data using it.
* Now if we click on add destination, we see there is a button to upload the image but we need to specify where that is going to get uploaded/saved to do that we open settings.py of project and scroll down to static files code and below that we will write the path. The code will look like this.

**MEDIA\_URL = 'media/'**

**MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')**

* Now we have to add the path to urls.py of project and the code should look like this.

***from* django.contrib *import* admin**

***from* django.urls *import* path, include**

***from* django.conf *import* settings**

***from* django.conf.urls.static *import* static**

**urlpatterns = [**

**path('', include('travello.urls')),**

**path('admin/', admin.site.urls),**

**]**

**urlpatterns = urlpatterns + static(settings.MEDIA\_URL, *document\_root*=settings.MEDIA\_ROOT)**

* After that we have to add the destination object in our views.py of app, the code should look like this

***from* django.shortcuts *import* render**

***from* travello.models *import* Destination**

**def index(*request*):**

**dests = Destination.objects.all()**

***return* render(*request*, 'index.html', {'dests' : dests})**

* Now we have to change the img src url of index.html to dest.img.url, like this

**<img src="{{dest.img.url}}" alt="">**

* Now we can see that it fetches the data from database.

**User Registration, Login and Logout**

* First of all we will create separate app/module for this, to do that we will use “py manage.py startapp accounts” command.
* After that create urls.py in accounts app and add mapping to register page. (Code should look like this)

***from* django.urls *import* path**

***from* . *import* views**

**urlpatterns = [**

**path('register', views.register, *name*='register')**

**]**

* Now we have to map this app (accounts) urls.py to project (firstproject) urls.py and the code will look like this.

**urlpatterns = [**

**path('', include('travello.urls')),**

**path('admin/', admin.site.urls),**

**path('accounts/', include('accounts.urls'))**

**]**

* After that we will add Register button in our site, for that we will add the code in index.html and it should look like this.

**<li><a href="accounts/register">Register</a></li>**

* Now we have to call that register page in views.py of accounts and the code will look like this.

***from* django.shortcuts *import* render**

**def register(*request*):**

***return* render(*request*, 'register.html')**

* We have not created register.html yet so to do that we will go to templates folder and create a register.html and write the code for registration form there. (Code will look like this)

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta http-equiv="X-UA-Compatible" content="IE=edge">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>Registration</title>**

**</head>**

**<body>**

**<form action="register" method="post">**

**{% csrf\_token %}**

**<input type="text" name="first\_name" placeholder="First Name"><br>**

**<input type="text" name="last\_name" placeholder="Last Name"><br>**

**<input type="text" name="username" placeholder="Username"><br>**

**<input type="email" name="email" placeholder="Email"><br>**

**<input type="password" name="password1" placeholder="Password"><br>**

**<input type="password" name="password2" placeholder="Confirm Password"><br>**

**<input type="Submit">**

**</form>**

**</body>**

**</html>**

* Now we fill that register form and submit it nothing is going to happen that data won’t be saved in our database table and user will not be created, so overcome that problem we have to check for the method POST in views.py and if it is POST we have to save it to database but before that we have to check if the password and confirm password is matching or not and we have to make sure that username and email doesn’t already exist in database. (Code will look like this)

***from* django.shortcuts *import* render, redirect**

***from* django.contrib *import* messages**

***from* django.contrib.auth.models *import* User, auth**

**def register(*request*):**

***if* *request*.method == 'POST':**

**first\_name = *request*.POST['first\_name']**

**last\_name = *request*.POST['last\_name']**

**username = *request*.POST['username']**

**email = *request*.POST['email']**

**password1 = *request*.POST['password1']**

**password2 = *request*.POST['password2']**

***if* password1 == password2:**

***if* User.objects.filter(*username*=username).exists():**

**messages.info(*request*, "Username Taken")**

***return* redirect('register')**

***elif* User.objects.filter(*email*=email).exists():**

**messages.info(*request*, "Email Taken")**

***return* redirect('register')**

***else*:**

**user = User.objects.create\_user(*username*=username, *password*=password1, *email*=email, *first\_name*=first\_name, *last\_name*=last\_name)**

**user.save()**

**print("User Created")**

***return* redirect('/')**

***else*:**

**messages.info(*request*, "Password not matching...")**

***return* redirect('register')**

***return* render(*request*, 'register.html')**

* To display those messages in out register.html page we will use div tag inside the register.html body, we can place it wherever we want to print the messages and we can decorate the message as our choice for example if it should be pop-up or in site message just beside the username for username error etc. the div code will look like this.

**<div>**

**{% for message in messages %}**

**<h3>{{message}}</h3>**

**{% endfor %}**

**</div>**

* Now to setup **Login** first of all we will add Login menu button in our index.html and the code for that is

**<li><a href="accounts/login">Login</a></li>**

* After that we will create login.html in templates and add the form and message section there. Code will look like this

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta http-equiv="X-UA-Compatible" content="IE=edge">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>Login</title>**

**</head>**

**<body>**

**<form action="login" method="post">**

**{% csrf\_token %}**

**<input type="text" name="username" placeholder="username"><br>**

**<input type="password" name="password" placeholder="password"><br>**

**<input type="submit">**

**</form>**

**<div>**

**{% for message in messages %}**

**<h3>{{message}}</h3>**

**{% endfor %}**

**</div>**

**</body>**

**</html>**

* Now we have to map this login page in urls.py of app and the code should look like this

**urlpatterns = [**

**path('register', views.register, *name*='register'),**

**path('login', views.login, *name*='login')**

**]**

* Now we have to define this function in views.py of app (which is accounts), and it will look like this

**def login(*request*):**

***if* *request*.method == 'POST':**

**username = *request*.POST['username']**

**password = *request*.POST['password']**

**user = auth.authenticate(*username*=username, *password*=password)**

***if* user is not None:**

**auth.login(*request*, user)**

***return* redirect('/')**

***else*:**

**messages.info(*request*, "Invalid Credentials")**

***return* redirect('login')**

***else*:**

***return* render(*request*, 'login.html')**

* And after the registration we will return the login page and to do that we have to add return redirect login code in register functions else part.

***else*:**

**user = User.objects.create\_user(*username*=username, *password*=password1, *email*=email, *first\_name*=first\_name, *last\_name*=last\_name)**

**user.save()**

**print("User Created")**

***return* redirect('login')**

* Logout and Using conditionals to that it doesn’t show us the register button when logged in, it should show us the logout button with our name. To do that we have to use if condition in our index.html and it will look like this.

**<li class="active"><a href="index.html">Home</a></li>**

**<li><a href="contact.html">Contact</a></li>**

**{% if user.is\_authenticated %}**

**<li>Hello, {{user.first\_name}}</li>**

**<li><a href="accounts/logout">Logout</a></li>**

**{% else %}**

**<li><a href="accounts/register">Register</a></li>**

**<li><a href="accounts/login">Login</a></li>**

**{% endif %}**

* Now we have to map this logout in our apps urls.py and it should look like this.

**urlpatterns = [**

**path('register', views.register, *name*='register'),**

**path('login', views.login, *name*='login'),**

**path('logout', views.logout, *name*='logout')**

**]**

* After that we have to define logout function in our views.py of app and the code should be this.

**def logout(*request*):**

**auth.logout(*request*)**

***return* redirect('/')**

* If the user click the logout button user will be logged out.