# Building web application with DJango

**Aim:** To learn Django basics and create a simple 'hello world' application.

**Requirements:** Python 3.X, Django 2.X, MySQL

**1. Introduction:** Django is the Web Framework of Python which can we used to build Web Applications. Django can be (and has been) used to build almost any type of website, from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc).

**2. Creating a new project**

1) Open terminal and go to the directory where you want to create the project.
2) Run following command,

***$django-admin startproject myproject***

This will create a new project directory named 'myproject' (instead of writing myproject, you may give appropriate name for your project.). The newly created files and folders inside the 'myproject' directory are as follows,

myproject
--> manage.py
--> myproject
--> __init__.py
--> settings.py
--> urls.py
--> wsgi.py

Description of these files and directories is as follows,

- The outer myproject/ root directory is just a container for your project. (Its name doesn't matter to Django; you can rename it to anything you like.)
- **manage.py**: A command-line utility to manage the project. (We will see its use with example.)
- The inner myproject/ directory is the actual Python package for your project. Its name is the Python package name you'll need to use to import anything inside it (e.g. **demo.urls**).
- **__init__.py**: An empty file that tells Python that this directory should be considered as a Python package.
- **settings.py**: This file stores Settings/configuration for the project.
- **urls.py**: The URL declarations for the project. (a "table of contents" of your Django-powered site.)
- **wsgi.py**: An entry-point for WSGI-compatible web servers to serve your project.

### 3. Creating an WebApplication inside the project

A WebApp is a Web application that provides some service through website – e.g., a Weblog system, a database of public records or a simple poll app. A project is a collection of configuration and apps for a particular website. A project can contain multiple apps. An app can be in multiple projects.

**Step 1 :** To create your app, make sure you're in the same directory as **manage.py** and type this command:

```
$python manage.py startapp hello_world
```

This will create an directory for 'hello_world' app, which contains following files,

--> hello_world

- --> admin.py
- --> __init__.py
- --> models.py
- --> tests.py
- --> views.py
- --> migrations

model.py is the file which specifies database table descriptions. Most of the functions (business logic) are specified in views.py. (We will discuss use of other files later)

**Step 2 :** now create **urls.py** file in hello_world app and write following code in it.

```
from django.conf.urls import url
from . import views
urlpatterns = [
        url('', views.index, name='index'),
]
```

**Step 3 :** Now modify myproject/urls.py (Project's urls.py file) as follows,

import include, e.g. from django.conf.urls url **,include**

==>insert following url in urlpatterns,

```
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^hello_world/', include('hello_world.urls')),
]
```

**Step 4 :** Write following code in views.py file (/hello_world/views.py) of the project.

```
from django.http import HttpResponse
def index(request):
    return HttpResponse("hello World!! Django is unchained!" )
```

**Step 5 :** Update the changes to server by calling make migration. From the project directory (where manage.py is located) run the following commands,

```
$python manage.py makemigrations
$python manage.py migrate
```

**Step 6 :** Start the server: Go to project directory (where manage.py is located) and run the following command.

```
$python manage.py runserver
```

**Step 7:** Run the application

Open the web browser and enter the following url, 127.0.0.1:8000/hello_world/