Django configuration on Raspberry Pi 3B+

Are you new to Django? or to IoT Web development in specific to Raspberry platform? Well, read this material to quickly get up and running.

1. Introduction to Django

Django is a wonderful web framework in Python and allows creating up-to-date web applications with comparably low efforts. It was designed to make common Web development tasks fast and easy.It is a Python-based free and open-source web framework, which follows the model-template-view (MTV) architectural pattern.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. (Don't repeat yourself is a principle of software development aimed at reducing repetition of software patterns, replacing it with abstractions or using data normalization to avoid redundancy.)

Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Some well-known sites that use Django include the Public Broadcasting Service, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket, and Nextdoor

2. Development tools with Django support

For developing a Django project, no special tools are necessary, since the source code can be edited with any conventional text editor. Nevertheless, editors specialized on computer programming can help increase the productivity of development, e.g., with features such as syntax highlighting. Since Django is written in Python, text editors which are aware of Python syntax are beneficial in this regard.

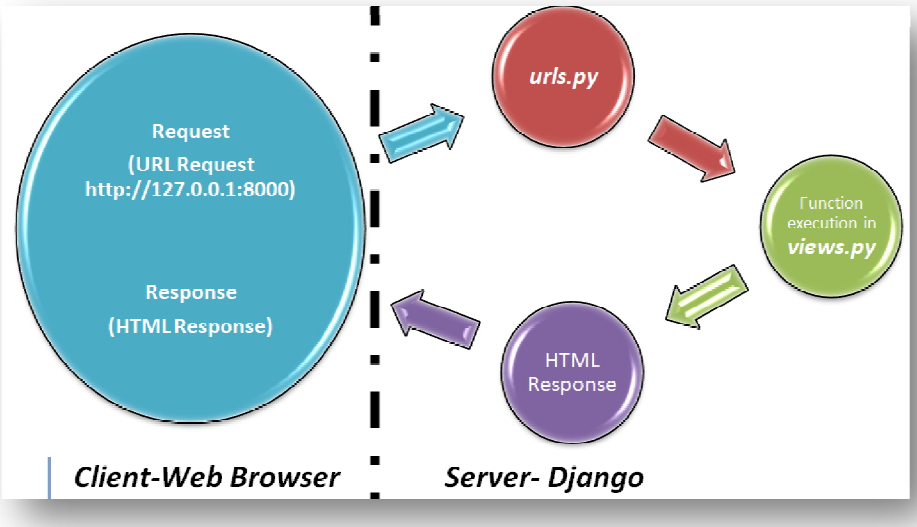
3. Django Project Layout

Consider a Django project created with command

django-admin startproject mysite

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

- The outer mysite/ 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 that lets you interact with this Django project in various ways.
- The inner mysite/ 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. mysite.urls).
- **__init__.py**: An empty file that tells Python that this directory should be considered a Python package.
- **settings.py**: Settings/configuration for this Django project
- **wsgi.py:** An entry-point for WSGI-compatible web servers to serve your project.
- **urls.py**: The URL declarations for this Django project; a "table of contents" of your Django-powered site. To design URLs for an app. This module is pure Python code and is a mapping between URL path expressions to Python functions (views).
- **Views.py**: A view function, or view for short, is simply a Python function that takes a Web request and returns a Web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image . . . or anything, really. views.py can be placed in project or application directory.



Part 1

Setting up Django Server with Creating Virtual Environment for separating all installations from RPi Core

Sr No	Command	Explanation
1	***sudo pip3 install virtualenv***	install virtual environment in python 3
2	***mkdir t1***	creating t1 named folder in RPi Home directory
3	***cd t1***	Enter into t1 folder
4	***virtualenv t1env***	creating instance of virtual env
5	***source t1env/bin/activate***	Activate virtual environment

Installing and starting Django Server

Sr No	Command	Explanation
1	***pip install django***	installing Django server, make sure it install for python 3
2	***django-admin startproject my***	Creating user defined django project 'my'
3	***cd my***	enter into my directory
4	***./manage.py runserver***	running server, note down the response, default IP - 127.0.0.1:8000
5	go to web browser and type http://127.0.0.1:8000	Now run this IP in web browser hit enter

Part 2

Defining own home page

Sr No	Steps	Explanation
1	Open django project, 'my', it contains two entities, 1. A folder with name 'my' 2. A python file named 'manage.py' open 'my' open urls.py and add following instructions ***from . import views*** ***urlpattens=[*** ***path('',views.index),*** ***]*** save it	referring views.py present in same folder dispatching to user-defined python function 'index', written in views.py
2	Create views.py file in 'my' folder and write following ***from django.http import HttpResponse*** ***def index(request):*** ***return(HttpResponse(' My First Web Page '))*** save it	upon execution of this function, it will retun HTML response, here a string. HttpResponse parameter must be in pair of ''
3	go to web browser and type http://127.0.0.1:8000	

Part 3

Blinking LED

Sr No	Steps
1	in 'my' folder, open views.py add following code **from django.http import HttpResponse** **import RPi.GPIO as GPIO** **def ON(request):** **GPIO.setmode(GPIO.BOARD)** **GPIO.setup(11, GPIO.OUT)** **GPIO.output(11,True)** **return(*HttpResponse(' LED ON ')*)** **def OFF(request):** **GPIO.setmode(GPIO.BOARD)** **GPIO.setup(11, GPIO.OUT)** **GPIO.output(11,False)** **return(*HttpResponse(' LED OFF ')*)** **def index(request):** **return(*HttpResponse(' Home Page ')*)**
2	in 'my' folder, open urls.py update the code as follows ***from . import views*** ***urlpattens=[*** ***path('',views.index),*** ***path('ON/',views.ON),*** ***path('OFF/',views.OFF),*** ***path('admin/',admin.site.urls),*** ***]*** save it
3	go to web browser and type a) URL for home page http://127.0.0.1:8000 b) URL for turning LED ON http://127.0.0.1:8000/ON/ c) URL for turning LED OFF http://127.0.0.1:8000/OFF/
4	On home page, buttons can be created in HTML, upon clicking which will open corresponding URLs for LED ON and OFF details are given in next step

5	Response for Home page can be be send through separate HTML page instead of ***HttpResponse(' Home Page ')***

follow the steps

create a folder with user defined name say **HomeFolder**, at project directory where you can see manage.py file

open it and create an empty file with extension .html, say **mypage.html**, open it with text editor and add following html code in it

```
<center>
<h1>Hello PICT</h1>
<form action="/ON/"><input type="submit" value="LED ON" /></form>
<form action="/OFF/"><input type="submit" value="LED OFF" /></form>
</center>
```

save it

open settings.py, in template section, find DIR=[ ], and edit it as follows

```
DIR=['HomeFolder/']
```

save it

open views .py and make changes in code as follows

```
def index(request):
    return render(request,'mypage.html',)

LED_PIN = 32
def ON(request):
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(LED_PIN,GPIO.OUT)
    GPIO.output(LED_PIN, 1)
    return HttpResponse('LED is ON...<form action="/">
                                    <input type="submit" value="Back" /></form>')

def OFF(request):
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(LED_PIN,GPIO.OUT)
    GPIO.output(LED_PIN, 0)
    return HttpResponse('LED is OFF...<form action="/">
                                    <input type="submit" value="Back" /></form>')
```

go to web browser and type
URL for home page

http://127.0.0.1:8000

References

1. https://docs.djangoproject.com/en/1.8/intro/tutorial01/

2. https://iotbyhvm.ooo/iot-communication-models/

3. https://en.wikipedia.org/wiki/Django_(web_framework)