Flask

Building web applications in flask using Python, HTML and CSS

The web applications are mainly websites and apps.

Binding to a host name, listening to the external request, clients request is returned by the server.

All these above protocols are taken care by the framework.

All python based web frameworks WSGI compliant- Web server gateway interface  
eg: Django, Flask, Pyramid, TurboGears, Web2py.

Whereas Flask is a micro-framework.

Werkzeug (work-zug) toolkit, WSGI compliant

++++++++++++++++++++++++++++++++

Starting a project it is advised to have a virtual environment.  
pip(Python package Installer) install virtualenv.

… … … … … … … … … … To avoid conflict of libraries.

<0>  
<1>virtualenv env -> where env is the name we give to the virtual envo  
<2>env/Scripts/Activate  
<3> env/Scripts/deactivate

++++++++++++++++++++++++++++++++

Import flask class from flask modle,

Create Flask class Object. WSGI

Flask constructor – name of the current module – (\_\_name\_\_)

\_\_name\_\_

from flask import \*

#app=Flask(\_\_name\_\_)

flak\_machine=Flask(\_\_name\_\_)

route() – define url – map func – decorator

app.route(rule, options)

Rule: url

Options: parameters to pass

@flak\_machine.route('/')

def index():

    return render\_template('index.html')

App.run(host,port,debug,options)

Host: localhost (defaults: 127.0.0.1) , set it 0.0.0.0 to make it external (debug must be False)

Port: default to 5000

Debug: debug in realtime, defaults to false

Options: Informations to pass to werkzeug function

#app.run()

if \_\_name\_\_ == '\_\_main\_\_':

    flak\_machine.run(*debug*=True)

++++++++++++++++++++++++++++++++

from flask import Flask 🡪 importing Flask class from flask module

app = Flask(\_\_name\_\_) 🡪 Name for flask constructor..creates flask object

@app.route(‘/’) 🡪 route decorator

def hello\_world(): 🡪 View function

return ‘Hello World’

if \_\_name\_\_ == ‘\_\_main\_\_’: 🡪 Initialize as Main

app.run()

++++++++++++++++++++++++++++++++

app.debug=True

app.run(*debug*=True)

debug helps in updating the flask app every time changes have been made with a debugger to track. Instead of stopping and starting every time changes have been made.

++++++++++++++++++++++++++++++++

MODEL VIEW CONTROL – MVC frameworks

Routing – route decorator – defines an url and binds a function (view function)

++++++++++++++++++++++++++++++++

We can pass variable data into the view function,

@app.route('/myurl/<variable\_1>')

🡪 Notice the <angular brackets> within which the variable data is passed.

def view\_func(*variable\_1*):

🡪 Notice the parameter in which the variable is passed as argument

return f'hello {variable\_1}'

🡪 Notice the string formatting method used to pass the variable data into the returned data, **the variable data is used**

++++++++++++++++++++++++++++++++

**URL Building**

url\_for()

from flask import Flask, redirect, url\_for

app= Flask(\_\_name\_\_)

@app.route('/page\_1')

def hello():

    return 'hello world'

@app.route('/page\_2/<variable\_1>')

def hello\_var\_1(*variable\_1*):

    return f'hello {*variable\_1*}'

#Upto this there are two urls, one defined and another with variable passing

++++++++++++++++++++++++++++++++

**#DYNAMIC URL FETCHING**

#Now we are creating a unique url that we can pass any variables inside it, that redirects it to any of the above two urls.

@app.route('/page\_3/<variable\_1>')

def hello\_var\_2(*variable\_1*):

    if *variable\_1* == 'admin':

        return redirect (url\_for ('hello\_var\_1'))

    else:

        return redirect (url\_for ('hello\_var\_2',*variable\_1* = *variable\_1*))

Depending upon the data passed inside the variable, the url is redirected using the redirect() function and inside it the url\_for() function is passed with argument with the view function name of other urls.

i.e, Depending upon the Condition,  
The view function name is passed within quotations ‘string’ to url\_for()

The url\_for(‘view\_function’) passed as argument for redirect()

The redirect(url\_for(‘view\_function’) is returned.

++++++++++++++++++++++++++++++++

HTTP = Hyper Text :structured text : nodes :transfer hypertext, exchange, http are the **methods** upon the resource for communication, resource are data exchanged between nodes controlled by http. http 1.0 – GET & POST

http 1.1 = PUT DELETE HEAD TRACE & CONNECT

GET – unencrypted default method (server remembers data)

POST – send html to server (send html form data, Not Cached by serer)

HEAD – much like GET, but no response body

PUT – replaces all current representations of target resource

DELETE - Deletess all current representations of target resource

TRACE – received request is echoed

CONNECT – converts connection to a transparent TCP/IP tunnel.

from flask import Flask, request  
app=Flask(\_\_name\_\_)  
@app.route(‘/’, methods= [‘POST’,’GET’])  
def function():  
 if request.method == ‘POST’:  
 #if method is post this statement executes  
 else:  
 #else this executes