

We have used flask and pywebio for deployment. PyWebIO provides a series of imperative functions to obtain user input and output on the browser, turning the browser into a "rich text terminal", and can be used to build simple web applications or browser-based GUI applications without the need to have knowledge of HTML and JS. PyWebIO can also be easily integrated into existing Web services.

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

**Installations**

Stable version:

pip install pywebio

When using the Flask as PyWebIO backend server, you need to install Flask by yourself and make sure the version is not less than 0.10. You can install it with the following command:

pip install -U flask >=0.10

**Deployment Implementation**

* we will import the webio\_view library to Get the view function for running PyWebIO applications in Flask. The view communicates with the browser by HTTP protocol.

from pywebio.platform.flask import webio\_view

* The path of the static file of PyWebIO is stored in pywebio.STATIC\_PATH

from pywebio import STATIC\_PATH

* Send a file from a given directory with send\_file(). This is a secure way to quickly expose static files from an upload folder or something similar. And for rendering html files we are using render\_template

from flask import Flask, send\_from\_directory,render\_template

* For storing and extracting data from mongo dB we are using the pymongo library.

from pymongo import MongoClient

* This module provides functions to get all kinds of input of user from the browser

from pywebio.input import \*

* This module provides functions to output all kinds of content to the user’s browser, and supply flexible output control.

from pywebio.output import \*

* configure the environment of the current session.

from pywebio.session import set\_env

* Flask constructor takes the name of the current module (\_\_name\_\_) as argument.

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

* The route() function of the Flask class is a decorator, which tells the application which URL should call the associated function.

@app.route("/")

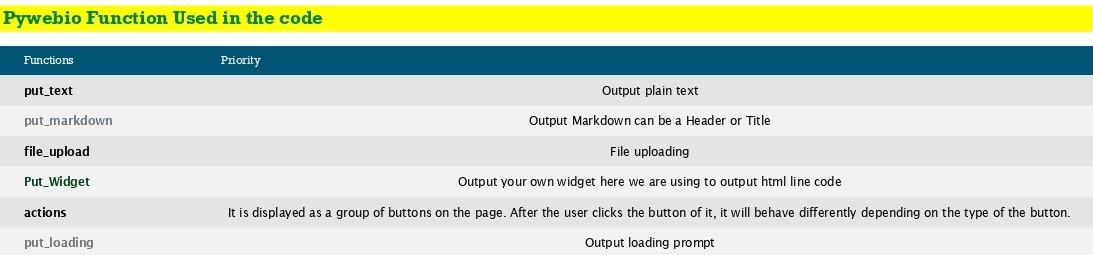
* ‘/’ URL is bound with hello\_world() function which returns the rendered Html template for making a homepage for the app.

For Rendering Html template we need to have a folder called templates in the same directory from where the python deployment code is saved and inside that an Html file to render .

def hello\_world():

    return render\_template('inner.html')

* Then we will define a function called model() where we will give different inputs and fileupload function and lastly, we will run our transformer model to generate an output



* we call our function for this we need to add a rule so in app we have a function called as add\_url\_rule so here I am going to use a url like ‘/tool’ I have to give second parameter which is ‘webio\_view’ and this web\_view which I have imported previously in this I just need to pass our function called model and this will do entire execution

app.add\_url\_rule('/tool','webio\_view',webio\_view(model),

methods=['GET','POST','OPTIONS'])

* Then we will use app.run to run our app on a local server you have to give the port number.

app.run(host='127.0.0.1',port=5000)

* Lastly our app is Running on <http://127.0.0.1:5000/>

**Conclusion**

Pywebio is a beautiful library which allows us to showcase our code on web as if we are just writing on terminal and its getting displayed on to the web For more understanding you can go to the below link:

[PyWebIO — PyWebIO 1.4.0 documentation](https://pywebio.readthedocs.io/en/latest/)