We now have entered the architecture of web services. Since you are already familiar in building a website, let us stretch that knowledge further by utilizing python implementation for your projects and publish them through world wide.

Now before you join the others let us first use a back-end framework called *Flask* , flask is lightweight module with supporting APIs that manages web application of its functionalities. If you want to learn more about flask visit [https://flask.palletsprojects.com/](https://flask.palletsprojects.com/" \t "https://www.google.com/_blank). This is a lunch program to familiarize you with pull and fetch between client to server.

This is a lunch program to familiarize you with pull and request between client to server.The snippets will coach you how the interaction works respectively to the language we are using..

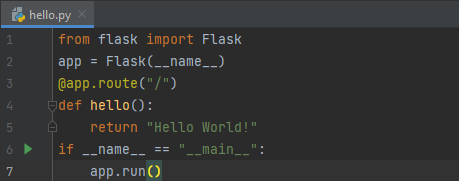
STEP 1: Installation of Flask

First, we would want to deploy flask in our extension. In your python IDE terminal type: pip install Flask.

If you don’t have the ‘pip’ installer on your environment download it from http://pypi.python.org/packages/source/F/Flask/Flask-0.10.1.tar.gz - unzip and start the setup.

STEP 2 : Provide a web server

In our web server, create a dictionary to hold a JSON objects for a couple of employee records and then add RESTful APIs for each supported operation. Save this file as hello.py and execute / run the code.



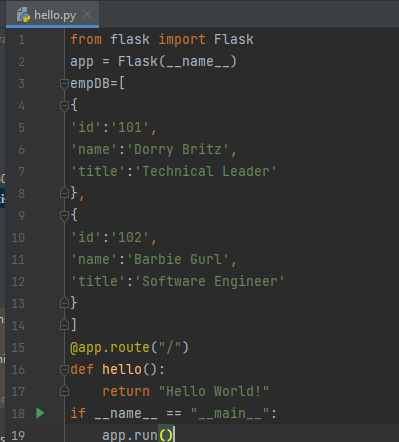
The arrangements we did here:

1. We imported the package in our hello program,
2. The line ‘app = Flask(\_ \_ name \_ \_)’ is our object,
3. app.run() starts the web server and ready to handle the request. At this state we only handled one request.

STEP 3 : Populate the web services.

To develop the restful services for the planned objective, let's create an in-memory database in

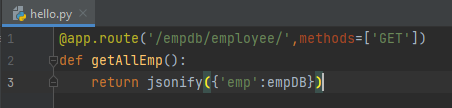
python using the dictionary. After the Flask app creation statement app = Flask(\_\_name\_\_),



STEP 3: using GET function

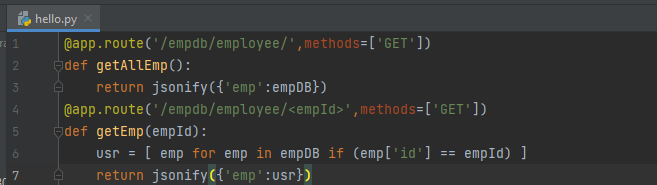
we have created two employees in the dictionary. Now let's write a code to retrieve them using web services. As per our plan, we need two implementations one is to retrieve all the employees and another one to retrieve the specific employee with the given id.

STEP 3.1: all



In the above code, we have created a URI named '/empdb/employee' and also we defined the method as "GET". To service the GET call for the URI, Flask will call the function getAllEmp(). It will in turn simply call the "jsonify" method with employeeDB as the argument. The "jsonify" is a flask method, will set the data with the given JSON object which is passed as a Python dictionary and set the headers appropriately, in this case "Content-type: application/json".

STEP 3.1.2 : specific



The above code will find the employee object with the given id and send the JSON object in the data.

STEP 4: Try for yourself:

This is the complete code of the session check each of the inputs on your browser.

from flask import Flask

from flask import jsonify

from flask import request

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

empDB=[

{

'id':'101',

'name':'Dorry Britz',

'title':'Technical Leader'

},

{

'id':'102',

'name':'Barbie Gurl',

'title':'Software Engineer'

}

]

@app.route("/")

def hello():

return "Hello World!"

@app.route('/empdb/employee/',methods=['GET'])

def getAllEmp():

return jsonify({'emp':empDB})

@app.route('/empdb/employee/<empId>',methods=['GET'])

def getEmp(empId):

usr = [ emp for emp in empDB if (emp['id'] == empId) ]

return jsonify({'emp':usr})

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

app.run()

Type these in your media one by one :

1. <http://127.0.0.1:5000/>
2. <http://localhost:5000/empdb/employee/>
3. <http://localhost:5000/empdb/employee/101>
4. <http://localhost:5000/empdb/employee/103>
5. On your cmd <http://localhost:5000/empdb/employee/>

**Conclusion >**

The intention to start with flask in contrary to other frameworks , as a developer option, it has basic concepts suitable for rookies working with maintainable projects.