**Webex Teams Hackathon 2018**



**Lab1 - Creating REST API Back End in Python**

# Objectives

In this lab, you will complete the following objectives:

* Create a simple REST API back end using Flask
* Create a simple GET API endpoint
* Create endpoint serving JSON response
* Create POST API endpoint
* Use browser and Postman to test the endpoints

# Background / Scenario

Python can be used to act as a REST API endpoint serving GET, POST and other API requests. In this lab, you will install the Flask module. Flask is a microframework for Python which can be used for serving API endpoints.

When completed, the **lab1-back-end-python.py** program will run as a REST API server and serve:

* simple GET API endpoint
* endpoint which sends back JSON response
* simple POST API endpoint which can modify the data stored in the back end system.

# Required Resources

* Postman application
* Python 3 with IDLE
* Python code files

**Step 1: Create your Flask project**

In this step, you will create your project, install Flask and import Flask to your project.

1. Create your project directory called lab1-back-end-python and create a lab1-back-end-python.py file in it.
2. Open a command line in administrative mode and install Flask using the following command:   
   pip install Flask
3. Edit your lab1-back-end-python.py file and import Flask. Add the following text to the beginning of the file:   
   from flask import Flask

**Step 2: Create and test your first API endpoint**

1. Create a Flask HTTP back end: app = Flask(\_\_name\_\_)
2. Add your first GET endpoint:

@app.route("/api/helloworld")

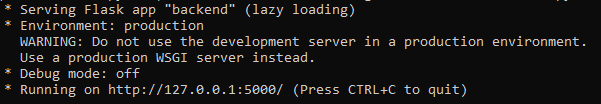
def hello():

return "Hello World!"

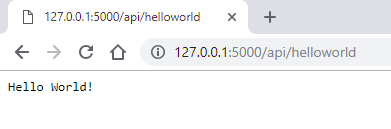
This endpoint can be accessed by the following URL: http://<your\_server\_name>/api/helloworld

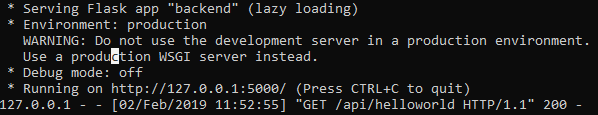
**Note:** It is not mandatory but recommended to use the ‘/api’ prefix for your REST API endpoints. In your real project you may use whatever prefix you want.

1. Add app.run() command to the end of the file to run your Flask HTTP back end instance.
2. Launch your back end app. In the command line type python lab1-back-end-python.py



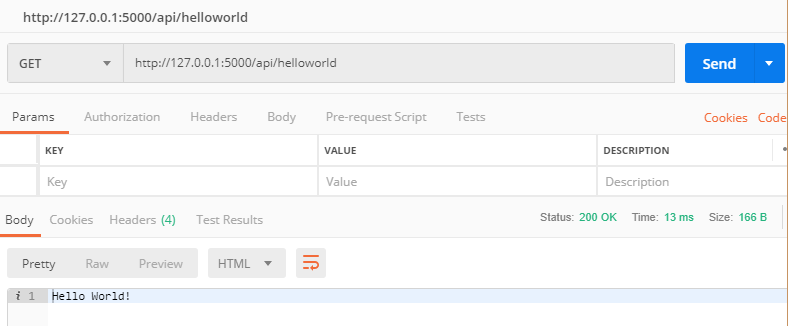
1. Open your Chrome browser and type the following url: <http://127.0.0.1:5000/api/helloworld>





Congratulation, you have created your first REST API endpoint!

1. Test your endpoint in Postman



1. Stop your back end app. Press Crtl-C in the command windows.

**Step 3: Create endpoint serving JSON response**

1. Import jsonify module from Flask. Change your Flask import line for the following:

from flask import Flask, jsonify

1. Add these variables before your endpoint created at Step 2:

name = "Charles Webex"

age = "15"

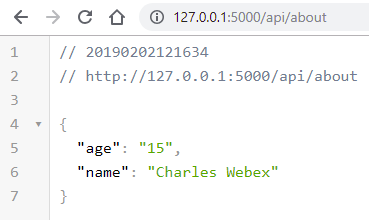
1. Create a new endpoint:

@app.route("/api/about")

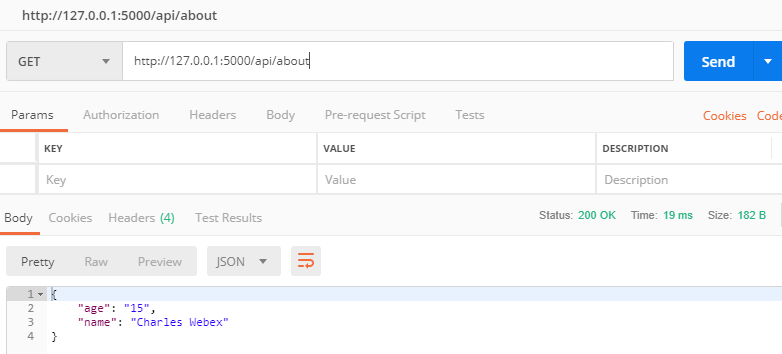
def about():

return jsonify(name = name, age = age)

1. Launch your back end app. In the command line type python lab1-back-end-python.py
2. Open your Chrome browser and type the following url: <http://127.0.0.1:5000/api/about>



1. Test your endpoint in Postman:



1. Stop your back end app. Press Crtl-C in the command windows.

**Step 2: Create POST API endpoint**

1. Import request module from Flask. Change your Flask import line for the following:

from flask import Flask, jsonify, request

1. Modify your endpoint created at Step 3:

@app.route("/api/about", methods = ['POST', 'GET'])

def about():

global name, age

if request.method == 'GET':

return jsonify(name = name, age = age)

elif request.method == 'POST':

r = request.json

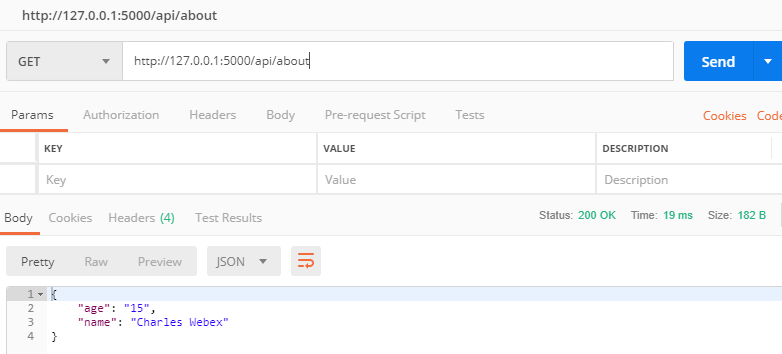
name = r["name"]

age = r["age"]

return jsonify(name = name, age = age)

**Note:** we use global command because we would like to use the global name and age variable. Without global command a new local (in the scope of about() function) variable would be created when we assign a new value to name or age variable.

1. Launch your back end app. In the command line type python lab1-back-end-python.py
2. Test your GET endpoint in Postman:



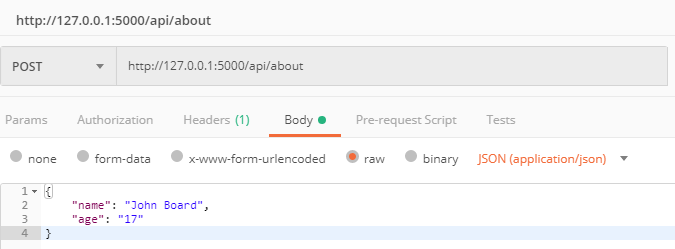
1. Test your POST endpoint in Postman. Use **raw** format and **JSON (application/json)** type in the Body settings then give the following body content:

{

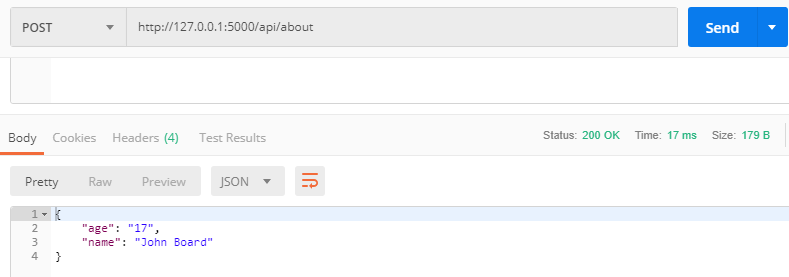
"name": "John Board",

"age": "17"

}



Click Send button to send your request to your back end.



**Note:** in the response from your back end you can find the name and age values in JSON format.

1. Check the new state using your GET endpoint.

