API Documents

Flask API Tool

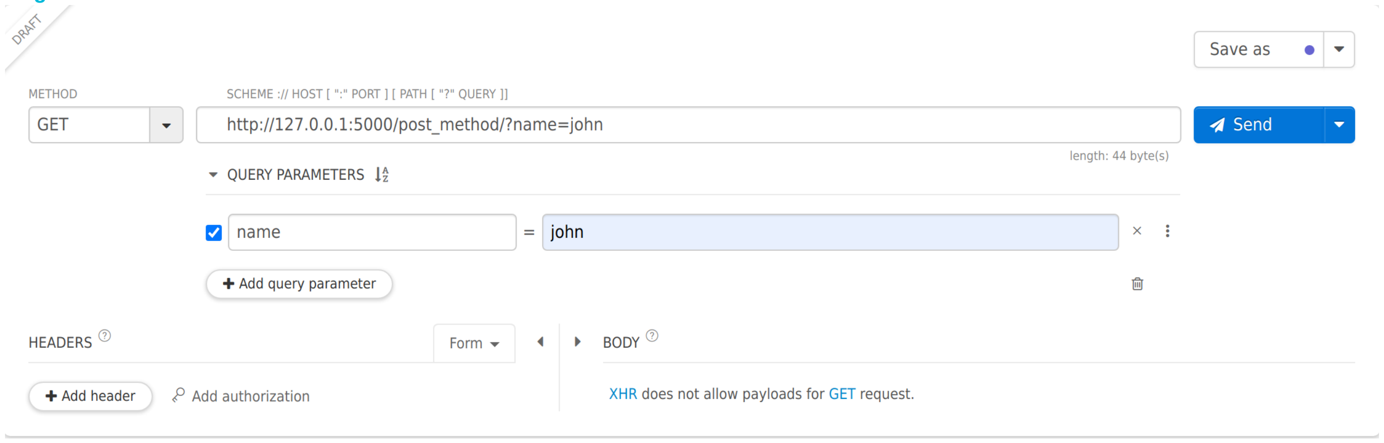
Flask query parameter Request or api router parameter

ex:

my route http://127.0.0.1:5000/post_method/

how to give parameter in route http://127.0.0.1:5000/post_method/?name=john

single value



http://127.0.0.1:5000/post_method/?name=john

length: 44 byte(s)

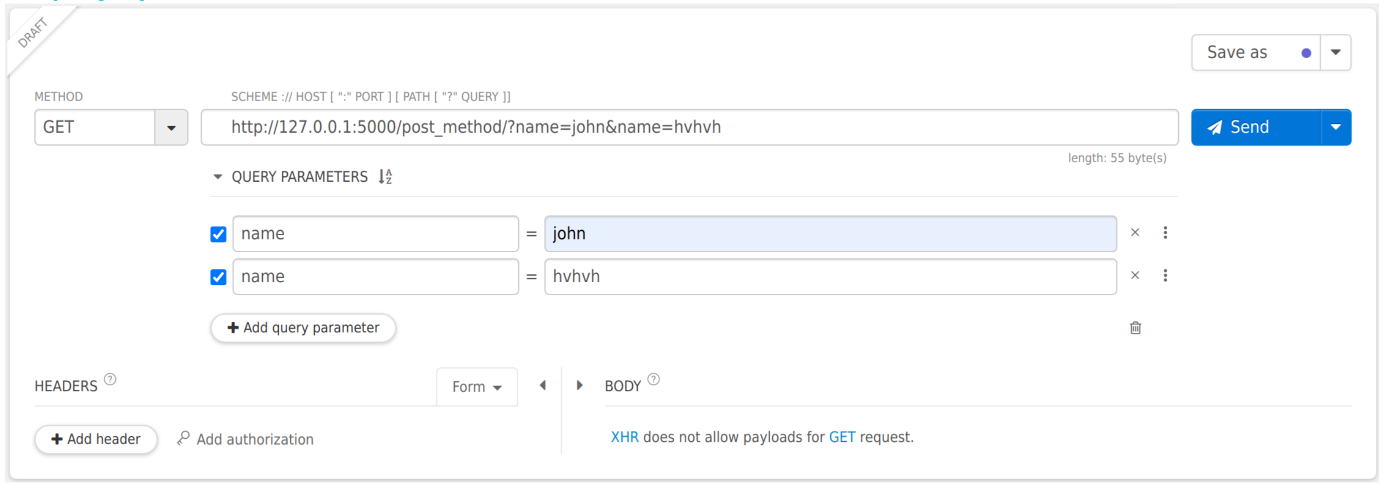
QUERY PARAMETERS

name = john

XHR does not allow payloads for GET request.

@app.route('/post_method/', methods=['GET'])
def mew_va():
name = request.args.get('name')
print(name)

Multiple Query value



http://127.0.0.1:5000/post_method/?name=john&name=hvhvh

length: 55 byte(s)

QUERY PARAMETERS

name = john

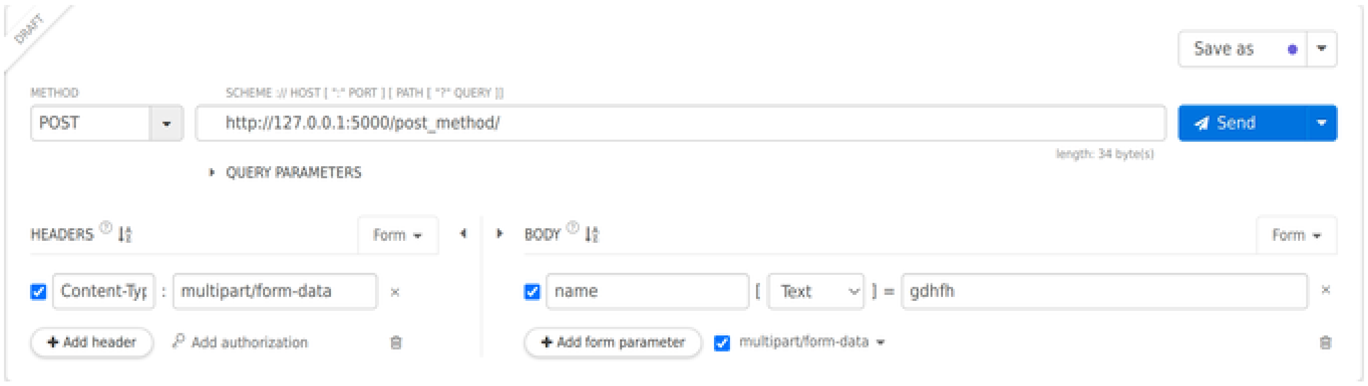
name = hvhvh

XHR does not allow payloads for GET request.

@app.route('/post_method/', methods=['GET'])
def mew_va():
name = request.args.getlist('name')
print(name[0], name[1])

Flask Form Request

Single Form



METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://127.0.0.1:5000/post_method/

length: 34 byte(s)

HEADERS: Form

✓ Content-Typ : multipart/form-data

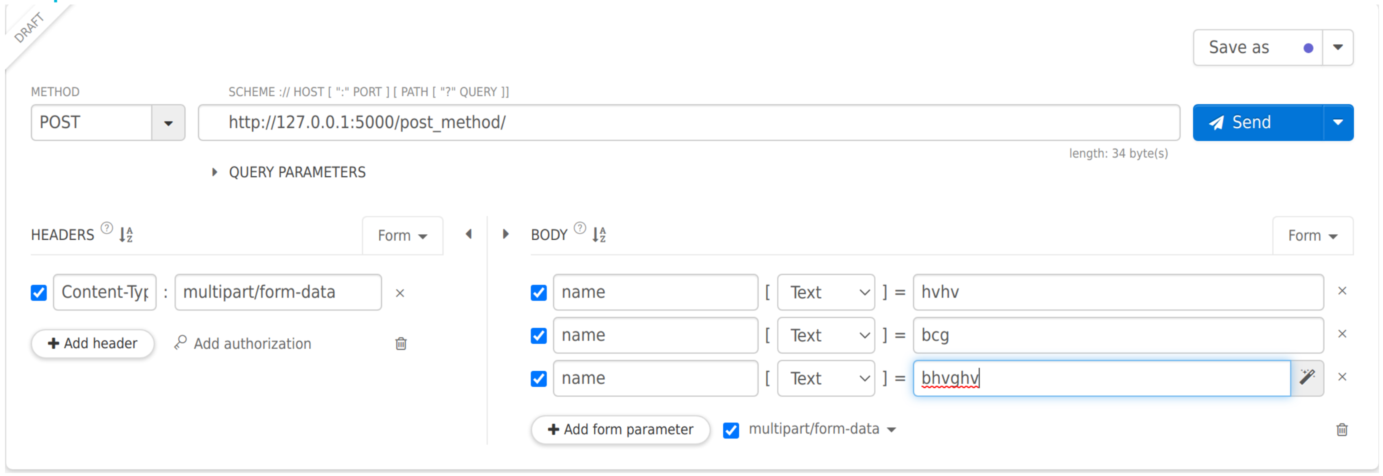
BODY: Form

✓ name [Text] = gdhfh

✓ multipart/form-data

request.form['name'] or request.form.get('name')

Multiple Form



METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://127.0.0.1:5000/post_method/

length: 34 byte(s)

HEADERS: Form

✓ Content-Typ : multipart/form-data

BODY: Form

✓ name [Text] = hvhv

✓ name [Text] = bcg

✓ name [Text] = bhvghv

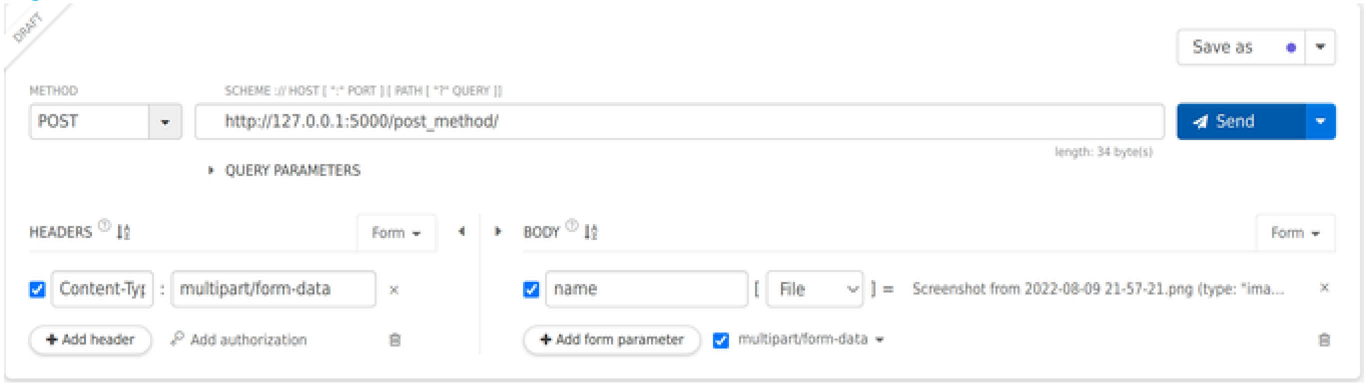
✓ multipart/form-data

data = request.form.getlist("name")

print(data[0])

Flask File Request

Single File



METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://127.0.0.1:5000/post_method/

length: 34 byte(s)

HEADERS: Form

✓ Content-Typ : multipart/form-data

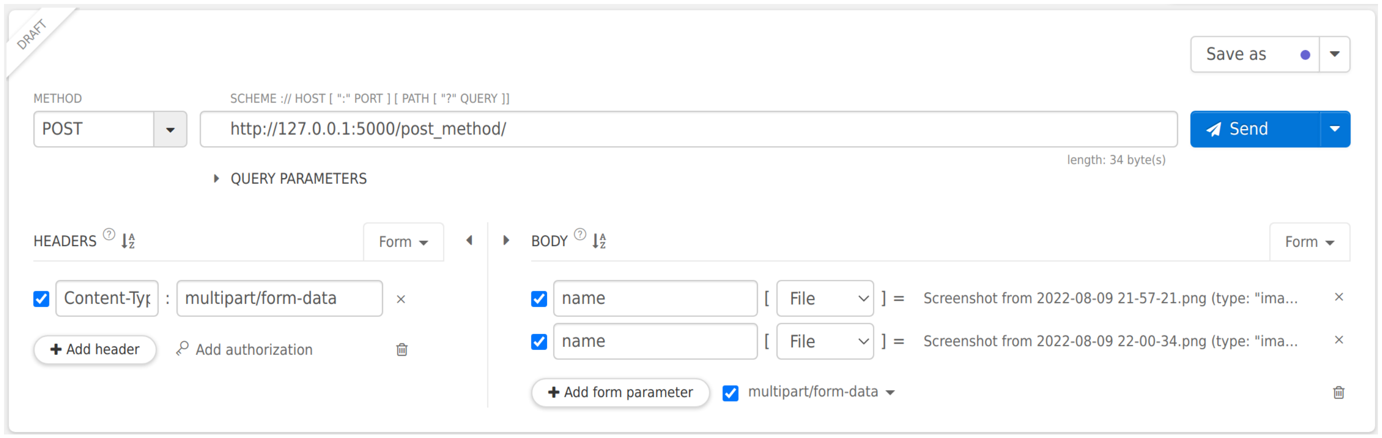
BODY: Form

✓ name [File] = Screenshot from 2022-08-09 21-57-21.png (type: "ima...

✓ multipart/form-data

imagefile = flask.request.files.get('name') or imagefile = request.files['name']

Multiple File


DRAFT

METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]

http://127.0.0.1:5000/post_method/

length: 34 byte(s)

HEADERS

Content-Typ : multipart/form-data

BODY

name [File] = Screenshot from 2022-08-09 21-57-21.png (type: "ima...

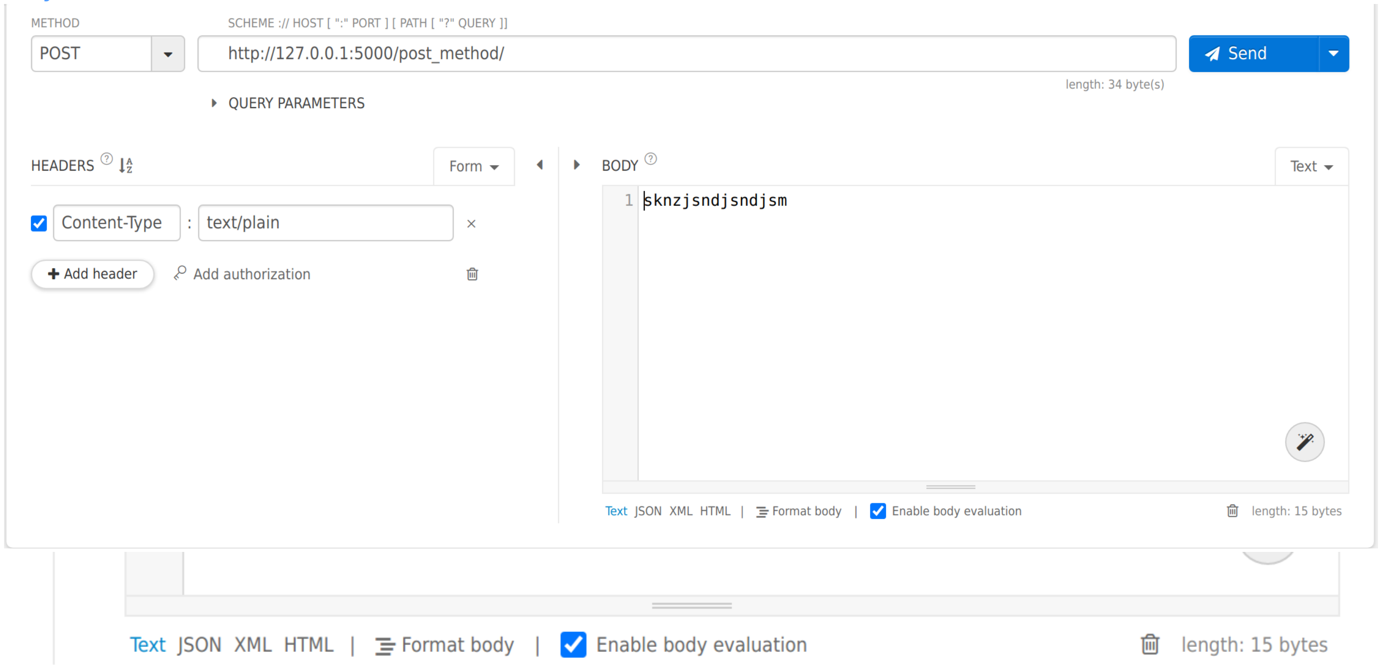
name [File] = Screenshot from 2022-08-09 22-00-34.png (type: "ima...

multipart/form-data

uploaded_files = request.files.getlist('file')
print(uploaded_files[0].filename)
print(uploaded_files[0].name)

Flask Text Request

Body Text


METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]

http://127.0.0.1:5000/post_method/

length: 34 byte(s)

HEADERS

Content-Type : text/plain

BODY: Text

```
sknzjsndjsndjsm
```

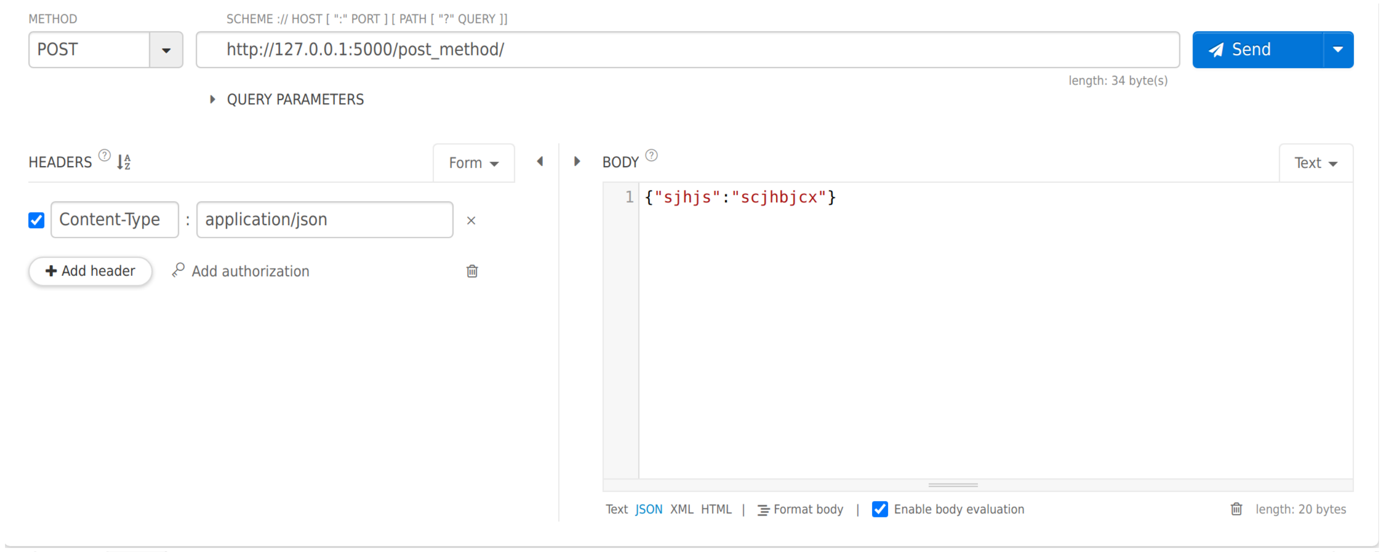
Text JSON XML HTML | Format body | ✓ Enable body evaluation | length: 15 bytes

data = request.get_data()

print(data)

Flask json Request

Body Json


METHOD POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]

http://127.0.0.1:5000/post_method/

length: 34 byte(s)

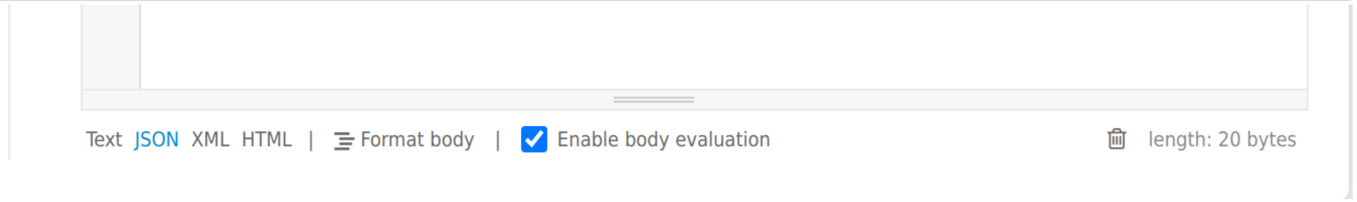
HEADERS

Content-Type : application/json

BODY

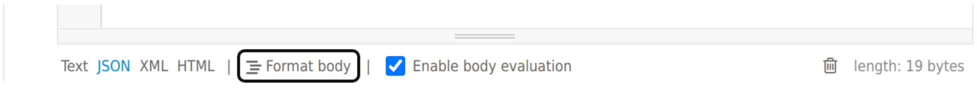
```
{"sjhjs":"scjhbjcx"}
```

length: 20 bytes


Text JSON XML HTML | Format body | Enable body evaluation length: 20 bytes

data = request.get_json()
print(data)

click format body button it shows better view


Text JSON XML HTML | Format body | Enable body evaluation length: 19 bytes

Flask With Mysql RestfulApi

what restful api

https://realpython.com/api-integration-in-python/

HTTP method	API endpoint	Description
GET	/customers	Get a list of customers.
GET	/customers/<customer_id>	Get a single customer.
POST	/customers	Create a new customer.
PUT	/customers/<customer_id>	Update a customer.
PATCH	/customers/<customer_id>	Partially update a customer.
DELETE	/customers/<customer_id>	Delete a customer.

HTTP method	API endpoint	Description
GET	/events/<event_id>/guests	Get a list of guests.
GET	/events/<event_id>/guests /<guest_id>	Get a single guest.
POST	/events/<event_id>/guests	Create a new guest.
PUT	/events/<event_id>/guests /<guest_id>	Update a guest.

`PATCH`	`/events/<event_id>/guests /<guest_id>`	Partially update a guest.
`DELETE`	`/events/<event_id>/guests /<guest_id>`	Delete a guest

Flask get post put delete code website

https://www.bogotobogo.com/python/python-REST-API-Http-Requests-for-Humans-with-Flask.php

Flask get post put delete with mysql

https://webdamn.com/create-restful-api-using-python-mysql/

Working Code

```
from flask import Flask, jsonify, request
import mysql.connector
import pandas as pd
import streamlit as st
from time import time
import json, yaml

app = Flask(__name__)

# Database Connection Information
mydb = mysql.connector.connect(host="localhost",
                               port="3306",
                               user="root",
                               password="root@123",
                               database="face")

# GET REQUEST 1 Method
@app.route('/get_method/', methods=['GET'])
def home():
    mydb.connect()
    if (request.method == 'GET'):
        if mydb.is_connected():
            mycursor = mydb.cursor(
            # query = "SELECT * FROM test;"
            # df = pd.read_sql(query, mydb)
            mycursor.execute("SELECT * FROM test;")
            df = pd.DataFrame(mycursor.fetchall())
            jsonfiles = json.loads(df.to_json(orient='records'))
            mycursor.close()
            mydb.close()
            return jsonify(jsonfiles)
        else:
            return jsonify({'data': "no"})

# GET REQUEST 2 Method
@app.route('/get_method/<int:num>', methods=['GET'])
def id_value(num):
    mydb.connect()
    if (request.method == 'GET'):
        if mydb.is_connected():
            mycursor = mydb.cursor()
            # query = f"select * from test where
id='{num}';"
            # df = pd.read_sql(query, mydb)
            mycursor.execute(f"select * from test where id='{num}';")
            df = pd.DataFrame(mycursor.fetchall())
            jsonfiles = json.loads(df.to_json(orient='records'))
            mycursor.close()
            return jsonify(jsonfiles)
        else:
            return jsonify({'data': "no"})
# POST REQUEST Method
@app.route('/post_method/', methods=['POST'])
def mew_va():
    mydb.connect()
    if (request.method == 'POST'):
        if mydb.is_connected():
            mycursor = mydb.cursor()
            id_val = request.form.get('id_val')
            name_val = request.form.get('name_val')
            in_val = request.form.get('intime_val')
            out_val = request.form.get('outtime_val')
            sql = "INSERT INTO test (id, name, intime, outtime) VALUES
(%s, %s, %s, %s)"
            val = (id_val, name_val, in_val, out_val)
            mycursor.execute(sql, val)
            mydb.commit()
            mydb.close()
            return jsonify({'data': "sucessfully inserted"})
        else:
            return jsonify({'data': "no"})
# PUT REQUEST Method
@app.route('/put_method/', methods=['PUT'])
def put_va():
    mydb.connect()
    if (request.method == 'GET'):
        if mydb.is_connected():
            mycursor = mydb.cursor()
            id_val = request.form.get('id_val')
            name_val = request.form.get('name_val')
            in_val = request.form.get('intime_val')
            out_val = request.form.get('outtime_val')
            # sql = "INSERT INTO test (id, name, intime, outtime)
VALUES (%s, %s, %s, %s)"
            sql = "UPDATE test SET name=%s, intime=%s, outtime=%s WHERE
id=%s"
            val = (name_val, in_val, out_val, id_val)
            mycursor.execute(sql, val)
            mydb.commit()
            mydb.close()
            respone = jsonify({'data': "Employee updated
successfully!"})
            respone.status_code = 200            return respone
        else:
            return jsonify({'data': "no"})
# DELETE REQUEST Method
@app.route('/del_method/<int:num>', methods=['DELETE'])
def del_value(num):
    mydb.connect()
    if (request.method == 'DELETE'):
        if mydb.is_connected():
            mycursor = mydb.cursor()
            mycursor.execute(f"DELETE FROM test WHERE id='{num}';")
            mydb.commit()
            mydb.close()
            respone = jsonify({'data': "Deleted successfully!"})
            respone.status_code = 200            return respone
        else:
            return jsonify({'data': "no"})

if _name_ == '__main__':
    app.run(debug=True)
```

Flask Restful API

```
from flask import Flask, jsonify, request
import mysql.connector
import pandas as pd
import streamlit as st
from time import time
import json

app = Flask(__name__)

mydb = mysql.connector.connect(host="localhost",port="3306",user="root",
password="root@123",database="face")
@app.route('/all_api/', methods=['GET', 'POST', 'PUT', 'DELETE'])
def new_va():
    if (request.method == 'GET'):
        if not request.args.get('name'):
            mydb.connect()
            if mydb.is_connected():
                mycursor = mydb.cursor()
                mycursor.execute("SELECT * FROM test;")
                df = pd.DataFrame(mycursor.fetchall())
                jsonfiles = json.loads(df.to_json(orient='records'))
                mycursor.close()
                mydb.close()
                return jsonify(jsonfiles)
            else:
                return jsonify({'data': "no"})
        else:
            num = request.args.get('name')
            mydb.connect()
            if mydb.is_connected():
                mycursor = mydb.cursor()
                mycursor.execute(f"select * from test where
id='{num}';")
                df = pd.DataFrame(mycursor.fetchall())
                jsonfiles = json.loads(df.to_json(orient='records'))
                mycursor.close()
                return jsonify(jsonfiles)
            else:
                return jsonify({'data': "no"})
    if (request.method == 'POST'):
        mydb.connect()
        if mydb.is_connected():
            mycursor = mydb.cursor()
            id_val = request.form.get('id_val')
            name_val = request.form.get('name_val')
            in_val = request.form.get('intime_val')
            out_val = request.form.get('outtime_val')
            sql = "INSERT INTO test (id, name, intime, outtime) VALUES
(%s, %s, %s, %s)"
            val = (id_val, name_val, in_val, out_val)
            mycursor.execute(sql, val)
            mydb.commit()
            mydb.close()
            return jsonify({'data': "sucessfully inserted"})
        else:
            return jsonify({'data': "no"})
    if (request.method == 'PUT'):
        mydb.connect()
        if mydb.is_connected():
            mycursor = mydb.cursor()
            id_val = request.form.get('id_val')
            name_val = request.form.get('name_val')
            in_val = request.form.get('intime_val')
            out_val = request.form.get('outtime_val')
            sql = "UPDATE test SET name=%s, intime=%s, outtime=%s WHERE
id=%s"
            val = (name_val, in_val, out_val, id_val)
            mycursor.execute(sql, val)
            mydb.commit()
            mydb.close()
            respone = jsonify({'data': "updated successfully!"})
            respone.status_code = 200             return respone
        else:
            return jsonify({'data': "no"})
    if (request.method == 'DELETE'):
        num = request.args.get('name')
        mydb.connect()
        if mydb.is_connected():
            mycursor = mydb.cursor()
            mycursor.execute(f"DELETE FROM test WHERE id='{num}';")
            mydb.commit()
            mydb.close()
            respone = jsonify({'data': "Deleted successfully!"})
            respone.status_code = 200             return respone
        else:
            return jsonify({'data': "no"})

if _name_ == '__main__':
    app.run(debug=True)
```

Method 2

Flask Restful API Using from flask_restful import Resource, Api

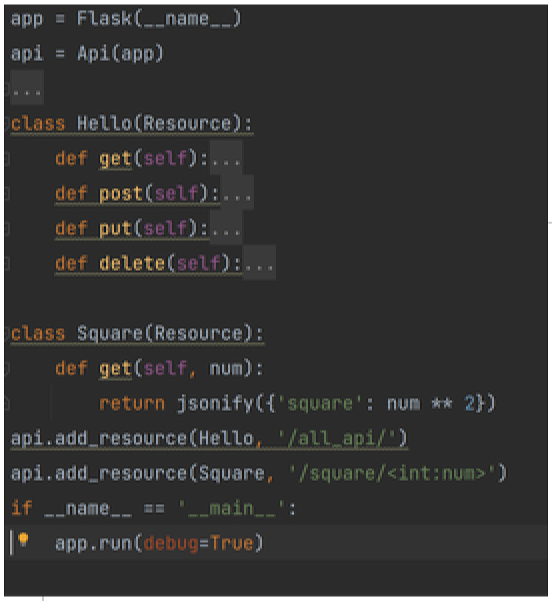
from flask import Flask, jsonify, request
from flask_restful import Resource, Api

```
from flask import Flask, jsonify, request
from flask_restful import Resource, Api
import mysql.connector
import pandas as pd
import streamlit as st
from time import time
import json

app = Flask(__name__)
api = Api(app)

mydb = mysql.connector.connect(
            host="localhost", port="3306", user="root",password="
root@123",database="face")

class Hello(Resource):
    def get(self):
        if not request.args.get('name'):
            mydb.connect()
            if mydb.is_connected():
                mycursor = mydb.cursor()
                mycursor.execute("SELECT * FROM test;")
                df = pd.DataFrame(mycursor.fetchall())
                jsonfiles = json.loads(df.to_json(orient='records'))
                mycursor.close()
                mydb.close()
                return jsonify(jsonfiles)
            else:
                return jsonify({'data': "no"})
        else:
            num = request.args.get('name')
            mydb.connect()
            if mydb.is_connected():
                mycursor = mydb.cursor()
                mycursor.execute(f"select * from test where
id='{num}';")
                df = pd.DataFrame(mycursor.fetchall())
                jsonfiles = json.loads(df.to_json(orient='records'))
                mycursor.close()
                return jsonify(jsonfiles)
            else:
                return jsonify({'data': "no"})
    def post(self):
        mydb.connect()
        if mydb.is_connected():
            mycursor = mydb.cursor()
            id_val = request.form.get('id_val')
            name_val = request.form.get('name_val')
            in_val = request.form.get('intime_val')
            out_val = request.form.get('outtime_val')
            sql = "INSERT INTO test (id, name, intime, outtime) VALUES
(%s, %s, %s, %s)"
            val = (id_val, name_val, in_val, out_val)
            mycursor.execute(sql, val)
            mydb.commit()
            mydb.close()
            return jsonify({'data': "sucessfully inserted"})
        else:
            return jsonify({'data': "no"})
    def put(self):
        mydb.connect()
        if mydb.is_connected():
            mycursor = mydb.cursor()
            id_val = request.form.get('id_val')
            name_val = request.form.get('name_val')
            in_val = request.form.get('intime_val')
            out_val = request.form.get('outtime_val')
            sql = "UPDATE test SET name=%s, intime=%s, outtime=%s WHERE
id=%s"
            val = (name_val, in_val, out_val, id_val)
            mycursor.execute(sql, val)
            mydb.commit()
```

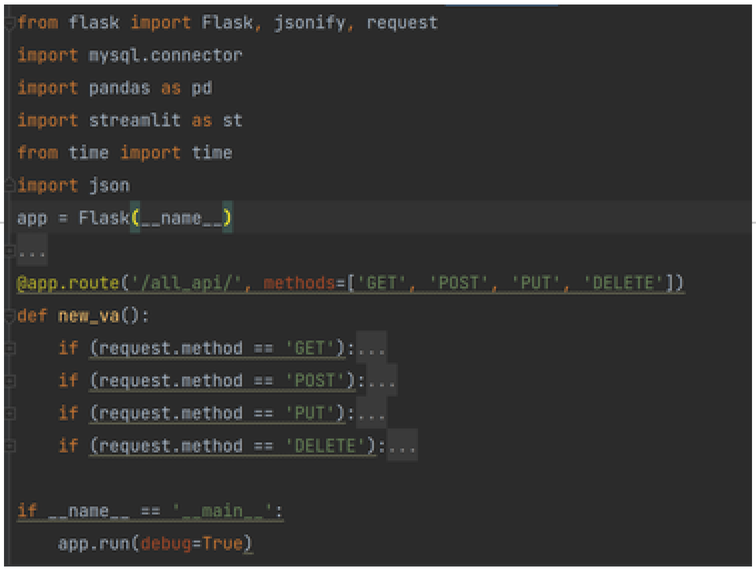
```
                mydb.close()
                respone = jsonify({'data': "updated successfully!"})
                respone.status_code = 200            return respone
            else:
                return jsonify({'data': "no"})
        def delete(self):
            num = request.args.get('name')
            mydb.connect()
            if mydb.is_connected():
                mycursor = mydb.cursor()
                mycursor.execute(f"DELETE FROM test WHERE id='{num}';")
                mydb.commit()
                mydb.close()
                respone = jsonify({'data': "Deleted successfully!"})
                respone.status_code = 200            return respone
            else:
                return jsonify({'data': "no"})

    class Square(Resource):
        def get(self, num):
            return jsonify({'square': num ** 2})

    api.add_resource(Hello, '/all_api/')
    api.add_resource(Square, '/square/<int:num>')

    if _name_ == '__main__':
        app.run(debug=True)
```

Both method different syntax

1. Method 2. Method



```
app = Flask(__name__)
api = Api(app)
...
class Hello(Resource):
    def get(self):...
    def post(self):...
    def put(self):...
    def delete(self):...

class Square(Resource):
    def get(self, num):
        return jsonify({'square': num ** 2})
api.add_resource(Hello, '/all_api/')
api.add_resource(Square, '/square/<int:num>')
if __name__ == '__main__':
    app.run(debug=True)
```



```
from flask import Flask, jsonify, request
import mysql.connector
import pandas as pd
import streamlit as st
from time import time
import json
app = Flask(__name__)
...
@app.route('/all_api/', methods=['GET', 'POST', 'PUT', 'DELETE'])
def new_va():
    if (request.method == 'GET'):...
    if (request.method == 'POST'):...
    if (request.method == 'PUT'):...
    if (request.method == 'DELETE'):...

if __name__ == '__main__':
    app.run(debug=True)
```

Flask with Mongodb

```
import pymongo
from flask import Flask, jsonify, request
import json
from bson.json_util import dumps

app = Flask(__name__)
myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mydatabase"]        # database namemycol = mydb
["customers"]            # collection name@app.route('/all_api/',
methods=['GET', 'POST', 'PUT', 'DELETE'])
def new_va():
    if (request.method == 'GET'):
        if not request.args.get('name'):
            result = [x for x in mycol.find({}, {"_id": 0, "name": 1,
"address": 1})]
            return jsonify({"News": result})
        else:
            num = request.args.get('name')
            mydoc = mycol.find({"name": num}, {"_id": 0, "name": 1,
"address": 1})
            result = [x for x in mydoc]
            return jsonify({"News": result})
    if (request.method == 'POST'):
        name_val = request.form.get('name')
        add_val = request.form.get('address')
        mydict = {"name": name_val, "address": add_val}
        mycol.insert_one(mydict)
        return jsonify({"News": "Inserted Successfully"})
    if (request.method == 'PUT'):
        name_val = request.form.get('name')
        add_val = request.form.get('address')
        mydoc = mycol.find({"name": name_val}, {"_id": 0, "name": 1,
"address": 1})
        result = [x for x in mydoc]
        myquery = {"name": result[0]['name'], "address": result[0]
['address']}
        newvalues = {"$set": {"name": name_val, "address": add_val}}
        mycol.update_one(myquery, newvalues)
        return jsonify({"News": "Updated Successfully"})
    if (request.method == 'DELETE'):
        num = request.args.get('name')
        myquery = {"name": num}
        mycol.delete_one(myquery)
        return jsonify({"News": "Deleted Successfully"})

if _name_ == '__main__':
    app.run(debug=True)
```

FASTAPI DOCUMENT

https://fastapi.tiangolo.com/tutorial/path-params/

Basic code without uvicorn including

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
async def root():
    return {"message": "Hello World"}
```

Run your terminal

uvicorn Fastapi_new_request:app --reload

Basic code with uvicorn including

```
from fastapi import FastAPI, Request
import uvicorn

app = FastAPI()

@app.get("/distance/")
async def check():
   return {'lat': 123}

if _name_ == '__main__':
    uvicorn.run('Fastapi_new_request:app', debug=True, reload=True)

### Second Method
from fastapi import FastAPI, File, UploadFile
import uvicorn
app = FastAPI()

@app.post("/chatbot_text")
async def analyze_route(input:str):
    try:
          res = input
          return {"result":res}
    except Exception as e:
        return {"Success": "false", "Result":str(e) }
if _name_ == '__main__':
    uvicorn.run('fast_face:app', port=8001, host='0.0.0.0',reload=True,
debug=True)
```

Path Parameters or Routing parameters Method

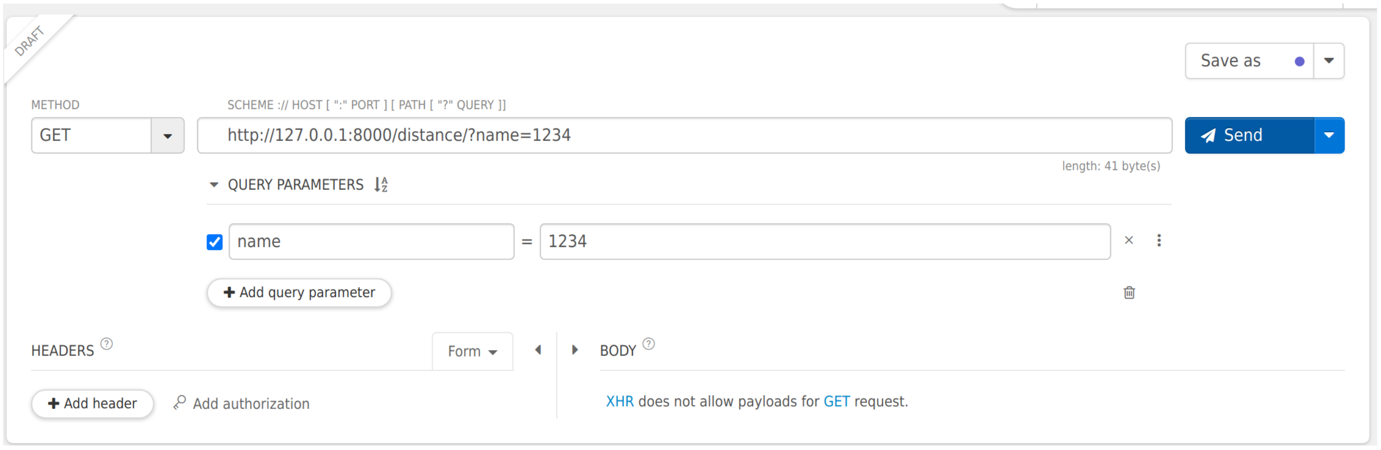
GET Method

ref = https://fastapi.tiangolo.com/tutorial/path-params/

1. **get single value** Ex : http://127.0.0.1:8000/distance/123

```
@app.get("/distance/{iten}")
async def check(iten):
    return {iten}
```

Query Parameters



http://127.0.0.1:8000/distance/?name=1234

name = 1234

XHR does not allow payloads for GET request.

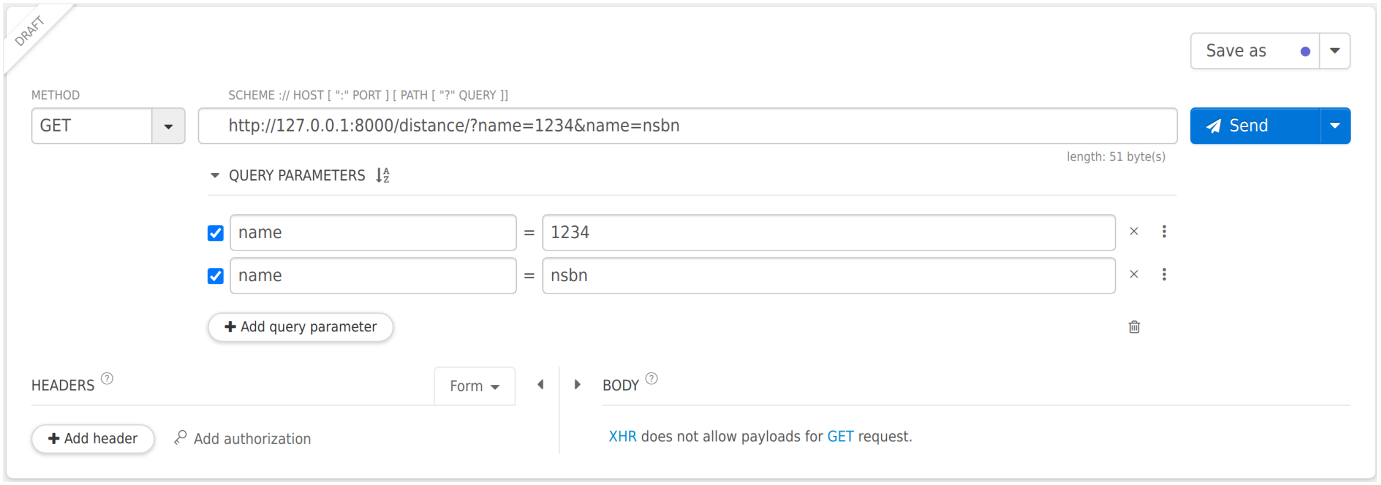
```
#get single value 2 method way 1 http://127.0.0.1:8000/distance/?
name=1234
************************************************************************
***
from fastapi import Request       ===> import

@app.get("/distance/")
async def check(name: float, request: Request):
    return {'lat': name }

#get single value 2 method way 2 http://127.0.0.1:8000/distance/?
name=1234
************************************************************************
***
# using query.params
@app.get("/distance/")
def check(request: Request):
    val = request.query_params['name']
    return val
```

Get Multiple (list) value method http://127.0.0.1:8000/distance/?name=1234&name=bar

please refer this website... the code will be different python version

ref == https://fastapi.tiangolo.com/tutorial/query-params-str-validations/#query-parameter-list-multiple-values

past and find =====>> http://localhost:8000/items/?q=foo&q=bar



http://127.0.0.1:8000/distance/?name=1234&name=nsbn

length: 51 byte(s)

name	1234
name	nsbn

XHR does not allow payloads for GET request.

CODE

```
import uvicorn
from typing import Union, List
from fastapi import FastAPI, Query

app = FastAPI()
@app.get("/distance/")
async def read_items(name: Union[List[str], None] = Query
(default=None)):
    query_items = {"q": name}
    return query_items

if _name_ == '__main__':
    uvicorn.run('Fastapi_new_request:app', debug=True, reload=True)
```

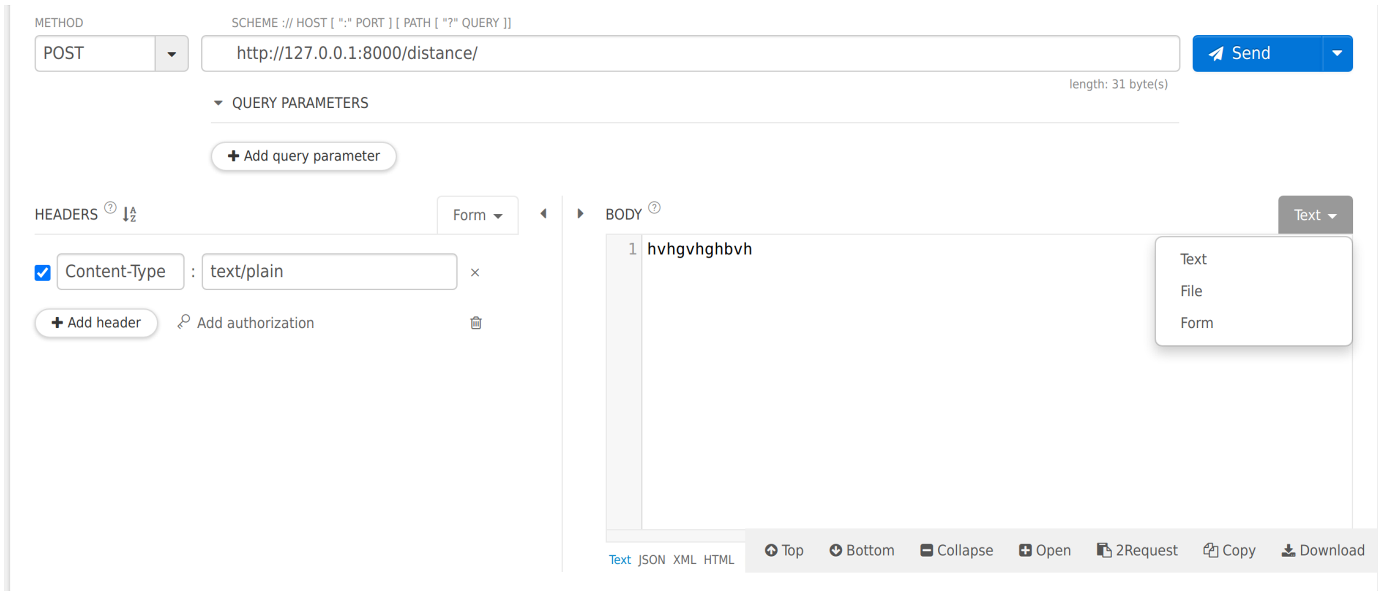
POST Method

How we going to know text and file and form post request

Text Method

we going to know about below image we have json & text & xml & Html

Fast api we don’t use text/plain everything is a json value


METHOD
POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]
http://127.0.0.1:8000/distance/

Send

length: 31 byte(s)

QUERY PARAMETERS

Add query parameter

HEADERS

Form

Content-Type : text/plain

Add header
Add authorization

BODY

Text

```
hvhgvhghbvh
```

Text
File
Form

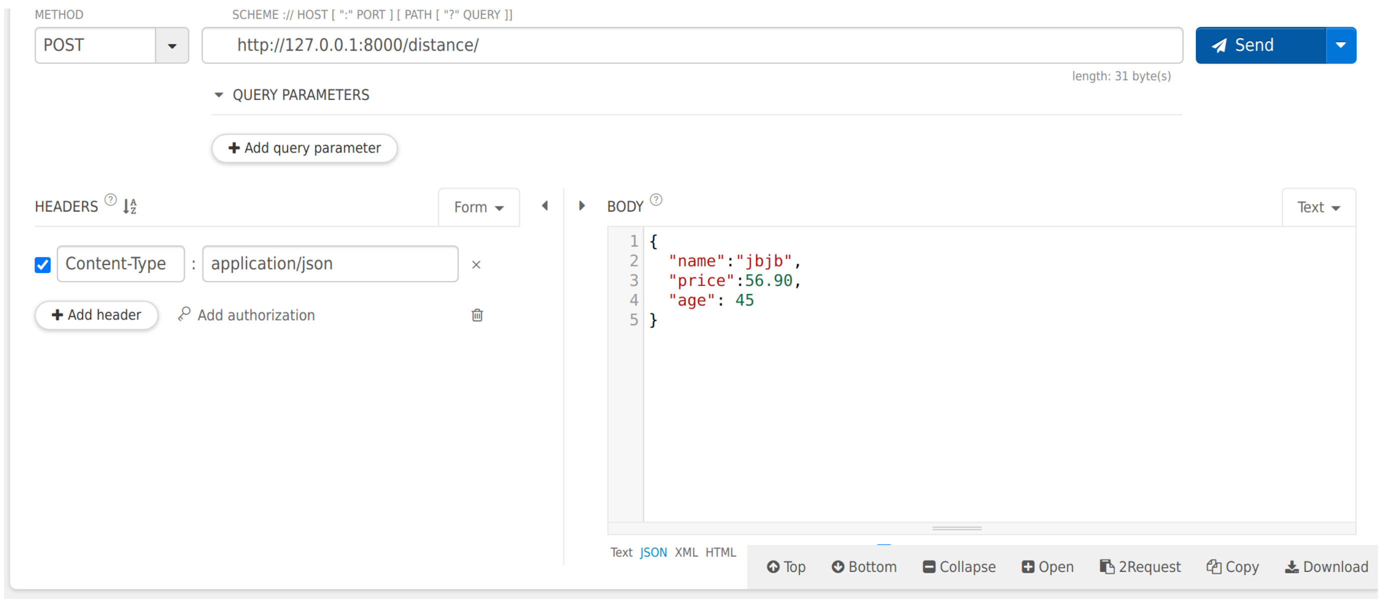
Top Bottom Collapse Open 2Request Copy Download

Text JSON XML HTML

Json code

```
from fastapi import FastAPI, Request
import uvicorn
from pydantic import BaseModel

class Item(BaseModel):
    name: str
    price: float
    age: int

app = FastAPI()

@app.post("/distance/")
async def create_item(item:Item):
    ad, asd, asdf = item.name, item.price, item.age
    return ad, asd, asdf
```

Json input



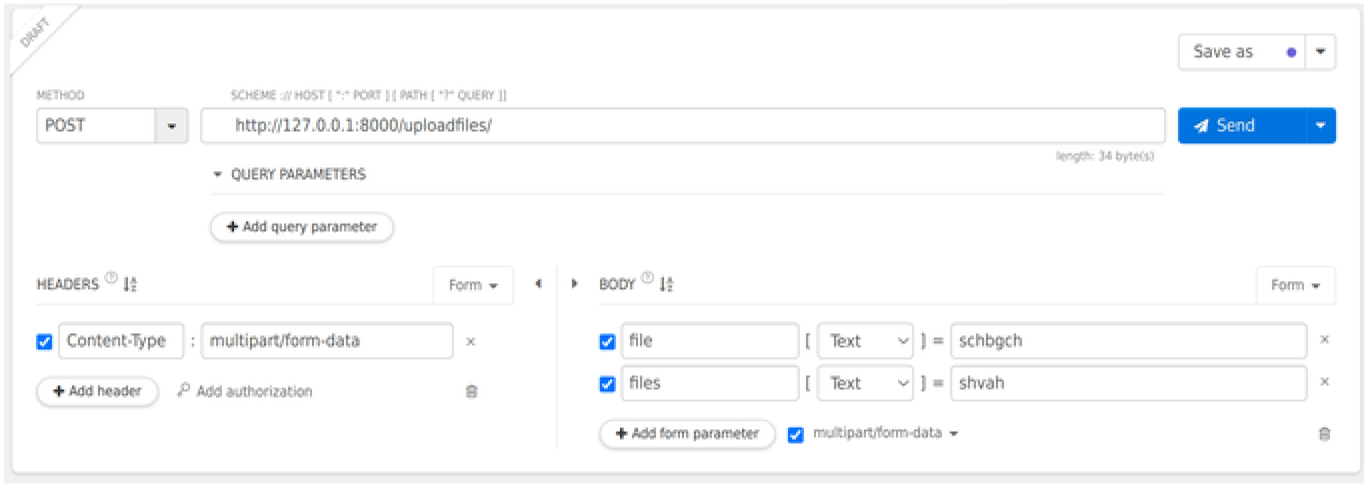
POST http://127.0.0.1:8000/distance/

length: 31 byte(s)

Content-Type : application/json

```
{
  "name":"jbjb",
  "price":56.90,
  "age": 45
}
```

Form Method



POST http://127.0.0.1:8000/uploadfiles/

length: 34 byte(s)

Content-Type : multipart/form-data

file [Text] = schbgch

files [Text] = shvah

multipart/form-data

CODE

```
from fastapi import FastAPI, Request, Form
import uvicorn
from pydantic import BaseModel

app = FastAPI()

@app.post("/login/")
async def login(username: str = Form(), password: str = Form()):
    return {username, password}

if _name_ == '__main__':
    uvicorn.run('Fastapi_new_request:app', debug=True, reload=True)
```

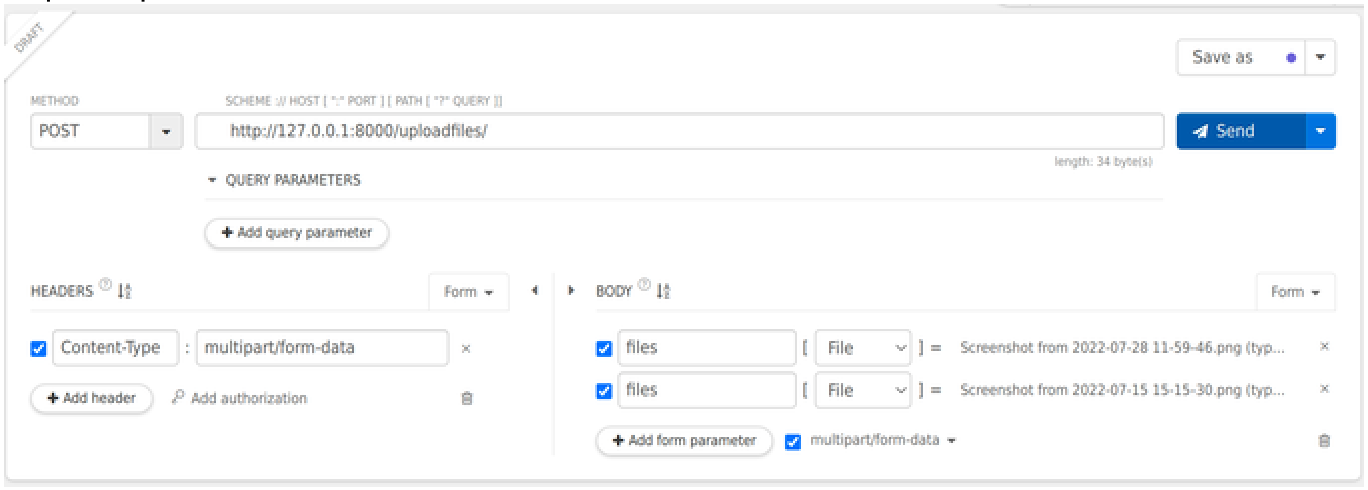
File uploading Method

ref == https://fastapi.tiangolo.com/tutorial/request-files/

Single file upload

```
import uvicorn
from pydantic import BaseModel
from fastapi import FastAPI, Request, Form, File, UploadFile

app = FastAPI()

@app.post("/files/")
async def create_file(file: bytes = File()):
    return {"file_size": len(file)}

@app.post("/uploadfile/")
async def create_upload_file(file: UploadFile):
    # print(file.file) ====>> using image processing ex opencv
    # print(file.filename)
    return {"filename": file.filename}

if _name_ == '__main__':
    uvicorn.run('Fastapi_new_request:app', debug=True, reload=True)
```

Multiple File Uploads



METHOD: POST
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://127.0.0.1:8000/uploadfiles/
length: 34 byte(s)
QUERY PARAMETERS

HEADERS
Content-Type : multipart/form-data

BODY
files [File] = Screenshot from 2022-07-28 11-59-46.png (typ...
files [File] = Screenshot from 2022-07-15 15-15-30.png (typ...
multipart/form-data

CODE

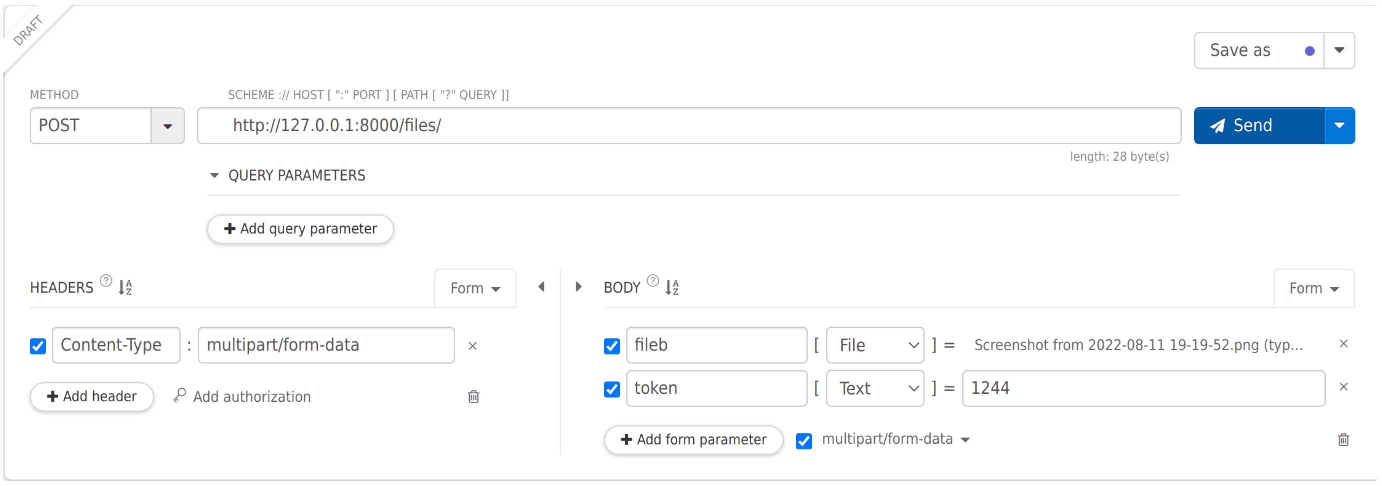
```
import uvicorn
from typing import List
from pydantic import BaseModel
from fastapi import FastAPI, Request, Form, File, UploadFile

app = FastAPI()

@app.post("/uploadfiles/")
async def create_upload_files(files: List[UploadFile]):   # files: List
[UploadFile] = File(...)
    return {"filenames": [file.filename for file in files]}

if _name_ == '__main__':
    uvicorn.run('Fastapi_new_request:app', debug=True, reload=True)
```

how to add both file and json body in a fastapi post request

ref =====>> https://stackoverflow.com/questions/65504438/how-to-add-both-file-and-json-body-in-a-fastapi-post-request

File and Form

```
import uvicorn
from fastapi import FastAPI, File, UploadFile, Form
from pydantic import BaseModel

app = FastAPI()

@app.post("/files/")
async def create_file(fileb: UploadFile = File(), token: str = Form()):
    return {
        "token": token,        "fileb_content_type": fileb.content_type
    }

if _name_ == '__main__':
    uvicorn.run('Fastapi_new_request:app', debug=True, reload=True)
```



DRAFT

Save as

METHOD

POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]

http://127.0.0.1:8000/files/

Send

length: 28 byte(s)

QUERY PARAMETERS

+ Add query parameter

HEADERS Form

✓	Content-Type	multipart/form-data

+ Add header Add authorization

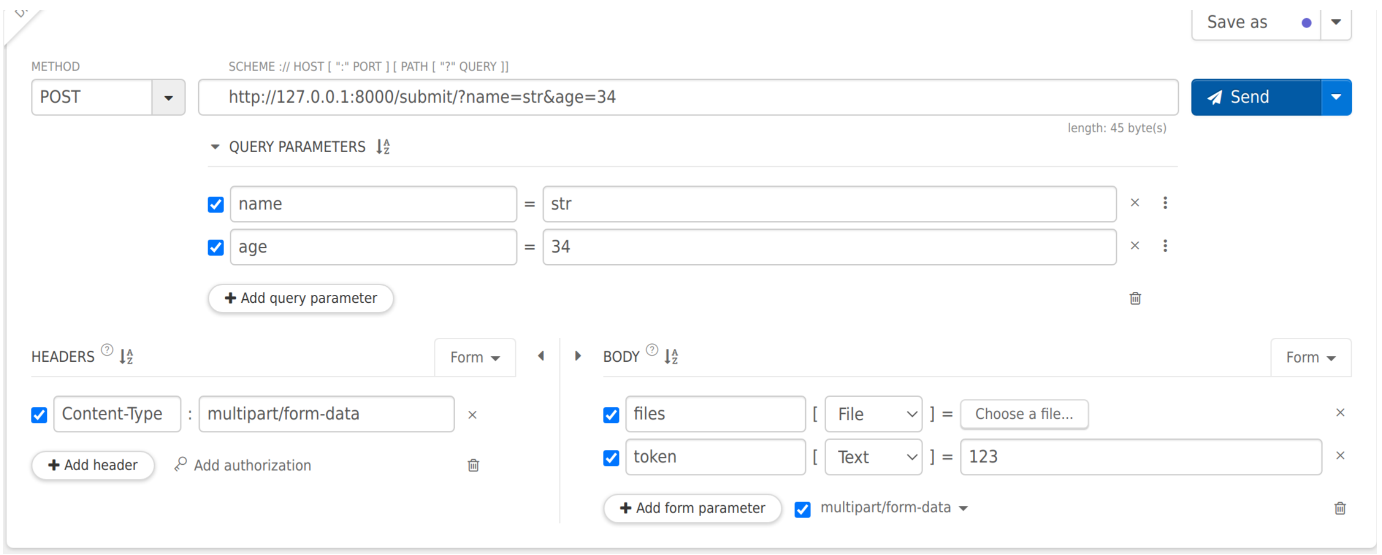
BODY Form

✓	fileb	File	Screenshot from 2022-08-11 19-19-52.png (typ...
✓	token	Text	1244

+ Add form parameter ✓ multipart/form-data

Form File with json

```
import uvicorn
from typing import List
from fastapi import FastAPI, File, UploadFile, Form, Depends
from pydantic import BaseModel

app = FastAPI()

class Base(BaseModel):
    name: str    age: int@app.post("/submit/")
async def submit(base: Base = Depends(), files: List[UploadFile] = File
(...),token: str = Form()):
    return {"JSON Payload ": base.dict(), "Filenames": [file.filename
for file in files], "token": token}

if _name_ == '__main__':
    uvicorn.run('Fastapi_new_request:app', debug=True, reload=True)
```



Save as

METHOD

POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]

http://127.0.0.1:8000/submit/?name=str&age=34

Send

length: 45 byte(s)

QUERY PARAMETERS

✓	name	str
✓	age	34

+ Add query parameter

HEADERS Form

✓	Content-Type	multipart/form-data

+ Add header Add authorization

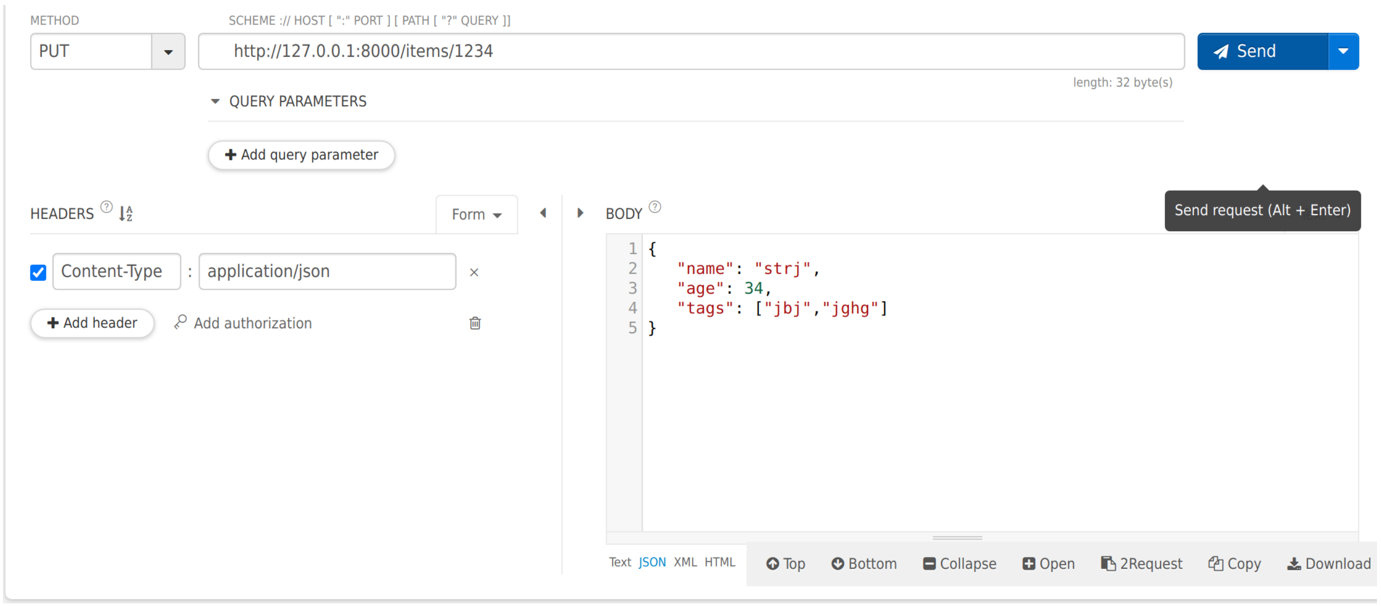
BODY Form

✓	files	File	Choose a file...
✓	token	Text	123

+ Add form parameter ✓ multipart/form-data

PUT Method

```
import uvicorn
from typing import List
from pydantic import BaseModel
from fastapi import FastAPI, Request, Form, File, UploadFile
app = FastAPI()

class Item(BaseModel):
    name: str    age: int    tags: list = []

@app.put("/items/{item_id}")
async def update_item(item_id: int, item: Item):
    results = {"item_id": item_id, "item": item}
    return results

if _name_ == '__main__':
    uvicorn.run('Fastapi_new_request:app', debug=True, reload=True)
```



PUT

http://127.0.0.1:8000/items/1234

length: 32 byte(s)

Content-Type : application/json

```
{
  "name": "strj",
  "age": 34,
  "tags": ["jbj","jghg"]
}
```

DELETE Method

```
import uvicorn
from fastapi import FastAPI
app = FastAPI()

@app.delete("/items/{item_id}")
async def update_item(item_id: int):
    return item_id

if _name_ == '__main__':
    uvicorn.run('Fastapi_new_request:app', debug=True, reload=True)
```

FASTAPI Restful APi

http://blog.adnansiddiqi.me/create-your-first-rest-api-in-fastapi/

```
import uvicorn
from typing import List
from fastapi import FastAPI, File, UploadFile, Form, Depends
from pydantic import BaseModel
import mysql.connector
import pandas as pd
import json

app = FastAPI()

mydb = mysql.connector.connect(
            host="localhost", port="3306",user="root",password="
root@123",database="face")

@app.get("/get_method/{num}")
async def get_met(num: int):
    mydb.connect()
    if mydb.is_connected():
        mycursor = mydb.cursor()
        mycursor.execute(f"select * from test where id='{num}';")
        df = pd.DataFrame(mycursor.fetchall())
        jsonfiles = json.loads(df.to_json(orient='records'))
        mycursor.close()
        return jsonfiles
    else:
        return {'data': "no"}

@app.get("/get_method/")
async def get_met():
    mydb.connect()
    if mydb.is_connected():
        mycursor = mydb.cursor()
        mycursor.execute("SELECT * FROM test;")
        df = pd.DataFrame(mycursor.fetchall())
        jsonfiles = json.loads(df.to_json(orient='records'))
        mycursor.close()
        mydb.close()
        return jsonfiles
    else:
        return {'data': "no"}

@app.post("/post_form/")
async def post_method(id_val: int = Form(), name_val: str = Form(),
intime_val: str = Form(), outtime_val: str = Form()):
    mydb.connect()
    if mydb.is_connected():
        mycursor = mydb.cursor()
        sql = "INSERT INTO test (id, name, intime, outtime) VALUES (%s,
%s, %s, %s)"
        val = (id_val, name_val, intime_val, outtime_val)
        mycursor.execute(sql, val)
        mydb.commit()
        mydb.close()
        return {'data': "sucessfully inserted"}
    else:
        return {'data': "no"}

@app.put("/put_method/{id_val}")
async def update_item(id_val: int, name_val: str = Form(), intime_val:
str = Form(), outtime_val: str = Form()):
    mydb.connect()
    if mydb.is_connected():
        mycursor = mydb.cursor()
        sql = "UPDATE test SET name=%s, intime=%s, outtime=%s WHERE id=%
s"
        val = (name_val, intime_val, outtime_val, id_val)
        mycursor.execute(sql, val)
        mydb.commit()
        mydb.close()
        respone = {'data': "updated successfully!"}
        return respone
    else:
        return {'data': "no"}

@app.delete("/del_method/{id_val}")
async def delete_item(id_val: int):
    mydb.connect()
    if mydb.is_connected():
        mycursor = mydb.cursor()
        mycursor.execute(f"DELETE FROM test WHERE id='{id_val}';")
        mydb.commit()
        mydb.close()
        respone = {'data': "Deleted successfully!"}
        return respone
    else:
        return jsonify({'data': "no"})

if _name_ == '__main__':
    uvicorn.run('Fastapi_new_request:app', debug=True, reload=True)
```

FastApi With Mongodb

```
import pymongo
import uvicorn
from typing import List
from fastapi import FastAPI, Form
import pandas as pd
import json

app = FastAPI()

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]        # database namemycol = mydb
["customers"]            # collection name@app.get("/get_method/{num}")
async def get_met(num: str):
    mydoc = mycol.find({"name": num}, {"_id": 0, "name": 1, "address":
1})
    result = [x for x in mydoc]
    return {"News": result}

@app.get("/get_method/")
async def get_met():
    result = [x for x in mycol.find({}, {"_id": 0, "name": 1,
"address": 1})]
    return {"News": result}

@app.post("/post_form/")
async def post_method(name_val: str = Form(), add_val: str = Form()):
    mydict = {"name": name_val, "address": add_val}
    mycol.insert_one(mydict)
    return {"News": "Inserted Successfully"}

@app.put("/put_method/{id_val}")
async def update_item(id_val: str, add_val: str = Form()):
    mydoc = mycol.find({"name": id_val}, {"_id": 0, "name": 1,
"address": 1})
    result = [x for x in mydoc]
    myquery = {"name": result[0]['name'], "address": result[0]
['address']}
    newvalues = {"$set": {"name": id_val, "address": add_val}}
    mycol.update_one(myquery, newvalues)
    return {'data': "updated successfully!"}

@app.delete("/del_method/{id_val}")
async def delete_item(id_val: str):
    myquery = {"name": id_val}
    mycol.delete_one(myquery)
    return {"News": "Deleted Successfully"}

if _name_ == '__main__':
    uvicorn.run('Fastapi_mongodb:app', debug=True, reload=True)
```