30 Days Of Python: Day 29 - Building an API

Building API

In this section, we will cove a RESTful API that uses HTTP request methods to GET, PUT, POST and DELETE data.

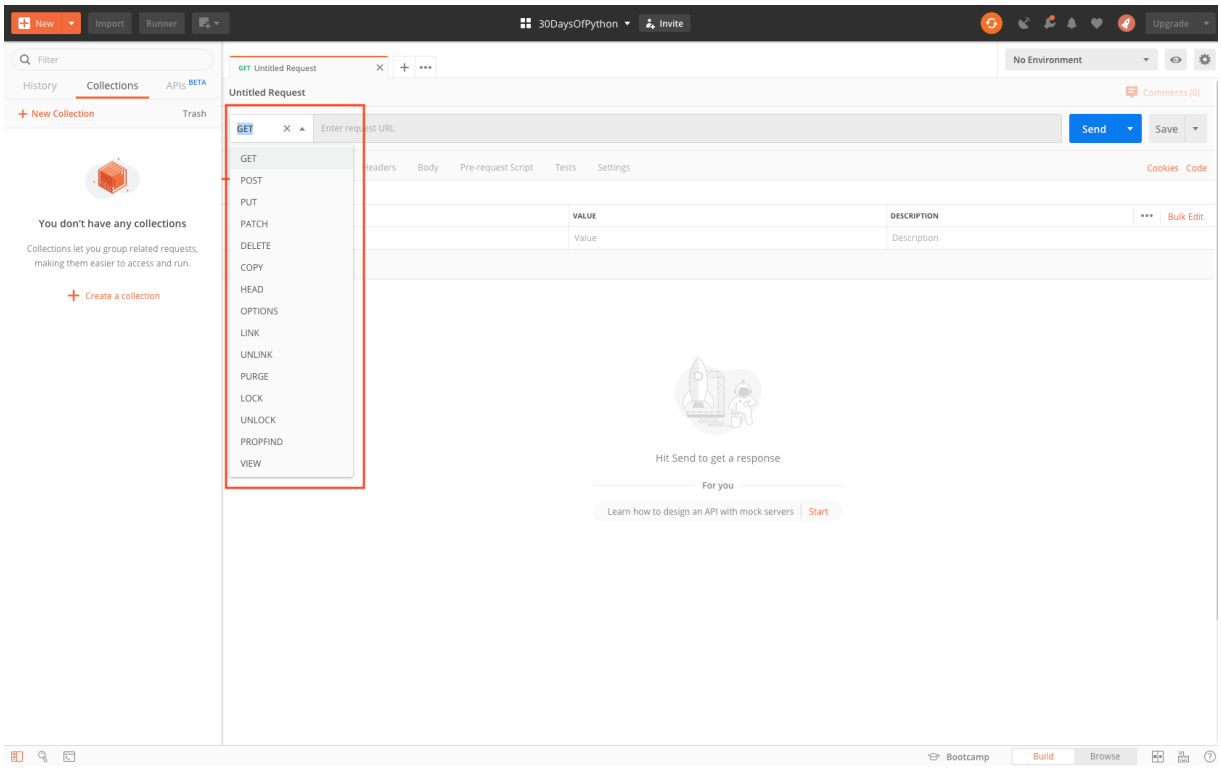
RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data. In the previous sections, we have learned about python, flask and mongoDB. We will use the knowledge we acquire to develop a RESTful API using python flask and mongoDB. Every application which has CRUD(Create, Read, Update, Delete) operation has an API to create data, to get data, to update data or to delete data from database.

The browser can handle only get request. Therefore, we have to have a tool which can help us to handle all request methods(GET, POST, PUT, DELETE).

Examples of API

- Countries API: https://restcountries.eu/rest/v2/all
- Cats breed API: https://api.thecatapi.com/v1/breeds

Postman is a very popular tool when it comes to API development. So, if you like to do this section you need to download postman. An alternative of Postman is Insomnia.



Structure of an API

An API end point is a URL which can help to retrieve, create, update or delete a resource. The structure looks like this:
Example: https://api.twitter.com/1.1/lists/members.json Returns the members of the specified list. Private list members will only be shown if the authenticated user owns the specified list. The name of the company name followed by version followed by the purpose of the API. The methods: HTTP methods & URLs

The API uses the following HTTP methods for object manipulation:

```
GET         Used for object retrieval
POST        Used for object creation and object actions
PUT         Used for object update
DELETE      Used for object deletion
```

Let us build an API which collects information about 30DaysOfPython students. We will collect the name, country, city, date of birth, skills and bio.

To implement this API, we will use:

- Postman
- Python

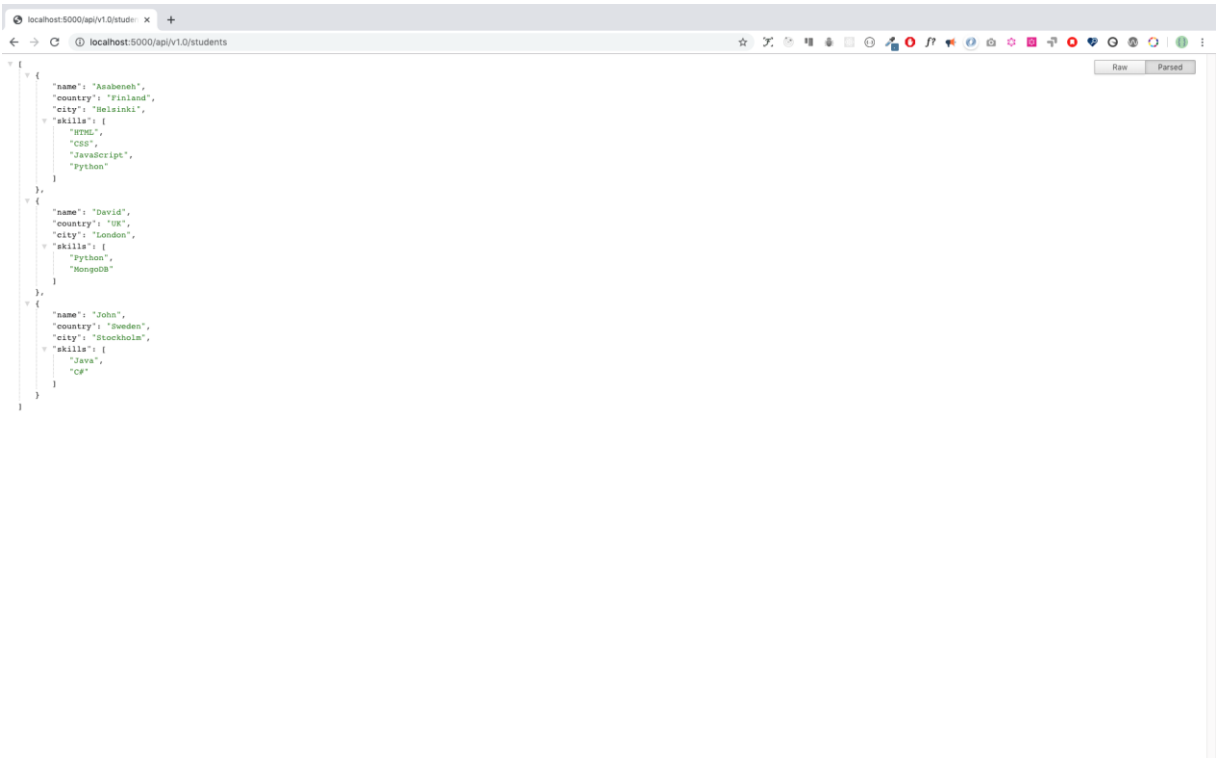
- Flask
- MongoDB

Retrieving data using get

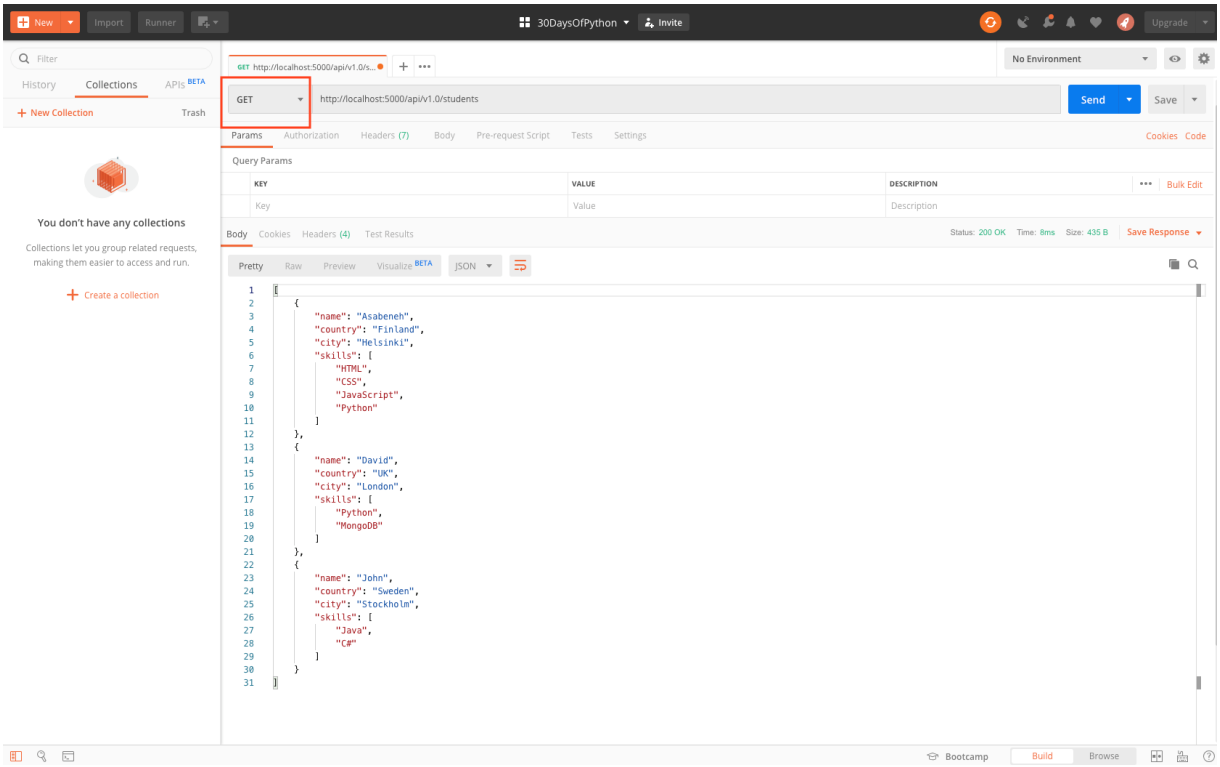
In this step, let us use dummy data and return it as a json. To return it as json, will use json module and Response module.

```
# let's import the flask

from flask import Flask,  Response
import json

app = Flask(__name__)

@app.route('/api/v1.0/students', methods = ['GET'])
def students ():
    student_list = [
        {
            'name':'Suniksha',
            'country':'Finland',
            'city':'Helsinki',
            'skills':['HTML', 'CSS','JavaScript','Python']
        },
        {
            'name':'David',
            'country':'UK',
            'city':'London',
            'skills':['Python','MongoDB']
        },
        {
            'name':'John',
            'country':'Sweden',
            'city':'Stockholm',
            'skills':['Java','C#']
        }
    ]
    return Response(json.dumps(student_list), mimetype='application/json')


if __name__ == '__main__':
    # for deployment
    # to make it work for both production and development
    port = int(os.environ.get("PORT", 5000))
    app.run(debug=True, host='0.0.0.0', port=port)
```

When you request the http://localhost:5000/api/v1.0/students url on the browser you will get this:



When you request the http://localhost:5000/api/v1.0/students url on the browser you will get this:



In stead of displaying dummy data let us connect the flask application with MongoDB and get data from mongoDB database.

```
# let's import the flask
```

```
from flask import Flask,  Response
import json
import pymongo


app = Flask(__name__)

#
MONGODB_URI='mongodb+srv://Suniksha:your_password@30daysofpython-
twxkr.mongodb.net/test?retryWrites=true&w=majority'
client = pymongo.MongoClient(MONGODB_URI)
db = client['thirty_days_of_python'] # accessing the database

@app.route('/api/v1.0/students', methods = ['GET'])
def students ():

    return Response(json.dumps(student), mimetype='application/json')


if __name__ == '__main__':
    # for deployment
    # to make it work for both production and development
    port = int(os.environ.get("PORT", 5000))
    app.run(debug=True, host='0.0.0.0', port=port)
```

By connecting the flask, we can fetch students collection data from the thirty_days_of_python database.

```
[
    {
        "_id": {
            "$oid": "5df68a21f106fe2d315bbc8b"
        },
        "name": "Suniksha",
        "country": "Finland",
        "city": "Helsinki",
        "age": 38
    },
    {
        "_id": {
            "$oid": "5df68a23f106fe2d315bbc8c"
        },
        "name": "David",
        "country": "UK",
        "city": "London",
        "age": 34
    },
    {
        "_id": {
            "$oid": "5df68a23f106fe2d315bbc8e"
        },
        "name": "Sami",
        "country": "Finland",
        "city": "Helsinki",
        "age": 25
    }
]
```

Getting a document by id

We can access signle document using an id, let's access Suniksha using his id. http://localhost:5000/api/v1.0/students/5df68a21f106fe2d315bbc8b

```
# let's import the flask

from flask import Flask,  Response
import json
from bson.objectid import ObjectId
import json
from bson.json_util import dumps
import pymongo


app = Flask(__name__)

#
MONGODB_URI='mongodb+srv://Suniksha:your_password@30daysofpython-
twxkr.mongodb.net/test?retryWrites=true&w=majority'
client = pymongo.MongoClient(MONGODB_URI)
db = client['thirty_days_of_python'] # accessing the database

@app.route('/api/v1.0/students', methods = ['GET'])
def students ():

    return Response(json.dumps(student), mimetype='application/json')
@app.route('/api/v1.0/students/<id>', methods = ['GET'])
def single_student (id):
    student = db.students.find({'_id':ObjectId(id)})
    return Response(dumps(student), mimetype='application/json')

if __name__ == '__main__':
    # for deployment
    # to make it work for both production and development
    port = int(os.environ.get("PORT", 5000))
    app.run(debug=True, host='0.0.0.0', port=port)
[
    {
        "_id": {
            "$oid": "5df68a21f106fe2d315bbc8b"
        },
        "name": "Suniksha",
        "country": "Finland",
        "city": "Helsinki",
        "age": 38
    }
]
```

Creating data using POST

We use the POST request method to create data

```
# let's import the flask

from flask import Flask,  Response
import json
```

```
from bson.objectid import ObjectId
import json
from bson.json_util import dumps
import pymongo
from datetime import datetime


app = Flask(__name__)

#
MONGODB_URI='mongodb+srv://Suniksha:your_password@30daysofpython-
twxkr.mongodb.net/test?retryWrites=true&w=majority'
client = pymongo.MongoClient(MONGODB_URI)
db = client['thirty_days_of_python'] # accessing the database

@app.route('/api/v1.0/students', methods = ['GET'])
def students ():

    return Response(json.dumps(student), mimetype='application/json')
@app.route('/api/v1.0/students/<id>', methods = ['GET'])
def single_student (id):
    student = db.students.find({'_id':ObjectId(id)})
    return Response(dumps(student), mimetype='application/json')
@app.route('/api/v1.0/students', methods = ['POST'])
def create_student ():
    name = request.form['name']
    country = request.form['country']
    city = request.form['city']
    skills = request.form['skills'].split(', ')
    bio = request.form['bio']
    birthyear = request.form['birthyear']
    created_at = datetime.now()
    student = {
        'name': name,
        'country': country,
        'city': city,
        'birthyear': birthyear,
        'skills': skills,
        'bio': bio,
        'created_at': created_at

    }
    db.students.insert_one(student)
    return ;
def update_student (id):
if __name__ == '__main__':
    # for deployment
    # to make it work for both production and development
    port = int(os.environ.get("PORT", 5000))
    app.run(debug=True, host='0.0.0.0', port=port)
```

Updating using PUT

```
# let's import the flask

from flask import Flask,  Response
import json
from bson.objectid import ObjectId
import json
```

```
from bson.json_util import dumps
import pymongo
from datetime import datetime


app = Flask(__name__)

#
MONGODB_URI='mongodb+srv://Suniksha:your_password@30daysofpython-
twxkr.mongodb.net/test?retryWrites=true&w=majority'
client = pymongo.MongoClient(MONGODB_URI)
db = client['thirty_days_of_python'] # accessing the database

@app.route('/api/v1.0/students', methods = ['GET'])
def students ():

    return Response(json.dumps(student), mimetype='application/json')
@app.route('/api/v1.0/students/<id>', methods = ['GET'])
def single_student (id):
    student = db.students.find({'_id':ObjectId(id)})
    return Response(dumps(student), mimetype='application/json')
@app.route('/api/v1.0/students', methods = ['POST'])
def create_student ():
    name = request.form['name']
    country = request.form['country']
    city = request.form['city']
    skills = request.form['skills'].split(', ')
    bio = request.form['bio']
    birthyear = request.form['birthyear']
    created_at = datetime.now()
    student = {
        'name': name,
        'country': country,
        'city': city,
        'birthyear': birthyear,
        'skills': skills,
        'bio': bio,
        'created_at': created_at

    }
    db.students.insert_one(student)
    return
@app.route('/api/v1.0/students/<id>', methods = ['PUT']) # this decorator
create the home route
def update_student (id):
    query = {"_id":ObjectId(id)}
    name = request.form['name']
    country = request.form['country']
    city = request.form['city']
    skills = request.form['skills'].split(', ')
    bio = request.form['bio']
    birthyear = request.form['birthyear']
    created_at = datetime.now()
    student = {
        'name': name,
        'country': country,
        'city': city,
        'birthyear': birthyear,
        'skills': skills,
        'bio': bio,
        'created_at': created_at
```

```
    }
    db.students.update_one(query, student)
    # return Response(dumps({"result":"a new student has been created"}),
mimetype='application/json')
    return
def update_student (id):
if __name__ == '__main__':
    # for deployment
    # to make it work for both production and development
    port = int(os.environ.get("PORT", 5000))
    app.run(debug=True, host='0.0.0.0', port=port)
```

Deleting a document using Delete

```
# let's import the flask

from flask import Flask,  Response
import json
from bson.objectid import ObjectId
import json
from bson.json_util import dumps
import pymongo
from datetime import datetime


app = Flask(__name__)

#
MONGODB_URI='mongodb+srv://Suniksha:your_password@30daysofpython-
twxkr.mongodb.net/test?retryWrites=true&w=majority'
client = pymongo.MongoClient(MONGODB_URI)
db = client['thirty_days_of_python'] # accessing the database

@app.route('/api/v1.0/students', methods = ['GET'])
def students ():

    return Response(json.dumps(student), mimetype='application/json')
@app.route('/api/v1.0/students/<id>', methods = ['GET'])
def single_student (id):
    student = db.students.find({'_id':ObjectId(id)})
    return Response(dumps(student), mimetype='application/json')
@app.route('/api/v1.0/students', methods = ['POST'])
def create_student ():
    name = request.form['name']
    country = request.form['country']
    city = request.form['city']
    skills = request.form['skills'].split(', ')
    bio = request.form['bio']
    birthyear = request.form['birthyear']
    created_at = datetime.now()
    student = {
        'name': name,
        'country': country,
        'city': city,
        'birthyear': birthyear,
        'skills': skills,
        'bio': bio,
        'created_at': created_at
```

```
    }
    db.students.insert_one(student)
    return
@app.route('/api/v1.0/students/<id>', methods = ['PUT']) # this decorator
create the home route
def update_student (id):
    query = {"_id":ObjectId(id)}
    name = request.form['name']
    country = request.form['country']
    city = request.form['city']
    skills = request.form['skills'].split(', ')
    bio = request.form['bio']
    birthyear = request.form['birthyear']
    created_at = datetime.now()
    student = {
        'name': name,
        'country': country,
        'city': city,
        'birthyear': birthyear,
        'skills': skills,
        'bio': bio,
        'created_at': created_at

    }
    db.students.update_one(query, student)
    # return Response(dumps({"result":"a new student has been created"}),
mimetype='application/json')
    return
@app.route('/api/v1.0/students/<id>', methods = ['PUT']) # this decorator
create the home route
def update_student (id):
    query = {"_id":ObjectId(id)}
    name = request.form['name']
    country = request.form['country']
    city = request.form['city']
    skills = request.form['skills'].split(', ')
    bio = request.form['bio']
    birthyear = request.form['birthyear']
    created_at = datetime.now()
    student = {
        'name': name,
        'country': country,
        'city': city,
        'birthyear': birthyear,
        'skills': skills,
        'bio': bio,
        'created_at': created_at

    }
    db.students.update_one(query, student)
    # return Response(dumps({"result":"a new student has been created"}),
mimetype='application/json')
    return ;
@app.route('/api/v1.0/students/<id>', methods = ['DELETE'])
def delete_student (id):
    db.students.delete_one({"_id":ObjectId(id)})
    return
if __name__ == '__main__':
    # for deployment
    # to make it work for both production and development
```

```
port = int(os.environ.get("PORT", 5000))
app.run(debug=True, host='0.0.0.0', port=port)
```

💻 Exercises: Day 29

1. Implement the above example and develop this

🎉 CONGRATULATIONS ! 🎉