Xml Assignmnet 1, Disi Pepiq, B00138946

Rest API for /getProducts

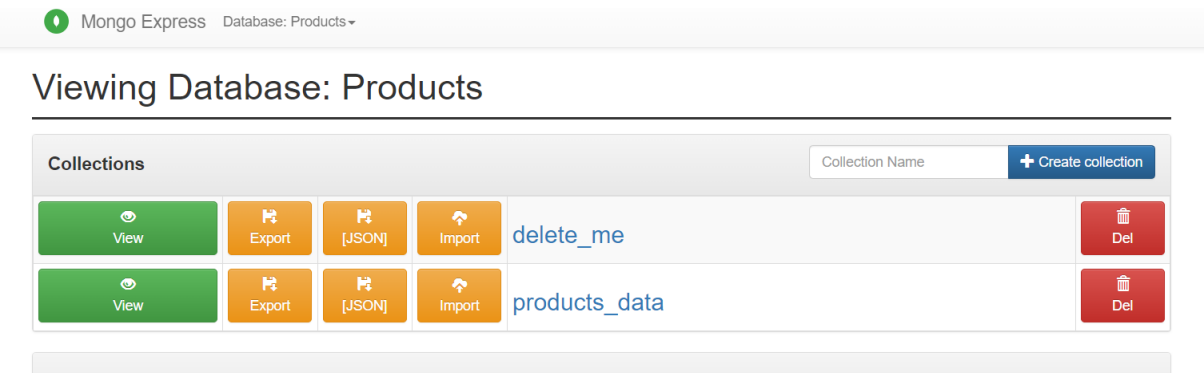
- So, I made python script called simple_api.py

```
from flask import Flask
from flask_restful import Resource, Api
import json
from bson.json_util import dumps, loads
from pymongo import MongoClient

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


class GetProducts(Resource):
  def get(self):
        client = MongoClient("mongodb://root:example@localhost:27017/")
        db = client.Products
        collection = db.products_data
        results = dumps(collection.find())
        print(results)
        return json.loads(results)
api.add_resource(GetProducts, '/getProducts')
```

- This python script makes a RESTful Api using flask and flask-RESTful to get the products from the MongoDB database.



- I created a Products Database and created collection called products_data in mongoDB
- In products_data I made a JSON list of products such as;

```
},
    "pname": "jam",
    "cost": "2.30"
  },
```

localhost:5000/getProducts

'retty-print


```
{
    "_id": {
        "$oid": "6616cee2b2ac89f2e2b33447"
    },
    "pname": "jam",
    "cost": "2.30"
},
{
    "_id": {
        "$oid": "6616cf05b2ac89f2e2b33449"
    },
    "pname": "cheese",
    "cost": "2.70"
}
```


- `/getProducts` endpoint prints the JSON list

Rest API for /getTitles

```python
# Define the schema
class Product(graphene.ObjectType):
    title = graphene.String()

# Define a query for GraphQL
class Query(graphene.ObjectType):
    products = graphene.List(Product)

    def resolve_products(root, info):
        # Fetch product titles from the /getTitles endpoint
        data = requests.get('http://localhost:5000/getTitles')
        json_content = data.json()
        titles = [item['pname'] for item in json_content]
        return [Product(title=title) for title in titles]


schema = graphene.Schema(query=Query)

#GetTitles Class
class GetTitles(Resource):
    def get(self):
        client = MongoClient("mongodb://root:example@localhost:27017/")
        db = client.Products
        collection = db.products_data
        results = dumps(collection.find({}, {"pname": 1, "_id": 0}))
        return json.loads(results)

api.add_resource(GetTitles, '/getTitles')
```

- This python script makes a RESTful Api using flask and flask-RESTful to get the product titles (pname) only from the MongoDB database.

localhost:5000/getTitles

Pretty-print


```
[
    {
        "pname": "jam"
    },
    {
        "pname": "cheese"
    }
]
```


- `/getTitles` endpoint prints the JSON list pname

Rest API for / and /instertProducts

```python
API_KEY = 'MY_APIKey'
# Function to check API key
def check_api_key(request):
    api_key = request.headers.get('API-Key')
    if api_key == API_KEY:
        return True
    else:
        return False

class Root(Resource):
    def get(self):
        return {
            ' Root page': [
                {'URL': '/getProducts', 'description':  'Returns a list of products from MongoDB.'},
                {'URL': '/getTitles' , 'description': 'Returns a list of product titles (pname) from MongoDB.'},
                {'URL': '/insertProduct', 'description':  'Inserts a new product into MongoDB.'}
            ]
        }

class InsertProduct(Resource):
    def post(self):

        data = request.get_json()
        _id = data.get('_id')
        pname = data.get('pname')
        cost = data.get('cost')

        new_record = {"_id": _id, "pname": pname, "cost": cost}
        self.db.products_data.insert_one(new_record)
        return {'status': 'inserted'}
api.add_resource(Root, '/')
api.add_resource(InsertProduct, '/insertProduct', methods=['POST'])
```

- This python script has Root class, where it provides a list of the API URLs
- InsertProduct endpoint inserts new products into MongoDB and Custom Api to authenticate the user before any product is inserted into the database.
- The InstertProduct Api also uses Postman and send across a id, product title and cost

DevOps Testing

I Tested the get products.

```python
import requests
f = open('test.log', 'w+')

def saveResult(name, url, result):
    f.write('Test name:' + str(name) + '\n')
    f.write('Test URL:' + str(url) + '\n')
    f.write('Test result:' + str(result) + '\n')
    f.write('--------------------------------------------\n ')


def checkServiceForWord(url, keyword):
    result = False
    try:
        x = requests.get(url)
        print(x.text)
        serverStatus=1
        if keyword in x.text:
            print("found keyword")
            result=True
    except:
            print("error")
            result= False
    return result




# Test 1
name = 'Test 1'
url = 'http://localhost:5000/getProducts'
result = checkServiceForWord(url, 'jam')
saveResult(name, url, result)
```

- So, I made python script called Test1Api.py
- The purpose of this script is to test the API endpoint of getProducts by checking if certain keywords are found in the response.
- This configuration allows for automated testing of the API endpoint

```
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 116-394-279
[
    {
        "_id": {
            "$oid": "65fc18c1a8d77a64ad3a786c"
        },
        "pname": "jam",
        "cost": "2.30"
    }
]

found keyword

C:\ProgramData\Jenkins\.jenkins\workspace\Freestyle job>exit 0
Process leaked file descriptors. See https://www.jenkins.io/redirect/troubleshooting/process-leaked-file-descriptors for more information
Archiving artifacts
Finished: SUCCESS
```

- So, in my Jenkins console output shows that the test was successfully and the found keyword ‘jam’