Τμήμα Ψηφιακών Συστημάτων
Πανεπιστήμιο Πειραιώς

Πληροφοριακά Συστήματα - Εργαστήριο 3ο

Χρυσόστομος Συμβουλίδης, simvoul@unipi.gr
Jean-Didier Totow, totow@unipi.gr

Πίνακας περιεχομένων

MongoDB

- Μία από τις πιο γνωστές NoSQL βάσεις δεδομένων
- **Document-based**: Τα δεδομένα αποθηκεύονται σαν **αντικείμενα (objects)**
- Η δομή δεδομένων που χρησιμοποιείται αναπαριστά τα αντικείμενα σαν ζεύγη πεδίων και τιμών και τα αποθηκεύει σε συλλογές (collections).
    - Τα **αντικείμενα (objects)** είναι ανάλογα με τις **εγγραφές** σε μία σχεσιακή βάση δεδομένων
    - Οι **συλλογές (collections)** είναι ανάλογες με τους **πίνακες** σε μία σχεσιακή βάση δεδομένων
    - Τα **πεδία** είναι αντίστοιχα με τις **στήλες (columns)** σε μία σχεσιακή βάση δεδομένων
    - Τιμές με διάφορους τύπους δεδομένων: αντικείμενα, πίνακες ή και πίνακες από αντικείμενα
    - Σε κάθε εγγραφή υπάρχει πάντα ένα **μοναδικό πρωτεύον κλειδί (Primary key)** που χρησιμοποιείται σαν αναγνωριστικό (`_id`)
- Αποθηκεύει **JSON(JavaScript Object Notation)-like** αρχεία **(BSON - Binary JSON)**
- Χρησιμοποιεί τη δική της query language που μας επιτρέπει:
    - Φιλτράρισμα των αποτελεσμάτων
    - Aggregation βάσει κειμένου, τοποθεσίας, κλπ
- Mongo Shell γραμμένο σε JavaScript
- Η βιβλιοθήκη **PyMongo** για τη Python μπορεί να χρησιμοποιηθεί για να υλοποιήσουμε όλες τις **CRUD (Create, Read, Update, Delete)** λειτουργίες που θέλουμε, στη βάση δεδομένων
- Χαρακτηρίζεται σαν μία **schema-less** βάση δεδομένων αφού δεν υπάρχει αυστηρό σχήμα στα collection

Πλεονεκτήματα και Μειονεκτήματα

Πλεονεκτήματα:

- Ευκολία στην εισαγωγή πολύπλοκων αντικειμένων Query language με σχετικά εύκολο συντακτικό
- Ελευθερία στην εισαγωγή αντικειμένων με διαφορετικά πεδία
- Πολύ καλύτερη από σχεσιακές βάσεις στο scaling

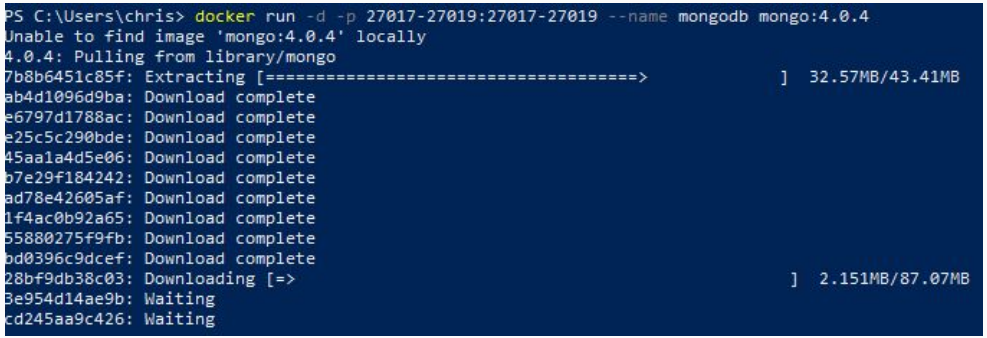
Μειονεκτήματα:

- Δεν υποστηρίζονται transactions (δηλαδή ακολουθίες από query)
- Ταχύτητα και αποδοτικότητα μόνο με τα σωστά indexes
- Δυσκολία στη δημιουργία join

MongoDB Docker container

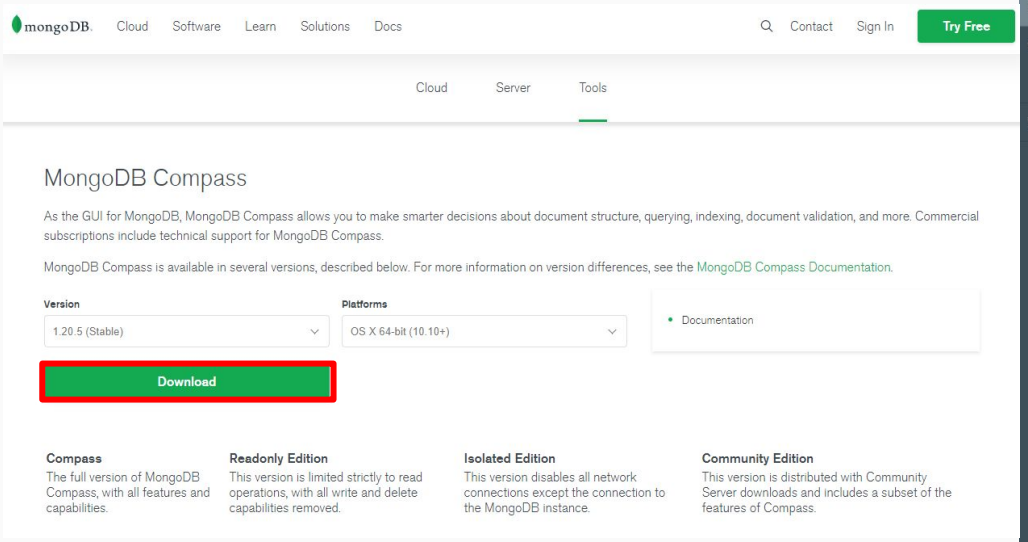
Ανοίγουμε το Powershell / Terminal:

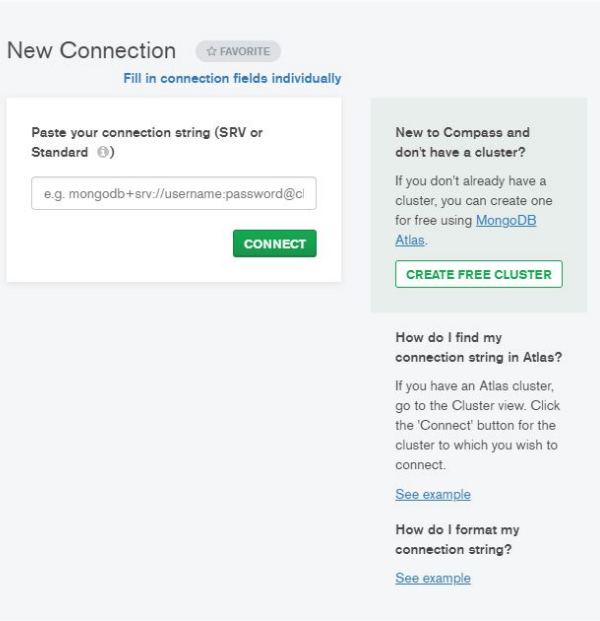
1. Για να κάνουμε pull το image από το Docker Hub:
   `(sudo) docker pull mongo`
2. Κατέβασμα της latest version του MongoDB image. Αν θέλουμε κάποια συγκεκριμένα version μπορούμε να τη κατεβάσουμε έτσι:
   `(sudo) docker pull mongo:4.0.4`
3. Για να κάνουμε deploy το image για πρώτη φορά:
   `(sudo) docker run -d -p 27017:27017 --name mongodb mongo:4.0.4`
4. Για να ξεκινήσουμε πάλι το container
   `(sudo) docker ps -a` // Επιλέγουμε το ID ή το όνομα του image που θέλουμε να ξεκινήσουμε
   `(sudo) docker start mongodb` // Για να ξεκινήσουμε το image
5. Για να σταματήσουμε το container
   `(sudo) docker stop mongodb`
6. Για να χρησιμοποιήσουμε το mongo shell
   `(sudo) docker exec -it mongodb mongo`
7. Για να χρησιμοποιήσουμε το mongo shell μίας mongo απομακρυσμένα
   `mongo --host IP:PORT` (π.χ. `mongo host --127.0.0.1:27017`)



MongoDB Compass

https://www.mongodb.com/download-center/compass





Mongo Shell (1/4)

Εντολές Mongo Shell:

- `show dbs` // Εμφάνιση λίστας των διαθέσιμων βάσεων δεδομένων
- `use db_name` // Χρήση της βάσης δεδομένων με το όνομα db_name (Αν δεν υπάρχει θα δημιουργηθεί εκείνη τη στιγμή)
- `db` // Εμφάνιση του ονόματος της βάσης δεδομένων που χρησιμοποιούμε
- `cls` // Καθαρισμός οθόνης

```
> use InfoSys
switched to db InfoSys
> db
InfoSys
```

```
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017/
MongoDB server version: 4.0.4
WARNING: shell and server versions do not match
Server has startup warnings:
2020-04-17T09:40:36.647+0000 I STORAGE  [initandlisten]
2020-04-17T09:40:36.647+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the
WiredTiger storage engine
2020-04-17T09:40:36.647+0000 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2020-04-17T09:40:38.416+0000 I CONTROL  [initandlisten]
2020-04-17T09:40:38.416+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-04-17T09:40:38.416+0000 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestr
icted.
2020-04-17T09:40:38.416+0000 I CONTROL  [initandlisten]
> dbs
2020-04-17T16:52:33.315+0300 E QUERY    [thread1] ReferenceError: dbs is not defined :
@(shell):1:1
> show dbs
InfoSys  0.000GB
admin    0.000GB
config   0.000GB
local    0.000GB
```

Mongo Shell (2/4)

Στο terminal γράφουμε τη παρακάτω εντολή για να εισάγουμε το `students.json` στη collection `Students` της βάσης `InfoSys`:

1. Κάνουμε copy τα δεδομένα από τον host στο container:
   ```
   docker cp students.json mongodb:/students.json
   ```
2. Εκτελούμε την εντολή:
   ```
   docker exec -it mongodb mongoimport --db=InfoSys --collection=Students --file=students.json
   ```

```
2020-04-24T10:29:53.511+0000    connected to: localhost
2020-04-24T10:29:53.542+0000    imported 2 documents
```

Αν είναι αρχείο διαφορετικού τύπου γράφουμε το ανάλογο type στο argument `--type`:

- `json`, για json
- `csv`, για csv (Comma Separated Values)
- `tsv` , για tsv (Tab Separated Values)

Mongo Shell (3/4)

Αναζήτηση αντικειμένων σε συλλογή (collection):

- `db.Students.find({})` //Εμφάνιση όλων των object του collection Students
- `db.Students.find({name: 'Michalis'})` // Εμφάνιση όλων των object που έχουν τη τιμή `'Alex'` στο key `'name'`
- `db.Students.findOne({name: 'Michalis'})` //Εμφάνιση μόνο του πρώτου object με τιμή `'Alex'` στο key `'name'`

Εισαγωγή αντικειμένων σε συλλογή:

- `db.Students.insertOne({name: 'Michalis Markou', email: 'markou@gmail.com'})` // Εισαγωγή ενός object
- `db.Students.insertMany([{name: 'Michalis Markou'}, {name: 'Giorgos Pantelopoulos'}])` // Εισαγωγή πολλών object στο collection

```
> db.Students.find().pretty()
{
        "_id" : ObjectId("5e80810aa066808a46221eaf"),
        "email" : "markou@gmail.com",
        "name" : "Michalis Markou",
        "yearOfBirth" : 1993
}
{
        "_id" : ObjectId("5e81ed3effeaee7ddf14769d"),
        "email" : "pantel@yahoo.com",
        "name" : "Giorgos Pantelopoulos",
        "yearOfBirth" : 1986
}
```

Mongo Shell (4/4)

Ενημέρωση αντικειμένου:

- Update One / Update many: Βρίσκει ένα ή περισσότερα αντικείμενα και τα ενημερώνει:
  - `db.Students.updateOne({name: 'Michalis Markou'}, {$set: {'yearOfBirth': 1993}})`
  - `db.Students.updateMany({name: 'Michalis Markou'}, {$set: {'yearOfBirth': 1993}})`
- Find One and Update: Βρίσκει ένα αντικείμενο, το ενημερώνει και το επιστρέφει πίσω:
  `db.Students.findOneAndUpdate({name: 'Michalis Markou'}, {$set: {'yearOfBirth': 1993}})`
- Find and Modify: Βρίσκει ένα object (το πρώτο), το επιστρέφει πίσω και το ενημερώνει ή/και διαγράφει και:
  - `db.Students.findAndModify(query: {name: 'Michalis Markou'}, {'yearOfBirth': 1993})`

Διαγραφή αντικειμένου:

- Delete One: Βρίσκει ένα object και το διαγράφει:
  - `db.Students.deleteOne({'email': 'name@mail.com'})`
- Delete Many : Βρίσκει ένα ή περισσότερα object και τα διαγράφει:
  - `db.Students.deleteMany({'email': 'name@mail.com'})`

Διαχείριση συλλογών (collection):

- Δημιουργία συλλογής:
  - Αρκεί να την καλέσουμε με κάποια εντολή για να δημιουργηθεί (όμοια με τις Βάσεις)
  - Όταν κάνουμε `insertOne()` το πρώτο αντικείμενο, δημιουργείται αυτόματα
- Διαγραφή συλλογής:
  - `db.Students.drop()`

Mongo - Τελεστές

Τελεστές ερωτημάτων και προβολών:

- $and // Λογικό ΚΑΙ
- $or // Λογικό Ή
- $not // Λογική Άρνηση
- $eq // Ίσο με
- $gt(e) // Μεγαλύτερο (ή ίσο) από
- $lt(e) // Μικρότερο (ή ίσο) από
- $ne // Διάφορο
- ...

Τελεστές ενημέρωσης αντικειμένων:

- $set // Ανάθεση τιμής
- $unsetor // Αφαίρεση τιμής
- $rename // Μετονομασία κλειδιού
- $inc // Αύξηση τιμής κατά
- $mul // Πολλαπλασιασμός τιμής
- ...

Περισσότερα για τους τελεστές μπορείτε να βρείτε εδώ:
https://docs.mongodb.com/manual/reference/operator/

PyMongo (1/3)

- Βιβλιοθήκη της Python η οποία μας επιτρέπει να αλληλεπιδρούμε με βάσεις δεδομένων MongoDB.
  - PyPI: https://pypi.org/project/pymongo/
  - Documentation: https://api.mongodb.com/python/current/api/index.html
- Μπορούμε να το εγκαταστήσουμε με τη παρακάτω εντολή:

```
pip install pymongo
```

- Αφού έχουμε κάνει activate το virtual environment

PyMongo (2/3)

- Εισαγωγή του pymongo package:
  - `import pymongo`
  - `from pymongo import MongoClient`
- Δημιουργία αντικειμένου `MongoClient()`:
  - `client = MongoClient('url_to_mongodb:27017')`
    Πχ: `client = MongoClient('localhost:27017')`
- Επιλογή βάσης δεδομένων:
  - `db = client['DB_name']`
- Επιλογή collection:
  - `collection = db['Collection_name']`
- Εισαγωγή αντικειμένου στο collection:
  - `collection.insert_one({'key_1': 'string_value', 'key_2':1993})`
  - `collection.insert_many({'key_1': 'value_1'}, {'key_1':'value_2'})`
- Εισαγωγή αντικειμένων σε collection από αρχείο:

```
import json
with open('file.json') as json_file:
     data = json.load(json_file)

collection.insert_many(json_file)
```

PyMongo (3/3)

- Αναζήτηση:
  - `collection.find()`
  - `collection.find_one({'key': 'value'})`
- Ενημέρωση:
  - `collection.update_one({'key': 'value'}, {'$set': {'key': 'different value'}})`
  - `collection.update_many({'key': 'value'}, {'$set': {'key': 'different value'}})`
  - `collection.find_one_and_update({'key': 'value'}, {'$set': {'key': 'different value'}})`
  - ```
    collection.find_and_modify(query={'key': 'value'},
                               update={'$set': {'key': 'different value'}})
    ```
- Διαχείριση συλλογών:
  - Δημιουργία συλλογής: Αρκεί να τη χρησιμοποιήσουμε για να δημιουργηθεί:
    `collection.insert_one({'key_1': 'string_value', 'key_2':1993})`
  - Διαγραφή συλλογής:
    `collection.drop()`
    ```

Web Services

Ως υπηρεσίες ιστού (**Web services**) μπορούν γενικά να θεωρούνται οι λειτουργίες μιας εφαρμογής, ή ακόμα και μια ολόκληρη εφαρμογή η οποία χρησιμοποιείται μέσα από το διαδίκτυο

- Στέλνουν και λαμβάνουν τυποποιημένα μηνύματα με άλλες εφαρμογές / web service χρησιμοποιώντας τεχνολογίες όπως:
  - Application Programming Interface (API)
  - Άλλους τρόπους επικοινωνίας όπως Publish / Subscribe συστήματα, κλπ

Μέθοδοι HTTP (HTTP Methods - 1/3)

POST:

- Χρησιμοποιείται για τη δημιουργία ή/και την ενημέρωση των δεδομένων.
- Μετά την ενημέρωση πληροφορίας με τη χρήση POST, πρέπει να καλείται η GET ώστε να επιβεβαιώνεται η σωστή ενημέρωση.
- Υλοποιεί τις λειτουργίες «C» (Create) και «U» (Update) του «CRUD».

PUT:

- Χρησιμοποιείται για τη μετατροπή ή/και εισαγωγή πληροφορίας στο server.
- Η διαφορά με το POST είναι ότι το PUT είναι σταθερό, δηλαδή ότι όσες φορές και να το καλέσουμε, το αποτέλεσμα πρέπει να παραμένει ίδιο.
- Μετά την ενημέρωση πληροφορίας με τη χρήση PUT, πρέπει να καλείται η GET και να επιστρέφει τα νέα δεδομένα.
- Υλοποιεί τη λειτουργία «U» (Update) του «CRUD».

Μέθοδοι HTTP (HTTP Methods - 2/3)

GET:

- Χρησιμοποιείται για να λάβουμε πληροφορίες από το server.
- Δεν μετατρέπουμε / προσθέτουμε / αφαιρούμε πληροφορία.
- Υλοποιεί την λειτουργία «R» (Read) του «CRUD».

PATCH:

- Χρησιμοποιείται μόνο για μερική μετατροπή δεδομένων στο server.
- Υλοποιεί τη λειτουργία «U» (Update) του «CRUD».

DELETE:

- Χρησιμοποιείται για τη διαγραφή δεδομένων από το server.
- Υλοποιεί την λειτουργία «D» (Delete) του «CRUD».

Μέθοδοι HTTP (HTTP Methods - 3/3)

Άλλες μέθοδοι είναι οι:

- HEAD: Παρόμοιο με το GET αλλά χωρίς να επιστρέφει το body
- OPTIONS: Επιστρέφει τις δυνατές μεθόδους / request που υποστηρίζει ο server
- ...

Αλλά δεν θα τις χρησιμοποιήσουμε στο συγκεκριμένο εργαστήριο.
Μία λίστα με όλες τις μεθόδους HTTP που υπάρχουν μπορείτε να δείτε εδώ:
http://www.iana.org/assignments/http-methods/http-methods.xhtml

Κωδικοί HTTP (HTTP Status/Response codes)

- 1XX: Ενημερωτικοί - Informational
- 2XX: Επιτυχία - Success
  - 200: OK
  - 201: Created
  - 202: Accepted
  - 204: No Content
- 3XX: Ανακατεύθυνση - Redirectional
- 4XX: Πρόβλημα από τη πλευρά του client - Client Error
  - 400: Bad Request
  - 404: Not Found
  - 405: Method not Allowed
- 5XX: Πρόβλημα από τη πλευρά του server - Server Error
  - 500: Internal Server Error

Μία λίστα με όλους τους HTTP status code μπορείτε να βρείτε εδώ: https://httpstatuses.com/

Flask microframework

- Παρέχει βιβλιοθήκες και εργαλεία με τα οποία μπορούμε υλοποιήσουμε web applications
- Pallets Projects: Συλλογή από web development βιβλιοθήκες για τη Python
  - Flask
  - Jinja
  - Click: CLI
  - ...
- Documentation: https://flask.palletsprojects.com/en/1.1.x/
- Γιατί microframework;
  - Γιατί δεν απαιτεί συγκεκριμένες βιβλιοθήκες και πακέτα για να τρέξει (πχ: MariaDB)

Θα χρησιμοποιηθεί για να δημιουργήσουμε τα API με τα οποία θα επικοινωνούν τα web service που θα υλοποιήσουμε μεταξύ τους.

Εγκατάσταση Flask

Αφού έχουμε κάνει **activate το περιβάλλον** μας στο οποίο θέλουμε να εγκαταστήσουμε το Flask:

- `conda install flask`

Εναλλακτικά:

- `pip install flask`

Flask - Παράδειγμα 1

example_1.py

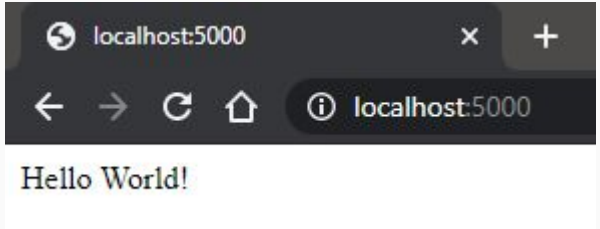
```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

@app.route('/<name>')
def hello_name(name):
    return f'Hello {name}!\n'

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

python example_1.py

```
(s-ehr-cloud) csymvoul@DESKTOP-4HT3TL7:~/projects/infosyslabs/lab1$ python web_service_example_1.py
 * Serving Flask app "web_service_example_1" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 252-170-097
```

curl localhost:5000/

```
(s-ehr-cloud) csymvoul@DESKTOP-4HT3TL7:~/projects/infosyslabs/lab1$ curl localhost:5000/
  Hello World!
```

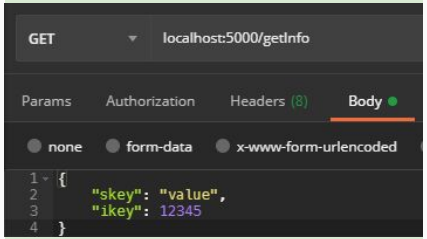


Flask - Παράδειγμα 2

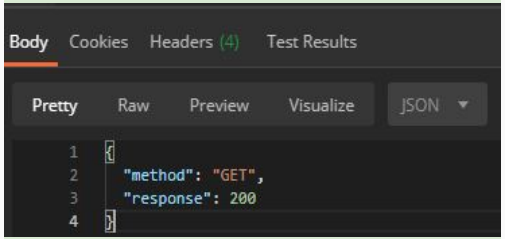
example_2.py

```
from flask import Flask, request
import json

app = Flask(__name__)
@app.route('/getInfo', methods=['GET','POST'])
def get_info():
    if request.method =='GET':
        return {'response': 200, 'method': 'GET'}
    elif request.method == 'POST':
        data = json.loads(request.data)
        data['method']= 'POST'
        data['response']=201
        return data

if __name__ == '__main__':
    app.run(debug=True, host= '0.0.0.0', port=5000)
```

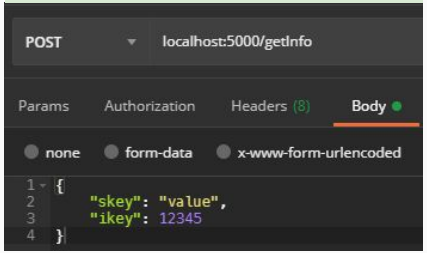
GET Request



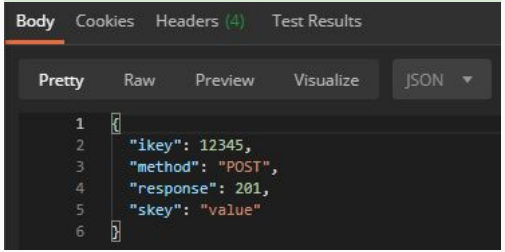
GET Response



POST Request



POST Response



Flask - PyMongo (1/2)

Υλικό εργαστηρίου 3:

https://github.com/csymvoul/Information-Systems-Lab/blob/master/lab3/flask_pymongo/app.py

POST:

- Το δηλώνουμε στο route:

```
@app.route('/post_endpoint', methods=['POST'])
```

- Και πρέπει να περιμένουμε κάποια πληροφορία στο body την οποία ζητάμε με το request:

```
data = json.loads(request.data)
```

Flask - PyMongo (2/2)

GET:

- Είναι η προεπιλογή στο Flask οπότε μπορούμε και να το παραλείψουμε:

```
@app.route('/get_endpoint')
@app.route('/get_endpoint', methods=['GET'])
```

- Μπορούμε να ζητήσουμε και κάποια πληροφορία στο URI το οποίο θα έχουμε σαν input στη συνάρτηση:

```
@app.route('/get_endpoint/<string:id>')
def get_endpoint(email):
```

- Μπορεί ένα endpoint να έχει πάνω από μία μεθόδους:

```
@app.route('/get_post_endpoint', methods=['GET', 'POST'])
```

- Και ανάλογα με την επιλογή του χρήστη να πράττουμε διαφορετικά:

```
def get_post():
        if request.method == 'GET':
                pass
        elif request.method == 'POST':
                pass
```

Προαιρετική Εργασία

Χρησιμοποιήστε ένα MongoDB container το οποία θα χρησιμοποιεί την port 27017 του host και να υλοποιηθεί το ένα web service:

1. Να βρίσκει τα άτομα που έχουν κάποια καταγεγραμμένη κατοικία
   `endpoint: /getAllStudentsAddress`
2. Να βρίσκει τη διεύθυνση της κατοικίας τους βάσει email
   `endpoint: /getStudentsAddress/<student_email>`
3. Να βρίσκει όλα τα άτομα που έχουν κατοικία και έχουν γεννηθεί τη δεκαετία του 1980.
   `endpoint: /getEightiesAddress`
4. Να επιστρέφεται ο αριθμός των ατόμων που έχουν δηλωμένη κάποια κατοικία.
   `endpoint: /countAddress`
5. Να υλοποιηθεί ξανά η συνάρτηση `insert_student()` ώστε να μπορεί να γίνει εισαγωγή στη βάση δεδομένων νέος φοιτητής μαζί με δεδομένα για τη κατοικία του.
6. Να επιστρέφεται ο αριθμός των ατόμων που έχουν γεννηθεί μία συγκεκριμένη χρονιά:
   `endpoint: /countYearOfBirth/<yearOfBirth>`

Η προαιρετική εργασία να ανέβει στο GitHub σε private repository στο οποίο θα δώσετε access στους υπεύθυνους του εργαστηρίου.
Τα GitHub accounts είναι τα : csymvoul, jdtototw