Facultatea Calculatoare, Informatica si Microelectronica

Universitatea Tehnica a Moldovei

Programarea retelelor

Lucrarea de laborator#2-3

HTTP Client with concurrency superpowers

Autor:
Ana Sutreac

lector asistent:
Alexandru Gavrisco
lector superior:
Dumitru Ciorba

Laboratory work #5

1 Scopul lucrarii de laborator

Study OSI model, HTTP and implement a client app which can do multiple HTTP requests concurrently.

2 Obiective

- Read about OSI model (definition, basic info)
- Read about (de)serialization
- Concurrency (definition, primitives etc)

3 Laboratory work implementation

3.1 Sarcina de baza (5 - 7)

Metrics Aggregator

Your new client is a super secret organization which needs to collect a bunch of different metrics from an object and aggregate them. An object is a real location that has some devices which collects data from sensors. Each device has an id, type or sensor_type and a value which describes device's state. So, metrics is just a fancy word for data which describes state of your object from a perspective (e.g. temperature, air pressure etc).

Device/Sensor Types:

- Temperature sensor - 0
- Humidity sensor - 1
- Motion sensor - 2
- Alien Presence detector - 3
- Dark Matter detector - 4

However, there are some problems about those devices:

- Each device has its own URL and can be accessed only using a secret key
- Devices can return values in different formats (JSON, XML, CSV)
- Depending on format, a device can return multiple values (e.g. a box which has multiple sensors)
- Sometimes a device can respond you in up to 29 sec.
- And the most important constraint - your secret key is valid only 30 sec. (because the organization is super secret and don't want to leak access keys).

Examples of some metrics:

A single device in JSON format:

A single device in XML format:

A device with multiple sensors in CSV format:

Here's what functionality your app must offer:

Request your secret key at https://desolate-ravine-43301.herokuapp.com/, in response you'll receive a list of URLs (for each device)

Using your secret key, request data from all devices concurrently

If you get an error related to your access key, go back to step 1 and retry

Parse data from all devices

Aggregate all responses ordering by sensor type (example of output below)

3.2 Anexa 1

lab_2_3.py

```
import sys
import requests
import threading
import queue
import json
import xmltodict
import csv
from io import StringIO

fileAllData = open("AllData", "ab+")

def process(path):
        result = requests.get(url + path, headers=headers)
        data_queue.put(result)
        for k in result:
                fileAllData.write(k)


def parseJsonData(result_text):
        result_dict = json.loads(result_text)
        return [result_dict]



def parseXMLData(result_text):
        result_dict = dict(xmltodict.parse(result_text)['device'])
        result_dict["device_id"] = result_dict.pop("@id")
        result_dict["sensor_type"] = int(result_dict.pop("type"))
        return [result_dict]


def parseCSVData(data):
        str_data = StringIO(data)
        reader = csv.reader(str_data)
        reader = list(map(list, reader))
        reader.pop(0)
        for line in reader:
```

```
                result_dict = dict(zip(metric_keys, line))
                result_dict["sensor_type"] = int(result_dict.pop("sensor_type
                result_dict["sensor_type"] = int(result_dict.pop("sensor_type
                result.append(result_dict)
                return result


def printData(sensor_data):
        for k in sensor_types:
                print("\n" + sensor_types[k])
                if k in sensor_data.keys():
                        for dct in sensor_data[k]:
                                print("Device " + dct['device_id'] + ' - ' +
                else:
                        print("Secret data not detected")



def processData(raw_data):
        for result_dict in raw_data:
                if result_dict["sensor_type"] not in sensor_data:
                        sensor_data[result_dict["sensor_type"]] = []
                        sensor_data[result_dict["sensor_type"]].append(result
                else:
                        sensor_data[result_dict["sensor_type"]].append(result


url = 'https://desolate-ravine-43301.herokuapp.com'

try:
        result = requests.post(url)
except:
        print("Check your conection")
        sys.exit(1)

key = result.headers['session']
headers = {'session': key}
paths = json.loads(result.text)
paths = [path['path'] for path in paths]
```

```
metric_keys = ["device_id","sensor_type","value"]
sensor_types = {0: 'Temperature sensor', 1: 'Humidity Sensor', 2: 'Motion Ser
sensor_data = {}
threads = []
data_queue = queue.Queue()

for path in paths:
        t = threading.Thread(target=process, args=(path,))
        threads.append(t)
        t.start()

for path in paths:
        data = data_queue.get()
        content_type = (data.headers['content-type']).lower()
        if content_type == 'application/json':
                result = parseJsonData(data.text)
        elif content_type == 'application/xml':
                result = parseXMLData(data.text)
        elif content_type == 'text/csv':
                result = parseCSVData(data.text)
                processData(result)

printData(sensor_data)
if __name__ == '__main__':
        pass
```

Concluzie

Aici trebuie sa fie concluzia ta.

References

1 Aldebran Robotics, *official page*, www.aldebaran.com/en

2 Timo Ojala, *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*, 2002

3 Biometric, www.biometricupdate.com/201501/history-of-biometrics