


| | | |
|---|---|---|
|  UNIVERSITY OF LEEDS | School of Computing University of Leeds Coursework 2 - Answers | Module Code COMP3211 |
|---|---|---|

Full Name: Thomas Moreno Cooper
Coursework Title: Web services

Username: sc17tm
Deadline Date: 22/11/2019

Question 1

| | |
|------------------------------|---|
| Web service No. 1 | |
| Name | Google Maps Static Api |
| SOAP-based or RESTful | RESTful |
| Name of publisher | Google |
| Brief description | The api provides a way to obtain a custom map as an image, showing desired locations or areas. |
| URL | https://maps.googleapis.com/maps/api/staticmap |

| | |
|------------------------------|---|
| Web service No. 2 | |
| Name | MeMemory Api |
| SOAP-based or RESTful | RESTful |
| Name of publisher | Translated LABS |
| Brief description | The api lets one translate a desired string of text into a different language. |
| URL | https://api.mymemory.translated.net/get |

| | |
|------------------------------|--|
| Web service No. 3 | |
| Name | Locapi |
| SOAP-based or RESTful | RESTful |
| Brief description | The api provides an interface for a database of capital cities and useful information for each: their coordinates, their country and the main language spoken. It also calculates the distance between two locations using the Haversine formula |

Composition of Originality

| Web Service | Input | Output | Output Parsing |
|-------------|--|-----------------------------------|---|
| 1 | Size, map type, markers (ie the positions to be shown on the map, separate each location with " "), api key. | A PNG image as an array of bytes. | , just save the array of bytes into a file, calling it "map.png". |

| | | | |
|---|---|---|---|
| 2 | Text to be translated, language code pair (first the code of the text's language, separating the requested language with " ") | A lot of data in JSON format containing the translated version of the text as a string and other information relating to the translation such as metrics (translation time, translation approximation etc). | The translated text is stored in the element with the "translatedText" key. |
| 3 | If querying the database, input a country (all country names are unique). If calculating distance, input name of start and end locations. | Data stored in JSON format. Always a single string value except for one command which returns an array of strings containing the name of all countries in the database. | Input the key of the desired data. Depending on the method called, these can be "capital", "code", "distance" or "locations". |

Implementation details

I - Google Maps Static Api

The google maps api is invoked in a function using the python requests library.

```

1. def getMap(location1, location2):
2.     response = requests.get(
3.         'https://maps.googleapis.com/maps/api/staticmap?size=512x512&maptpe=road
         map&markers='
4.         +location1+'|'+location2+'&key=MyApiKey')

```

The desired locations, a pair of strings, are pasted in their corresponding places in the url as parameters to obtain the desired map.

By calling response.content, we access the array of bytes contained in the response and store it as an image file using the python pillow library.

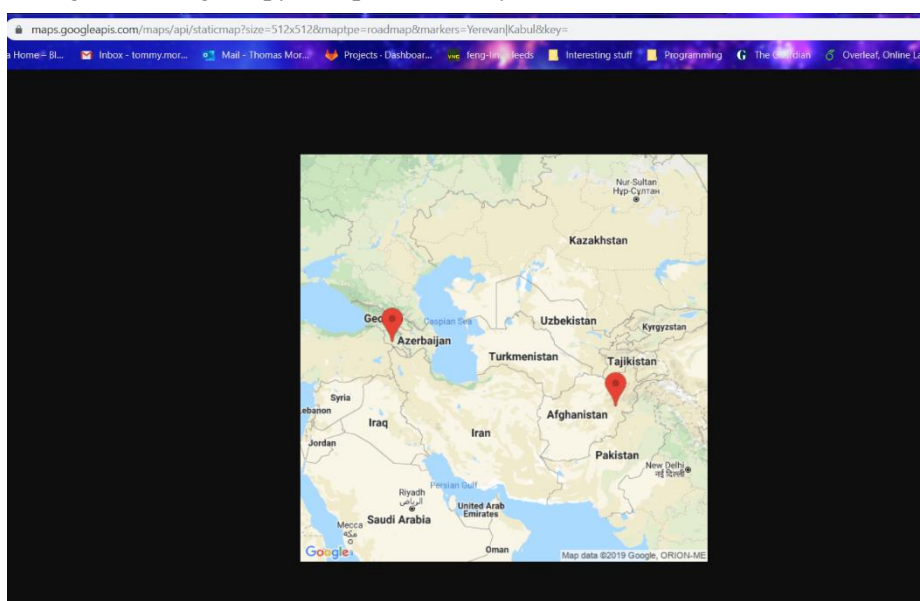


Figure 1: Maps api for Yerevan and Kabul, using the following link
https://maps.googleapis.com/maps/api/staticmap?size=512x512&maptpe=roadmap&markers=Yerevan/Kabul&key=*

The database follows the following model.

```
1. class Location(db.Model):
2.     __tablename__ = 'Locations'
3.
4.     id = db.Column(db.Integer, primary_key=True)
5.     country = db.Column(db.String(64))
6.     capital = db.Column(db.String(64))
7.     langCode = db.Column(db.String(10))
8.     latitude = db.Column(db.String(7))
9.     longitude = db.Column(db.String(7))
```

The resulting database is where the api will get all its data from.

The api is created from a flask app and its resources are defined as classes with a single method.

```
1. class LanguageCode(Resource):
2.     def get(self):
3.         country = request.args.get('country')
4.         query = m.Location.query.filter_by(country=country)
5.         return jsonify({"code":query[0].langCode})
6.
7. api.add_resource(LanguageCode, '/getCode')
```

Each method provides the desired functionality and returns the required data in json format. The resource is then added to the api with a url to invoke the command.

Like the other services, invoking Locapi is done using the python requests library in a separate python file containing a different function for each of the api's commands.



Figure 3: JSON response of the getDistance function for Vienna and Brussels

Integration / User Interface

All three web services have a dedicated python file located in the same directory called apiRequests that contains functions invoking the commands of the service. These functions also parse the data so that calling a service via these functions returns a string that can be stored in a dictionary which is useful for the web application.

Like for the api, the web-based application is hosted by flask. It has two pages, the first one lets the user select countries from two drop down fields. The countries available in those fields are fetched from the Locapi webservice. Once the user submits those choices, the results are displayed in a new page. This page is rendered from a template with a context. This context is a dictionary of strings which is created before the page is rendered by calling the webservices functions returning strings of data relevant to the choices made by the user.

Successful Execution

With a bit of css to embellish the app, we obtain the following pages.

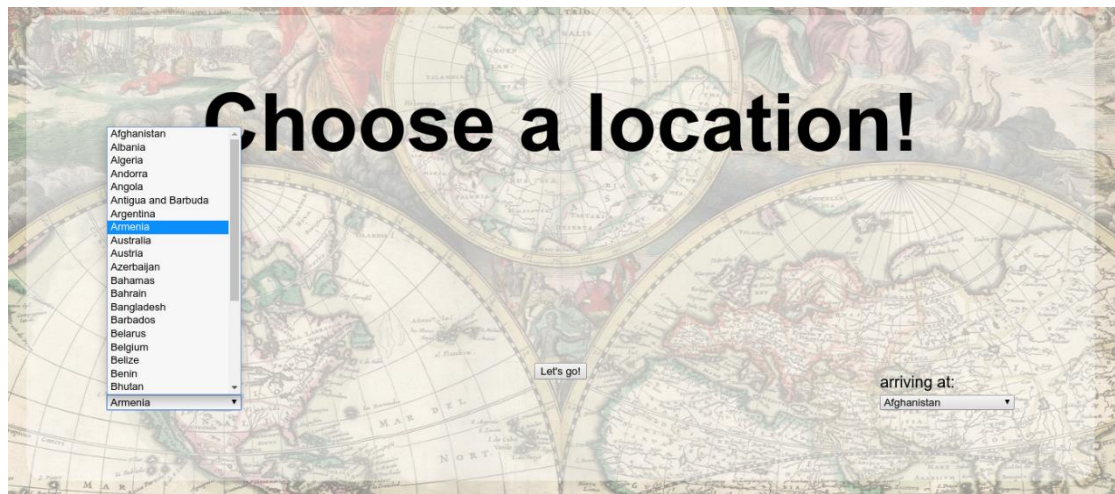


Figure 5: Select country page



Figure 4: Results page, displaying the map, the distance and some text translated in the destination country's language

Other Comments

The design only uses get requests, perhaps a way to make this application more interesting would be to let people add their own locations. This could be done by adding more functionality to Locapi.

Question 2

Developing the web application was quite like developing an RPC client. The web application needed proxies in the form of functions sending http requests to the various web services used. There was a need to implement the marshalling of requests, creating valid url strings to be sent to the servers. For Locapi, the requests had to be of a certain format with certain arguments, much like the server stubs in RPC and responses had to be marshalled in a universal format, in this case in JSON. Lastly, by having everything received in the JSON format, we mimic the unmarshalling of the responses by the client to make sure the data is the same and platform independent.

Question 3

| Web service | Time ms (n=5) | average | Standard deviation |
|------------------|---|---------|-----------------------|
| 1.Google maps | 0.228, 0.426, 0.209, 0.226, 0.265 | 0.2708 | 0.07 |
| 2.MyMemory | 0.584, 0.571, 0.563, 0.522, 0.609 | 0.5698 | 0.02 |
| 3.Locapi | 0.050, 0.016, 0.012, 0.015, 0.015 | 0.0216 | 0.014 |

By comparing the results of the analysis, there is a clear difference between the time taken to complete the first two services and the third. The obvious explanation is that latency is longer for the first two whereas in this test scenario, the third service is hosted on the same computer as the client running the tests which renders latency totally insignificant. Noticeably, the average time is a little higher for the MyMemory api. This could be due to the nature of the task executed which, according to the MyMemory api documentation, is querying a large database of pre existing translations to check whether the text inputted has already been translated and if the translations found are adequate. Additionally, google has undoubtedly access to higher performance servers to generate maps even faster.

Question 4

As mentioned in Question 3, latency is an issue and affects the performance of the application. If the service were to be used repeatedly, the services could implement some sort of caching of the most used and the most recent locations. Limiting the use of http requests by merging requests together or if the web services could be amalgamated together would improve performance. This would also reduce the size of the messages being although this would come at the cost of interoperability by limiting the number of potential service users. Python is a very high-level language and Flask is

a high-level framework. Most of the data handling, whether it be getting requests, parsing JSON, manipulating strings etc is done through objects and methods which add to the cost of the application. Switching to a more low-level language/framework, although harder to program, may improve performance.