4.7 Accessing APIs with Python

RESTful APIs

API (Application Programming Interface) is an interface that facilitates communication between two software. In web services, these two software are often a web service/server and a web client. APIs facilitate sending and retrieving data as well as modifying existing data stores at an API server.

RESTful APIs

There are different API standards used in various applications. One of the widely used type of APIs is called RESTful APIs or REST APIs. REST APIs work on top of the Hyper Text Transfer Protocol, in short the HTTP protocol to accept request. REST APIs can be accessed by sending HTTP requests.

Most common HTTP requests used in REST APIs:

GET : Retrieve existing data

POST : Add new data

PUT : Update existing data

PATCH : Partially update existing data

DELETE : Delete data

Response Codes

Once a REST API receives an HTTP request, it returns a response as an HTTP status code.Status codes can be categorized based on the starting digit as follows:

1xx: Information response

2xx: Successful operation

3xx: Redirection

4xx: Client error

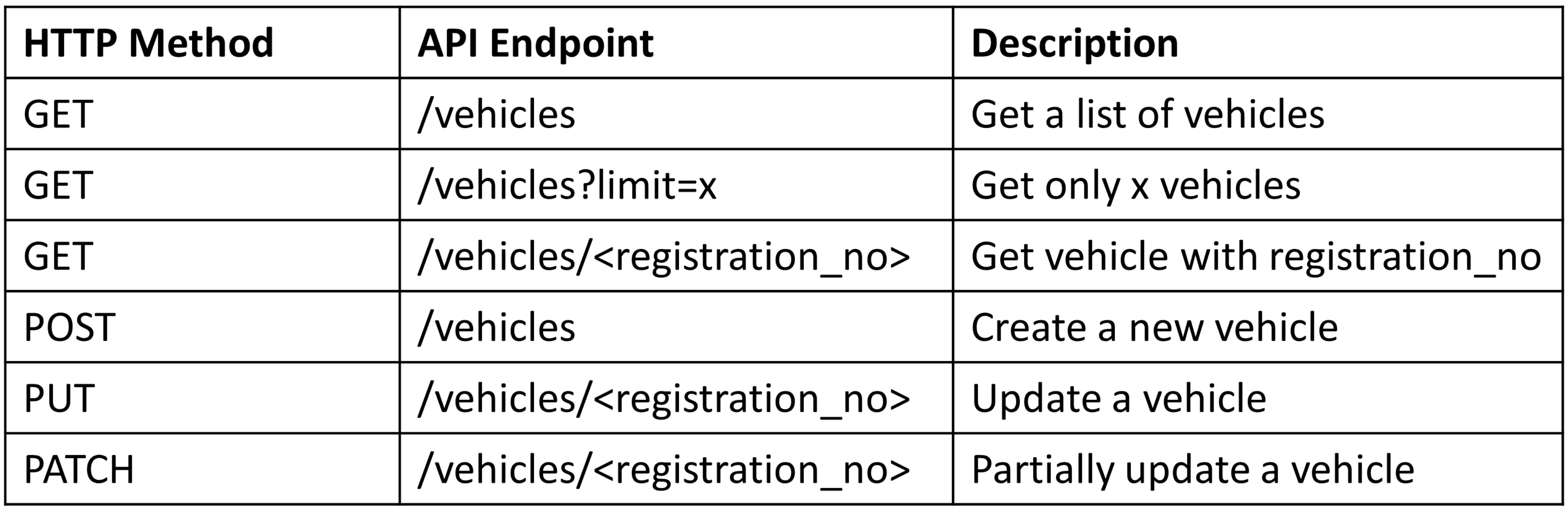
5xx: Server error

API Endpoints

API Endpoints are Public URLs exposed by the API server or the web service. Software clients can access APIs by sending requests to these exposed URLs.

For example, let's say that there is an API that provides vehicle registration information and it is hosted at the base URL https://vehicleapi.com.

The below table shows some example endpoints for our imaginary vehicle API:



HTTP Method	API Endpoint	Description
GET	/vehicles	Get a list of vehicles
GET	/vehicles?limit=x	Get only x vehicles
GET	/vehicles/<registration_no>	Get vehicle with registration_no
POST	/vehicles	Create a new vehicle
PUT	/vehicles/<registration_no>	Update a vehicle
PATCH	/vehicles/<registration_no>	Partially update a vehicle

Accessing APIs with Python Requests Library

REST APIs can be accessed on Python using **requests** library. requests is an external library, thus needs to be installed on the

Python environment before using.

Requests library can be installed using pip as follows:

```
pip install requests
```

This command needs to be given in the command prompt. pip is a utility that helps us install and manage python libraries.

GET Requests

Here is an example GET request to retrieve vehicle information from the imaginary API we created, which is having the base URL https://vehicleapi.com. According the API endpoint table shown above, the API endpoint for retrieving vehicle information is BASEURL/vehicles.

Thus, we create a GET request to BASEURL/vehicles. You can see that there is a variable called response which is assigned the return value of the requests.get() function call. The vehicle records received by the GET requests are captured in this variable. After the GET request is completed, we can process the received data.

```
import requests
BASE_URL = ‘https://vehicleapi.com’
response = requests.get(f"{BASE_URL}/vehicles")
print(response.status_code)
print(response.json())
```

Here is another example of GET request. In this example, we assume an API endpoint which takes in the registration number of a vehicle and returns associated data.

```
import requests
BASE_URL = ‘https://vehicleapi.com’
response = requests.get(f"{BASE_URL}/vehicles/CBA4476”)
print(response.status_code)
print(response)
```

POST Request

POST requests are used to create new data entries in the API server. In this example we are sending new data entry about a vehicle with registration number CBB1456 to be stored at the API server.

```
import requests
BASE_URL = ‘https://vehicleapi.com’
vehicle_entity = {  "motorvehicles": {      "vehicle": {“type": “car", "registration_no": "CBB1456",      "mal
response = requests.post(f"{BASE_URL}/vehicles", json=vehicle_entity)
print(response.status_code)
print(response.json())
```

we assume that our API server works with JSON data, thus the sent information has to be formatted according to JSON format.

You may notice that in the requests.post function, we have mentioned json as an argument. If we indicate as an argument like this, Python automatically sets the content type of the HTTP request to “application/json”, meaning that we are sending JSON content. This is required for HTTP requests.

PUT Request

POST request can be used to create a new data entry in the API server. PUT request is used to replace an existing record of data with a new record.

```
import requests
BASE_URL = ‘https://vehicleapi.com’
updated_entity = {  "motorvehicles": {      "vehicle": {“type": “car", "registration_no": "CBB1456",      "mal
response = requests.put(f"{BASE_URL}/vehicles/CBB1456", json=updated_entity)
print(response.status_code)
print(response.json())
```

PATCH Request

PATCH request is also used to update an existing record, but partially. If you compare the PATCH request with the PUT request, PUT request replaces the entire record with new data whereas PATCH request only modifies a subset of fields in the existing data record. In the example shown, PATCH request will update only the owner information of the vehicle with registration number CBB1456.

```
import requests
import requests
BASE_URL = ‘https://vehicleapi.com’
updated_entity = {"owner": {"first_name": “Marlon",      "last_name": “Samuels", "nic": “870124770V" }  }
response = requests.patch(f"{BASE_URL}/vehicles/CBB1456", json=updated_entity)
print(response.status_code)
print(response.json())
```

DELETE Request

The DELETE record allows us to delete a record in the API server.

```
import requests
BASE_URL = ‘https://vehicleapi.com’
response = requests.delete(f"{BASE_URL}/vehicles/CBB1456")
print(response.status_code)
print(response.json())
```

Lecture Notes 2 / 5

◄ 4.6 Quiz

Jump to...