HTTP Requests

Contents

Overview

HTTP requests are used for many communications between services and for accessing information over the internet.
The `requests` library for Python can be used for making HTTP requests in Python.

Installation

The `requests` library can be installed using Pip:

```
pip3 install requests
```

And then imported accordingly when you would like to use it in your Python scripts:

```
import requests
```

Usage

GET Request

The `.get()` function can be used for making HTTP GET requests with the `requests` library:

```
import requests

response = requests.get('https://example.api.com')
```

POST Request

The `.post()` function can be used for making HTTP POST requests:

```
import requests

response = requests.post('https://example.api.com')
```

Sending Data

Data can be set to the named parameter `data` when making post requests to also send information:

```
import requests

response = requests.post('https://example.api.com', data='my data')
```

Sending JSON

In may cases when you want to POST data, you will send it as a JSON Object, this can be set as a Dictionary in the `json` named parameter when calling `.post()`:

```
import requests

response = requests.post('https://example.api.com', json={"message": "My JSON"})
```

Responses

When making HTTP requests you will often need to read and process the responses.
The examples in this section are for HTTP GET requests however you will be able to access response information in the same way as the other request types.

Response Text

Once you have made a request you will have a response which you can get the contents of from the `.text` property:

```
import requests

# response = requests.get('URL')
response = requests.get('https://example.api.com')
```

Response JSON

API servers will commonly return the data in JSON format, `requests` has a `json()` function built in for this.
Calling this function on the response will return a Python dictionary:

```
import requests

# response = requests.get('URL')
response = requests.get('https://example.api.com')
json_response = response.json()
print(json_response['SOME_KEY'])
```

Tutorial

This tutorial will guide through:

- Creating basic Flask application which has several routes to test with the `requests` library
- Making requests to the various routes on the example Flask application

Prerequisites

- Fresh Debian/Ubuntu VM
  This will help avoid conflicts with any other work that you have been doing.

- **Python 3 & Virtual Environment**
  Neccessary for configuring the Python applications we are building, use this command on a terminal to make sure that they are installed:

```
sudo apt update
sudo apt install -y python3 python3-venv
```

- **Git**
  Git will be needed to download the example API application:

```
sudo apt update
sudo apt install -y git
```

- **Example Flask API**
  To test the `requests` module an example API Flask Application has been provided.
  It is a very simple application which returns text and JSON responses, feel free to look at the code for this here.

  The file structure of the repository looks like:

```
.
├── README.md
├── app.py
├── flask-app.service
├── hack
│   ├── deploy.sh
│   └── uninstall.sh
└── requirements.txt
```

  Install this API on the VM using these commands:

```
mkdir -p ~/projects/gitlab.com/qacdevops && cd $_
git clone https://gitlab.com/qacdevops/python-flask-example-api
cd python-flask-example-api
sudo ./hack/deploy.sh
```

  The API setup is running on port `5000` and has the following routes configured:

Route	Description	Content Type
`/get/text`	Return a simple text message: *"Hello from Flask"*	`text/plain`
`/post/text`	Return the data from request back as a part of a string: *"Data you sent: `[DATA_YOU_SENT]`"*	`text/plain`
`/get/json`	Return a JSON object containing a `data` property with a message: `{"data": "Hello from Flask"}`	`application/json`
`/post/json`	Return the data from request back as a `data` property inside of a JSON: `{"data": "[DATA_THAT_WAS_SENT]"}`	`application/json`

Setup a Project for Making Requests

We'll need a project setup now for using the `requests` library:

```
mkdir -p ~/projects/http-requests-tutorial/ && cd $_
python3 -m venv venv
source ./venv/bin/activate
pip3 install requests
```

You now a virtual environment with `requests` installed, create an `__init__.py` file in this folder and enter the following:

```
import requests
api = 'http://localhost:5000'
```

Here we have imported the `requests` module for making requests and made an `api` variable so that we aren't going to be repeating ourselves as much on all the requests.

Basic HTTP GET Request

There's a route: `/get/text` we can create requests for to get a text response, add this to the `__init__.py`:

```
print('HTTP GET Request (text):')
response = requests.get(api + '/get/text')
print('Response: ', response.text)
```

When you run `python __init__.py` there should be an output like this:

```
HTTP GET Request (text):
Response: Hello from Flask
```

Basic HTTP POST Request

Some text can be sent to `/post/text`, the response will change depending on the data that you send to it:

```
print('HTTP POST Request (text):')
response = requests.post(api + '/post/text', 'My Data')
print('Response: ', response.text)
```

You should see an output like this:

```
HTTP POST Request (text):
Response: Data you sent: My Data
```

JSON HTTP GET Request

A simple GET request to `/get/json` can be made to get a JSON response:

```
print('HTTP GET Request (json):')
response = requests.get(api + '/get/json')
print('Whole Response: ' + str(response.json()))
print('"data" Property of the Response: ' +  str(response.json()["data"]))
```

You should see an output like this:

```
HTTP GET Request (json):
Whole Response: {'data': 'Hello from Flask'}
"data" Property of the Response: Hello from Flask
```

JSON HTTP POST Request

A POST request to `/post/json` will return a JSON object containing the JSON object that you have sent:

```
print('HTTP POST Request (json):')
response = requests.post(api + '/post/json', json={"message": "mydata"})
print('Whole Response: ' + str(response.text))
print('"data" Property of the Response: ' +  str(response.json()["data"]))
```

You should see an output like this:

```
HTTP POST Request (json):
Whole Response: {"data":{"message":"mydata"}}
"data" Property of the Response: {'message': 'mydata'}
```

Clean Up

To uninstall the Python API server, run the `uninstall.sh` script:

```
sudo hack/uninstall.sh
```

Exercises

There are no exercises for this module.