REST APIs

Definitions

- Representational State Transfer
- Used in most of the modern APIs on the web
- Used to support modern single page web applications
- API = Application Programming Interface
- ReST API : API to store/manage/retrieve data
- uses HTTP (protocol for web communications)

Resources and actions

- An API allows users to access "resources" (= data)
- A resource is represented by a URI (Uniform Resource Identifier)
- Resources are nouns. For example: `students`.
- Example: `https://my_api.com/students/2` is the URI that allows to manipulate data about the resource "student #2"
- Actions can be performed on resources.
- The HTTP protocol defines several actions when accessing resources.
- The 4 most important ones are:

Action	HTTP verb
Retrieve resource	GET
Create new resource	PUT
Update existing resource	POST
Delete resource	DELETE

HTTP response codes

The server returns a status code with the response. The status code contains information about the request and whether it was successfully processed.
The status code is a 3-digit number.

Code	Type	Comment
1xx	informational	request received, continuing process
2xx	successful	request was successfully received, understood, and accepted
3xx	redirection	further action needs to be taken in order to complete the request
4xx	client error	the request contains bad syntax or cannot be fulfilled
5xx	server error	the server failed to fulfill an apparently valid request

HTTP status codes you need to know

Code	Comment
`200 OK`	everything is fine
`204 OK`	everything is fine (empty response)
`400 Bad Request`	the client did something wrong
`403 Forbidden`	Access to this resource is forbidden
`404 Resource Not Found`	This resource does not exist
`405 Method Not Allowed`	The resource can not be manipulated with that method
`500 Internal Server Error`	You should be worried - bug in the application?

Making HTTP requests with the `requests` library

- Install it in the virtual environment: `pip install requests`
- You can make GET, POST, PUT, DELETE requests with the relevant methods.
- The following are examples using a demo API that gives you info about the request just made.

Example: GET

```
>>> response = requests.get("https://httpbin.org/get")
>>> response.text
'{\n  "args": {}, \n  "headers": {\n    "Accept": "*/*", \n    "Accept-Encoding": "gzip, deflate", \n
"Host": "httpbin.org", \n    "User-Agent": "python-requests/2.24.0", \n
"X-Amzn-Trace-Id": "Root=1-5f948529-1af85529337c3efc7189b2f3"\n  }, \n
"origin": "172.103.147.11", \n  "url": "https://httpbin.org/get"\n}\n'
>>> response.json()
{'args': {},
 'headers': {'Accept': '*/*', 'Accept-Encoding': 'gzip, deflate',
 'Host': 'httpbin.org', 'User-Agent': 'python-requests/2.24.0',
 'X-Amzn-Trace-Id': 'Root=1-5f948529-1af85529337c3efc7189b2f3'},
 'origin': '172.103.147.11', 'url': 'https://httpbin.org/get'
}
```

Example: PUT JSON data

```
>>> response = requests.put("https://httpbin.org/put", json={"hello": "world", "something": 12345})
>>> response.json()
{'args': {},
 'data': '{"hello": "world", "something": 12345}',
 'files': {}, 'form': {},
 'headers': {
   'Accept': '*/*', 'Accept-Encoding': 'gzip, deflate',
   'Content-Length': '38', 'Content-Type': 'application/json',
   'Host': 'httpbin.org', 'User-Agent': 'python-requests/2.24.0',
   'X-Amzn-Trace-Id': 'Root=1-5f948572-6f73cf4c41406d523a3eb244'
 },
 'json': {'hello': 'world', 'something': 12345},
 'origin': '172.103.147.11', 'url': 'https://httpbin.org/put'
}
```

Note the `data` attribute: this is what we sent to the API.

Example: status codes

The HTTPBIN API allows us to try out status codes as well:

```
>>> response = requests.get("https://httpbin.org/status/404")
>>> response.status_code
404
>>> response = requests.get("https://httpbin.org/status/503")
>>> response.status_code
503
```

Making HTTP requests with httpie

Install it in your virtual environment: `pip install httpie`.

You can make requests from the command line with the `http` command:

```
> http get https://httpbin.org/get
HTTP/1.1 200 OK
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Length: 298
Content-Type: application/json
Date: Sat, 24 Oct 2020 19:55:09 GMT
Server: gunicorn/19.9.0

{
    "args": {},
    "headers": {
        "Accept": "*/*",
        "Accept-Encoding": "gzip, deflate",
        "Host": "httpbin.org",
        "User-Agent": "HTTPie/2.2.0",
        "X-Amzn-Trace-Id": "Root=1-5f94869d-187ce78308c9cb23422797dc"
    },
    "origin": "172.103.147.11",
    "url": "https://httpbin.org/get"
}
```

PUT requests with httpie

Specify the `--json` option, use `put` for the HTTP method, and pass your values as `key=value` parameters:

```
> http --json put https://httpbin.org/put name=Tim score=1234
HTTP/1.1 200 OK
Content-Type: application/json
Server: gunicorn/19.9.0

{
    [...]
    "data": "{\"name\": \"Tim\", \"score\": \"1234\"}",
    [...]
    "json": {
        "name": "Tim",
        "score": "1234"
    },
}
```

Note how `data` has the quotes `"` characters escaped.