Contents

SECTION - HTTP REQUESTS AND RESPONSES **3**

Overview of Section - HTTP Requests and Responses **3**

HTTP Request sent from Postman **4**
HTTP GET Request sent from cURL 4
HTTP Response to Postman GET /heartbeat request 4
Raw HTTP Requests and Responses 5

Basic HTTP Verbs **5**

References **5**

HTTP GET Verb **5**

HTTP GET Verb Example **6**

User-Agent Header **6**

Accept Header **6**

HTTP Status Codes **7**

Common HTTP Status Codes **7**

HTTP Status code references **7**

Common HTTP Status codes in response to a GET **8**

Basic Auth Header **8**

Create Basic Auth Header in Postman **8**

Create Basic Auth Header in Insomnia **8**

HTTP POST Verb **8**

HTTP POST Verb Send Example **10**

HTTP POST Verb Request Example **10**

HTTP POST Verb Response Example **10**

Common HTTP Status codes in response to a POST **11**

HTTP Message Body Format - JSON **11**

JSON Example Explained **11**

XML Example Explained **12**

HTTP Message Body Format - XML **12**
http://countwordsfree.com/xmlviewer 13

HTTP DELETE Verb **13**

HTTP DELETE Send Example **13**

HTTP DELETE Request Example **13**

HTTP DELETE Response Example **13**

Common HTTP Status codes in response to a DELETE **14**

Postman Collections **14**

Postman Collections **14**

Insomnia Workspaces **15**

Insomnia Workspaces **15**

Environment Variables **15**
just type `host` for auto complete in URL editing 16

Postman Create Environment **16**

Postman Add Environment Variables **16**

Insomnia Environment Management **16**

How to test with this information **18**

Exercises **18**

SECTION - MORE HTTP REQUESTS AND RESPONSES **18**

Overview of Section - More HTTP Requests and Responses **18**

URI - Universal Resource Identifier **19**

URI vs URL vs URN **19**

Scheme(s) **19**

Query Strings **19**

More About Query Strings **20**

HTTP Standards? **20**

HTTP PUT Verb **20**

HTTP PUT Send Example **20**

HTTP PUT Request Example **21**

HTTP PUT Response Example **21**

HTTP OPTIONS Verb **22**

HTTP OPTIONS Send Example **22**

HTTP OPTIONS Request Example **22**

HTTP OPTIONS Response Example **22**

Common HTTP Status codes in response to a OPTIONS **22**

HTTP OPTIONS Verb - Example swapi.co **23**

Exercises **23**

SECTION - HTTP REQUESTS AND RESPONSES

Overview of Section - HTTP Requests and Responses

- HTTP Verbs - GET, POST, DELETE
- Headers
- Responses
  - Status Codes - e.g. 200, 404, 500
- This is the foundation for most web, HTTP, REST testing and automating.

HTTP Request sent from Postman

```
GET http://localhost:4567/heartbeat HTTP/1.1
cache-control: no-cache
Postman-Token: ddf30bfe-b7e2-4d3c-b478-1103a5a174e5
User-Agent: PostmanRuntime/6.2.5
Accept: */*
Host: localhost:4567
accept-encoding: gzip, deflate
Connection: keep-alive
```

- important stuff: Verb (GET), Http version (1.1), User-Agent, Accept, Host, endpoint

---

HTTP GET Request sent from cURL

Command:

```
curl http://localhost:4567/heartbeat ^
-H "accept: application/xml" ^
--proxy 127.0.0.1:8888
```

Request:

```
GET http://localhost:4567/heartbeat HTTP/1.1
User-Agent: curl/7.39.0
Host: localhost:4567
Connection: Keep-Alive
accept: application/xml
```

HTTP Response to Postman GET /heartbeat request

```
HTTP/1.1 200 OK
Date: Thu, 17 Aug 2017 10:34:32 GMT
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(9.4.4.v20170414)
```

- cURL response was same but content-type was `application/xml`
- important stuff: Status Code (200 OK), Http version (1.1), Date, Content-Type

---

Raw HTTP Requests and Responses

- we need to be able to read them
- we will rarely have to create them by hand
- lookup headers you don't know
  - https://en.wikipedia.org/wiki/List_of_HTTP_header_fields
- some fields are for the server some are for the application some are documentation

---

Basic HTTP Verbs

- GET - retrieve data
- POST amend/create from partial information
- PUT - create or replace from full information
- DELETE - delete items
- OPTIONS - verbs available on this url

References

- W3c Standard
- IETF standard
- httpstatuses.com
- http://www.restapitutorial.com/lessons/httpmethods.html

HTTP GET Verb

- GET - retrieve data
- GET verbs can be issued by a browser
  - click on link
  - visit a site
- GET `http://compendiumdev.co.uk/apps/api/mock/reflect`
- Important Headers
  - User-Agent - tells server app type
  - Accept - what format response you prefer

Demo

HTTP GET Verb Example

```
curl http://localhost:4567/heartbeat ^
-H "accept: application/xml" ^
--proxy 127.0.0.1:8888
```

```
GET http://localhost:4567/heartbeat HTTP/1.1
User-Agent: curl/7.39.0
Host: localhost:4567
Connection: Keep-Alive
accept: application/xml
```

User-Agent Header

- Often not sent when accessing an API
- Marks request as coming from a browser

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/60.0.3112.90 Safari/537.36
```

Accept Header

- Defines the payload types that the receiver will accept
- If this was an API call it would likely return XML

```
Accept: text/html,application/xhtml+xml,application/xml;
q=0.9,image/webp,image/apng,*/*;q=0.8
```

Common values:

- `text/html`
- `application/json`
- `application/xml`

HTTP Status Codes

- 1xx Informational
  - 100 Continue
- 2xx Success
  - e.g. 200 OK
- 3xx Redirection
  - e.g. 301 Moved Permanently
- 4xx Client Error
  - e.g. 404 Not Found
- 5xx Server Error
  - e.g. 500 Internal Server Error

Common HTTP Status Codes

Status Code	**Status Code**
200 OK	405 Method Not Allowed
201 Created	409 Conflict
301 Moved Permanently	500 Internal Server Error
307 Temporary Redirect	501 Not Implemented
400 Bad Request	502 Bad Gateway
401 Unauthorized	503 Service Unavailable
403 Forbidden	504 Gateway Timeout
404 Not Found	

HTTP Status code references

- https://httpstatuses.com/
- https://moz.com/blog/response-codes-explained-with-pictures
- https://http.cat/
- https://httpstatusdogs.com/

Common HTTP Status codes in response to a GET

- **200** - OK, found the url, returned contents
- **301, 307, 308** - content has moved, new url in `location` header
- **404** - url not found
- **401** - you need to give me authorisation details see `WWW-Authenticate` header
- **403** - url probably exists but you are not allowed to access it

---

Basic Auth Header

- This application uses Basic Auth Authentication
- `Authorization` Header

e.g. `Authorization: Basic dXNlcjpwYXNzd29yZA==`

`dXNlcjpwYXNzd29yZA==` is base64 encoded "user:password"

see base64decode.org

- cURL you need to add the header
- Postman & Insomnia use the Authorization and Auth tabs

Create Basic Auth Header in Postman

Create Basic Auth Header in Insomnia

HTTP POST Verb

- POST amend/create from partial information
- send a 'body' format of content in the 'content-type' header

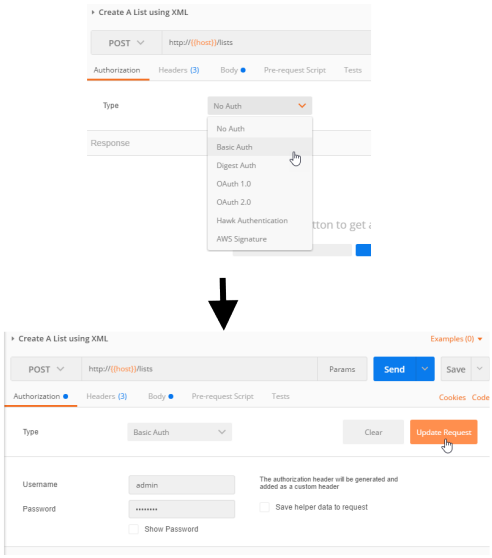

Figure 1: Create Basic Auth Header in Postman

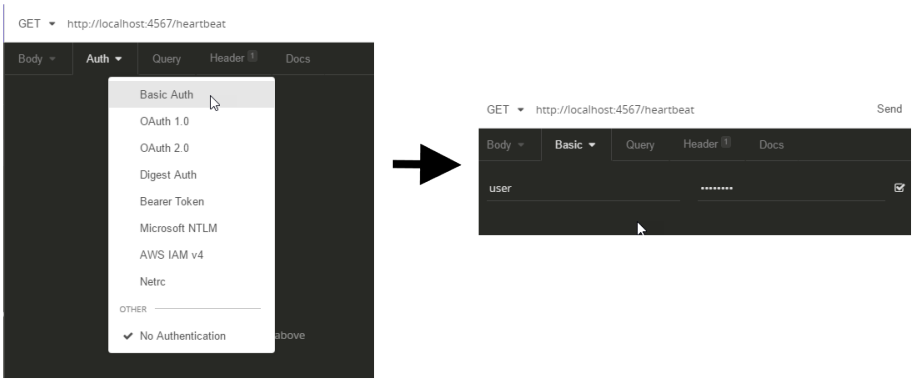

Figure 2: Create Basic Auth Header in Insomnia

- usually used to create or amend data
- browser will usually send a POST request when submitting a form

Demo

HTTP POST Verb Send Example

```
curl -X POST http://localhost:4567/lists ^
-H "accept: application/xml" ^
-H content-type:application/json ^
-H "Authorization: Basic dXNlcjpwYXNzd29yZA==" ^
-d "{title:'a list title'}" ^
--proxy 127.0.0.1:8888
```

HTTP POST Verb Request Example

```
POST http://localhost:4567/lists HTTP/1.1
User-Agent: curl/7.39.0
Host: localhost:4567
Connection: Keep-Alive
accept: application/json
content-type: application/json
Authorization: Basic dXNlcjpwYXNzd29yZA==
Content-Length: 22

{title:'a list title'}
```

HTTP POST Verb Response Example

```
HTTP/1.1 201 Created
Date: Thu, 17 Aug 2017 12:11:12 GMT
Content-Type: application/json
Location: /lists/f8134dd6-a573-4cf5-a6c6-9d556118ed0b
Server: Jetty(9.4.4.v20170414)
Content-Length: 171
```

```
{"lists":[{
"guid":"f8134dd6-a573-4cf5-a6c6-9d556118ed0b",
"title":"a list title",
"description":"",
"createdDate":"2017-08-17-13-11-12",
"amendedDate":"2017-08-17-13-11-12"}]}
```

Common HTTP Status codes in response to a POST

- **200** - OK, did whatever I was supposed to
- **201** - OK created new items
- **202** - OK, I'll do that later
- **204** - OK, I have no more information to give you
- **400** - what? that request made no sense
- **404** - I can't post to that url it is not found
- **401** - need authorisation see `WWW-Authenticate` header
- **403** - url probably exists but you are not allowed to access it
- **409** - can't do that, already exists
- **500** - your request made me crash

HTTP Message Body Format - JSON

- JSON - JavaScript Object Notation
- an actual Object in JavaScript
- common data transfer and marshalling format for other languages
- https://en.wikipedia.org/wiki/JSON
- http://json.org
- http://countwordsfree.com/jsonviewer
- schema exists for JSON http://json-schema.org/

JSON Example Explained

```
{
 "lists":
  [
```

```
   {
    "guid":"f8134dd6-a573-4cf5-a6c6-9d556118ed0b",
    "title":"a list title",
    "description":"",
    "createdDate":"2017-08-17-13-11-12",
    "amendedDate":"2017-08-17-13-11-12"
   }
  ]
}
```

- An object, which has an array called "lists".
- the lists array contains an object with fields: `guid`, `title`, `description`, `createdDate`, `amendedDate` - all String fields.

XML Example Explained

```
<?xml version="1.0" encoding="UTF-8" ?>
<lists>
  <list>
    <guid>f8134dd6-a573-4cf5-a6c6-9d556118ed0b</guid>
    <title>a list title</title>
    <description></description>
    <createdDate>2017-08-17-13-11-12</createdDate>
    <amendedDate>2017-08-17-13-11-12</amendedDate>
  </list>
</lists>
```

- elements, nested elements
- tags, values

HTTP Message Body Format - XML

- XML - eXtended Markup Language
- HTML is often XML
- another common marshalling format
- can be validated against XML schema
-

http://countwordsfree.com/xmlviewer

HTTP DELETE Verb

- DELETE - delete items

Demo

HTTP DELETE Send Example

```
curl -X DELETE http://localhost:4567/lists/{guid} ^
-H "Authorization: Basic YWRtaW46cGFzc3dvcmQ=" ^
--proxy 127.0.0.1:8888
```

HTTP DELETE Request Example

```
DELETE http://localhost:4567/lists/{guid} HTTP/1.1
User-Agent: curl/7.39.0
Host: localhost:4567
Accept: */*
Connection: Keep-Alive
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
```

HTTP DELETE Response Example

```
HTTP/1.1 204 No Content
Date: Thu, 17 Aug 2017 12:20:35 GMT
Content-Type: application/json
Server: Jetty(9.4.4.v20170414)
```

Common HTTP Status codes in response to a DELETE

- **200** - OK, did whatever I was supposed to
- **202** - OK, I'll do that later
- **204** - OK, I have no more information to give you
- **404** - I can't post to that url it is not found
- **401** - you need to give me authorisation details see `WWW-Authenticate` header
- **403** - url probably exists but you are not allowed to access it
- **500** - your request made me crash

Postman Collections

- "save as" requests to collections for re-use
- can share collections or export to file

Postman Collections

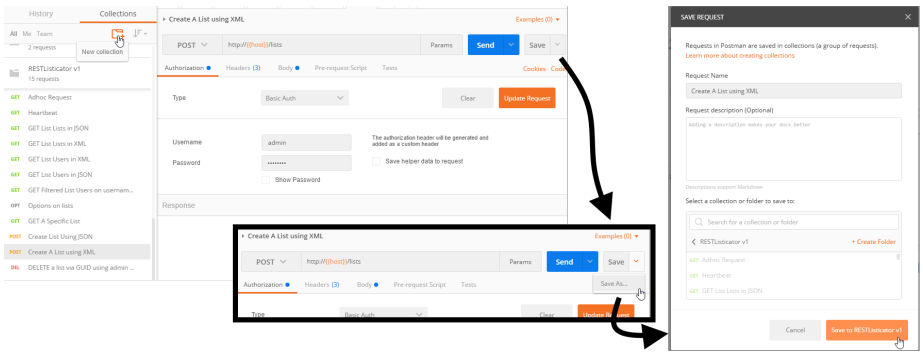

Figure 3: Postman Collections

Insomnia Workspaces

- create new Workspace
- create new Request in Workspace
- changes automatically saved to workspace
- can export workspace to files

Insomnia Workspaces

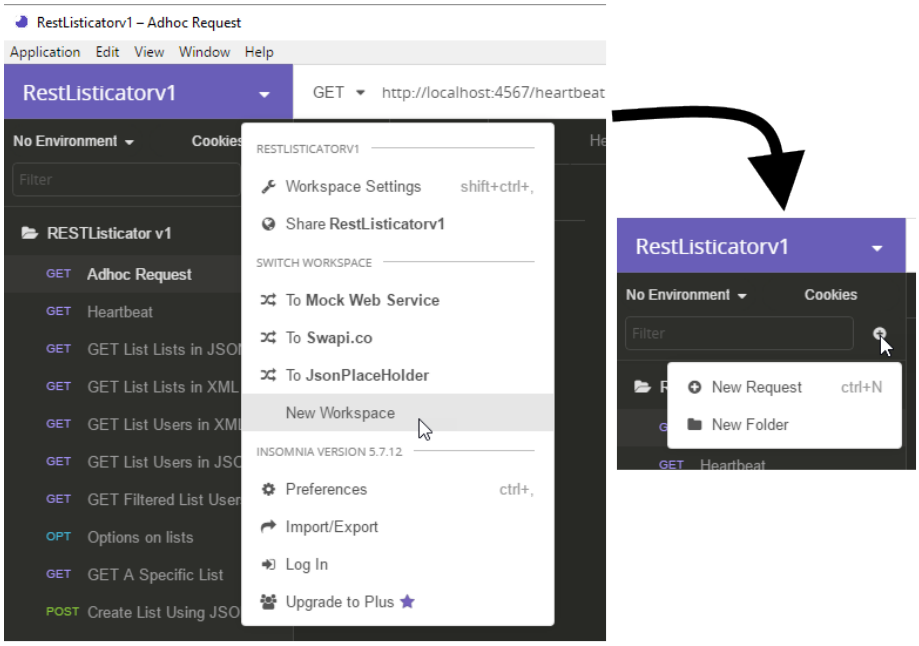

Figure 4: Insomnia Workspaces

Environment Variables

Postman:

- use environment variables e.g. `{{host}}` instead of `localhost:4567`
- `GET httt://{{host}}/lists`

Insomnia:

- use environment variables e.g. `{'host':'localhost:4567'}`
-

just type `host` for auto complete in URL editing

Postman Create Environment

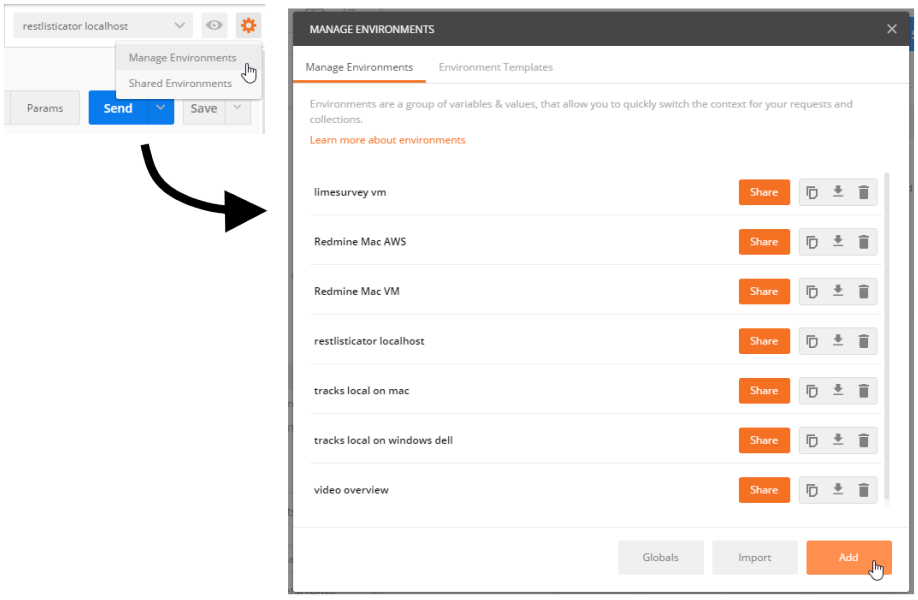

Figure 5: Postman Create Environment

Postman Add Environment Variables

Insomnia Environment Management

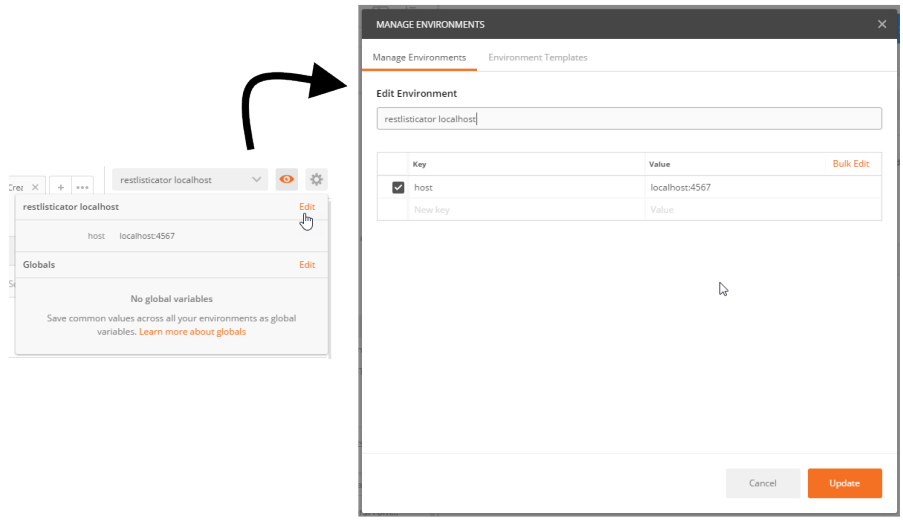

Figure 6: Postman Add Environment Variables

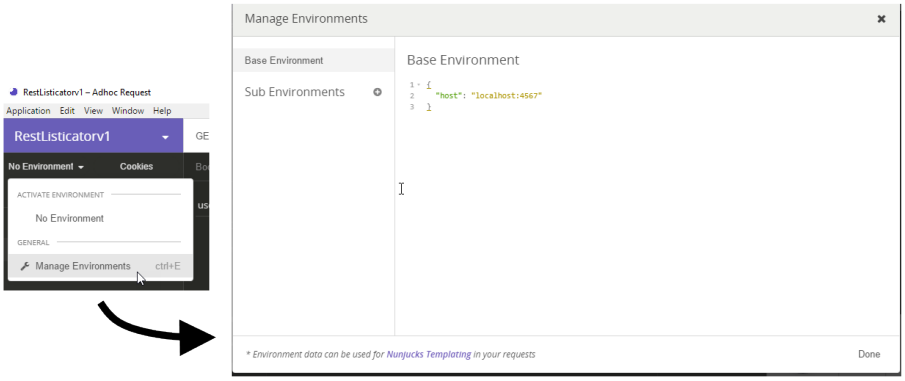

Figure 7: Insomnia Environment Management

How to test with this information

- Read the standards for the verbs and the status codes.
- Projects often argue about interpretations.
- Some of the standards are exact enough that it is possible to say "I observed X" it does not match the standard - include links and quotes to the standards.

Exercises

- install the REST listicator
  - http://compendiumdev.co.uk/downloads/apps/restlisticator
    - /v1/rest-list-system.jar download the `.jar` file
    - /v1/documentation.html read the documentation
- in the directory you downloaded it to type:
  - `java -jar rest-list-system.jar`
- try out the above GET, POST, DELETE, OPTIONS and POST using a GUI client e.g. Postman, or Insomnia
- see more exercises in exercise section
- explore the HTTP Client functionality and test the API based on its documentation
- see Exercises section for more exercises

SECTION - MORE HTTP REQUESTS AND RESPONSES

Overview of Section - More HTTP Requests and Responses

- HTTP Verbs - PUT, OPTIONS
- URI vs URL
- HTTP Standards
- cURL

URI - Universal Resource Identifier

`scheme:[//[user[:password]@]host[:port]][/path][?query][#fragment]`

- `http://compendiumdev.co.uk/apps/api/mock/reflect`
  - scheme = `http`
  - host = `compendiumdev.co.uk`
  - path = `apps/api/mock/reflect`

wikipedia.org/wiki/Uniform_Resource_Identifier

A URL is a URI

URI vs URL vs URN

- URI - Universal Resource Identifier
  - 'generic' representation - might not include the 'scheme'
  - `http://compendiumdev.co.uk/apps/api/mock/reflect`
  - `compendiumdev.co.uk/apps/api/mock/reflect`
  - `/apps/api/mock/reflect`
- URL - Universal Resource Locator
  - `http://compendiumdev.co.uk/apps/api/mock/reflect`
  - defines how to locate the identified resource
- URN - Universal Resource Name
  - not often used - uses scheme `urn`

Scheme(s)

- http
- https
- ftp
- mailto
- file

Query Strings

```
GET /lists/{guid}?without=title,description
GET http://localhost:4567/lists/f13?without=title,description
```

Query String:

`?without=title,description`

- starts with `?`
- params separated with `&`

More About Query Strings

`GET /lists/{guid}?without=title,description`

- usually `name=value` pairs separate by '&'
  - convention since anything after the `?` is the Query string
  - app then parses as required
- can be used with any verb
- `GET` request - all params are send as query strings

https://en.wikipedia.org/wiki/Query_string

HTTP Standards?

- rfc7231 (HTTP/1.1): Semantics and Content
- rfc7230 (HTTP/1.1): Message Syntax and Routing

HTTP PUT Verb

- PUT - create or replace from full information

Full information means it should be idempotent - send it again and get exactly the same request

Demo

HTTP PUT Send Example

```
curl -X PUT http://localhost:4567/lists ^
```

```
-H "Authorization: Basic dXNlcjpwYXNzd29yZA==" ^
--proxy 127.0.0.1:8888 ^
-d @createlistwithput.txt
```

where `createlistwithput.txt` file contains

```
{"title":"title added with put",
"description":"list description",
"guid": "guidcreatedwithput201708171440",
"createdDate": "2017-08-17-14-40-34",
"amendedDate": "2017-08-17-14-40-34"}
```

HTTP PUT Request Example

```
PUT http://localhost:4567/lists HTTP/1.1
User-Agent: curl/7.39.0
Host: localhost:4567
Accept: */*
Connection: Keep-Alive
Authorization: Basic dXNlcjpwYXNzd29yZA==
Content-Length: 180
Content-Type: application/json

{"title":"title added with put",
"description":"list description",
"guid": "guidcreatedwithput201708171440",
"createdDate": "2017-08-17-14-40-34",
"amendedDate": "2017-08-17-14-40-34"}
```

HTTP PUT Response Example

```
HTTP/1.1 201 Created
Date: Thu, 17 Aug 2017 13:41:46 GMT
Content-Type: application/json
Server: Jetty(9.4.4.v20170414)
Content-Length: 0
```

HTTP OPTIONS Verb

- OPTIONS - verbs available on this url
- returns an `Allow` header describing the allowed HTTP Verbs

HTTP OPTIONS Send Example

```
curl -X OPTIONS http://localhost:4567/lists ^
--proxy 127.0.0.1:8888
```

Demo

HTTP OPTIONS Request Example

```
OPTIONS http://localhost:4567/lists HTTP/1.1
User-Agent: curl/7.39.0
Host: localhost:4567
Accept: */*
Connection: Keep-Alive
```

HTTP OPTIONS Response Example

```
HTTP/1.1 200 OK
Date: Thu, 17 Aug 2017 12:24:39 GMT
Allow: GET, POST, PUT
Content-Type: text/html;charset=utf-8
Server: Jetty(9.4.4.v20170414)
Content-Length: 0
```

Common HTTP Status codes in response to a OPTIONS

- **200** - OK, did whatever I was supposed to

- **404** - I can't post to that url it is not found

HTTP OPTIONS Verb - Example swapi.co

e.g. swapi.co

OPTIONS - https://swapi.co/api/people/1/


```
{
    "name": "People Instance",
    "description": "",
    "renders": [
        "application/json",
        "text/html",
        "application/json"
    ],
    "parses": [
        "application/json",
        "application/x-www-form-urlencoded",
        "multipart/form-data"
    ]
}
```

Exercises

- try out the above OPTIONS and PUT using a GUI client e.g. Postman, or Insomnia
- try with different body content e.g. xml vs json
- try requesting `application/xml` instead of `application/json`
- explore the HTTP Client functionality and test the API based on its documentation
- see Exercises section for more exercises
