HTTP Can Do That?!  

A collection of bad ideas  

by Sumana Hariharanar
HTTP

Hypertext
Transfer
Protocol
Diagrams!

request >
UA =------------------------------------------ 0
< response

- Internet Engineering Task Force (IETF) RFC 7230
  Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing
HTTP: crash course
HTTP: crash course
HTTP: crash course
An HTTP Message
(Request or Response)

START-LINE

HTTP version (1.1)
Request method (GET, POST)
Response status code (200, 404, 500)
An HTTP Message (Request or Response)

START-LINE

HTTP version (1.1)
Request method (GET, POST)
Response status code (200, 404, 500)

HEADERS

Content-Type
Content-Length
......
An HTTP Message
(Request or Response)

START-LINE

HTTP version (1.1)
Request method (GET, POST)
Response status code (200, 404, 500)

HEADERS

Content-Type
Content-Length
......

BODY
Example Request

START-LINE
GET / HTTP/1.1

HEADERS
Host: www.sumana.biz
Accept: text/html
User-Agent: ScraperBot

BODY
Example Response

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 203
Date: Tue, 16 Jun 2015 16:21:56 GMT
Last-Modified: Tue, 16 Jun 2015 13:27:14 GMT

<html>
<head>
<title>Welcome to Sumanaville</title>
</head>
<body>
<center>
<h1>Rockin'</h1>
<p>This is a pretty rockin' site. I'm glad you decided to visit.</p>
<h2>Wheee!</h2>
</center>
</body>
</html>
Popular request methods ("verbs")

- **GET**
  - gimme

- **POST**
  - here you go
First bad idea: **POST** but not **GET**

class APIHTTPRequestHandler(BaseHTTPServer.BaseHTTPRequestHandler):
    def do_POST(self):
        more: https://github.com/brainwane/secureapi
POST but not GET:
use cases

letters to Santa Claus
POST but not GET:
use cases

employee suggestion box
POST but not GET:
use cases

extremely moderated blog comments
(a logistical note)
Bad Idea Scale
Giving client no way to **GET** – bad idea
Remember “CRUD”?

Create  Read

Update  Delete
Remember “CRUD”?

Create

**POST**

Read

**GET**

Update

**POST**

Delete

**POST**
Remember “CRUD”?

Create
POST

Read
GET

Update
POST

Delete
POST
Remember “CRUD”? INELEGANT!
Underappreciated methods

DELETE

delete a resource!
Implementing DELETE

`sumanah@perspective ~/test/http $ python serverwdelete.py`

serving at port 8000
Implementing DELETE

```
sumanah@perspective ~/test/http $ python serverwdelete.py
serving at port 8000

>>> import requests
>>> r = requests.get("http://localhost:8000")
```
Implementing **DELETE**

```python
>>> import requests
>>> r = requests.get("http://localhost:8000")
```

```
sumanah@perspective ~/test/http $ python serverwDELETE.py
serving at port 8000
127.0.0.1 - - [18/Jul/2015 19:53:49] "GET / HTTP/1.1" 200 -
```
Implementing **DELETE**

class APIHTTPRequestHandler(BaseHTTPServer.BaseHTTPRequestHandler):

    def do_DELETE(self):
        def delete_file(name):
            self.send_response(204)
            os.remove(name)
            self.end_headers()
        if self.path == '/':
            self.send_response(403)
            self.end_headers()
        elif self.path == '/FileToDelete.txt':
            delete_file("FileToDelete.txt")
        else:
            self.send_response(404)
            self.end_headers()
Implementing DELETE

summah@perspective ~/test/http $ date && ls FileToDelete*
Thu Jun 18 19:55:49 EDT 2015
FileToDelete.txt
Implementing DELETE

sumanah@perspective ~/test/http $ date && ls FileToDelete*
Thu Jun 18 19:55:49 EDT 2015
FileToDelete.txt

>>> d = requests.delete("http://localhost:8000/FileToDelete.txt")
Implementing DELETE

```
sumanah@perspective ~/test/http $ date && ls FileToDelete*
Thu Jun 18 19:55:49 EDT 2015
FileToDelete.txt

>>> d = requests.delete("http://localhost:8000/FileToDelete.txt")

sumanah@perspective ~/test/http $ python serverwdelete.py
serving at port 8000
127.0.0.1 - - [18/Jun/2015 19:53:49] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [18/Jun/2015 19:56:52] "DELETE /FileToDelete.txt HTTP/1.1" 204 -
```
Implementing DELETE

```
sumanah@perspective ~/test/http $ date && ls FileToDelete*
Thu Jun 18 19:55:49 EDT 2015
FileToDelete.txt

sumanah@perspective ~/test/http $ date && ls FileToDelete*
Thu Jun 18 19:56:54 EDT 2015
ls: cannot access FileToDelete*: No such file or directory

sumanah@perspective ~/test/http $ 
```
Implementing `DELETE`

```python
>>> d.status_code
204
>>> d.reason
'No Content'
```
DELETE – good idea?
Underappreciated methods

PUT

“here you go”
I thought **POST** meant “here you go”
So what is **POST**, anyway?

The standard says it means:

“Above our pay grade; take this to the boss”

a.k.a. Overloaded **POST**
So what is **POST**, anyway?

Often, we use it for:

“Create a new item in this set”

a.k.a. **POST**-to-append
PUT VS. POST

**PUT** /cards/5

Body:

![Image of a cardboard box with rows of numbers]

Means:

“Put this picture at /cards/5.”

**POST** /cards/5

Body:

![Image of a cardboard box with rows of numbers]

Means:

“Tell the webapp that this picture applies to /cards/5 somehow – figure it out.”
“CRUD” & HTTP verbs

Create
  PUT

Read
  GET

Update
  PUT

Delete
  DELETE
PUT – good idea?
More underused methods

- **PATCH**
  
  update just part of this document/resource
PATCH – good idea?
More underused methods

- **PATCH**
  update just part of this document/resource

- **OPTIONS**
  ask what verbs the client’s allowed to use (for a specific path, or server-wide)
OPTIONS – good idea?
A super-cool method

HEAD

like GET, but just for metadata
GET vs. HEAD

Request:
- **GET** / HTTP/1.1

Response:
- Start-line
- Headers
- Body

Request:
- **HEAD** / HTTP/1.1

Response:
- Start-line
- Headers
HEAD saves time

sumanah@perspective ~/test/http $ ipython
HEAD saves time

```
sumanah@perspective ~/test/http $ ipython

In [1]: import requests

In [2]: uri = "https://upload.wikimedia.org/wikipedia/commons/3/3c/Lisebergskaninen_liseberg_goteborg_sweden_20100718.jpg"

In [3]: %timeit rfull = requests.get(uri)
1 loops, best of 3: 1.33 s per loop

In [4]: %timeit rhead = requests.head(uri)
1 loops, best of 3: 163 ms per loop
```
HEAD saves time

In [3]: %timeit rfull = requests.get(uri)
1 loops, best of 3: **1.33 s per loop**

In [4]: %timeit rhead = requests.head(uri)
1 loops, best of 3: **163 ms per loop**
You don’t need the body to check:

Does it exist?
Do I have permission to **GET** it?

- **Content-Length**
- **Last-Modified**
- **Content-Type**
- **ETag**
- **Retry-After**
HEAD – good idea?
Popular headers include:

Content-Type
Content-Length
Popular headers include:

Also known as MIME or Mime

- Content-Type
- Content-Length
Popular headers include:

- text/*
- Content-Type
- Content-Length
Popular headers include:

- `Content-Type`
- `Content-Length`
- `application/*`
Popular headers include:

- chemical/*
- Content-Type
- Content-Length
Popular headers include:

Content-Encoding
Accept-Encoding
Content-Language
Accept-Language
More headers

ETag
If-Match
If-None-Match
More headers

If-Modified-Since
If-Unmodified-Since
Last-Modified
Cache-Control
A popular header

User-Agent
An unpopular header

From

The email address of the person making the request
Uses for From

Really bad auth
Uses for From

“Yes, I saw your site launch”
Uses for From

Coded messages meant for network surveillor
From – bad idea
Another spy trick

“Each header field consists of a case-insensitive field name followed by a colon (":")...”

– Internet Engineering Task Force (IETF) RFC 7230
  Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing

So: vary the case of the headers you send!!!
Header casing as secret info channel – bad idea
A popular header

Host
A required header

Host

required in request messages
Host & path work together

$ netcat myhostname.tld 80
GET /bicycle HTTP/1.1
Host: myhostname.tld
Host & path work together
Host & path work together

<table>
<thead>
<tr>
<th>Headers</th>
<th>Preview</th>
<th>Response</th>
<th>Cookies</th>
<th>Timing</th>
</tr>
</thead>
</table>

**General**
- Remote Address: 162.242.246.191:80
- Request URL: http://www.astoriabookshop.com/event/storytime-73
- Request Method: GET
- Status Code: 200 OK

**Response Headers (16)**

**Request Headers**
- GET /event/storytime-73 HTTP/1.1
  - Host: www.astoriabookshop.com
Host & path work together
Host & path work together
A popular header

Host

(wait –
why do we need to repeat this?
It's in the URL!
right?)
How Host helps

HTTP
is separate from
the Domain Name System
How Host helps

Host helps route requests among different domains that sit on the same server
Examples of virtual hosts

www.debian.org
Examples of virtual hosts

bugs.debian.org
Examples of virtual hosts

lists.debian.org
Examples of virtual hosts

wiki.debian.org
But watch out...

```
sumanah@perspective ~/test/http $ netcat www.debian.org 80
GET / HTTP/1.1
Host: fdjsahfakjhfkwf
```
But watch out...

HTTP/1.1 200 OK
Date: Tue, 16 Jun 2015 20:57:36 GMT
Server: Apache
Last-Modified: Thu, 09 Oct 2014 22:11:40 GMT
ETag: "lea-50504b789ac12"
Accept-Ranges: bytes
Content-Length: 490
Vary: Accept-Encoding
X-Clacks-Overhead: GNU Terry Pratchett
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
  <title>Welcome to mirror-csail!</title>
</head>
<body>

<h1>Welcome to mirror-csail!</h1>

This is mirror-csail, a system run by and for the <a href="http://www.debian.org">Debian Project</a>. She does stuff. What kind of stuff and who our kind sponsors are you might learn on <a href="http://db.debian.org/machines.cgi?host=mirror-csail">db.debian.org</a>.

<p>
<hr noshade />
<font size="-1">DSA</font>
</p>

</body>
</html>
But watch out...

Welcome to mirror-csail!

This is mirror-csail, a system run by and for the Debian Project.
A spam story
A spam story

My 404 logs (Drupal admin console):

TYPE page not found
DATE Thursday, October 9, 2014 – 10:46
USER Anonymous (not verified)
LOCATION http://myphishingsite.biz
REFERRER ttp://myphishingsite.biz
MESSAGE ttp://myphishingsite.biz
SEVERITY warning
HOSTNAME [IP address]
A spam story

My 404 logs (Drupal admin console):

TYPE page not found
DATE Thursday, October 9, 2014 – 10:46
USER Anonymous (not verified)
LOCATION http://myphishingsite.biz
REFERRER http://myphishingsite.biz
MESSAGE http://myphishingsite.biz
SEVERITY warning
HOSTNAME [IP address]
A spam story

My access logs:

[IP address] -- --
"GET http://myphishingsite.biz HTTP/1.1" 404 7574 "-" [User-Agent]
A spam story

Legit mistakes would look like:

[IP address]  --  
"GET /http://berkeley.edu HTTP/1.1"
404 7574 "-" [User-Agent]
A spam story

Intentionally malform your request!

$ netcat myhostname.tld 80
GET http://spam.com HTTP/1.1
Host: spam.com
A spam story

Intentionally malform your request!

$ netcat myhostname.tld 80
GET /viagra-bitcoin HTTP/1.1
Host: spam.com
404 spamming – bad idea
Define your own header!

“Header fields are fully extensible: there is no limit on the introduction of new field names, each presumably defining new semantics, nor on the number of header fields used in a given message.”

–(RFC 7230)
Define your own header!

x–blah–blah–blah
Define your own header!

x-Clacks-Overhead:
GNU Terry Pratchett
Define your own header!

```
sumanah@perspective ~ $ nc -l 8000 <>!
HTTP/1.1 200 OK
Glub: Sumanariffic
Content-Type: text/html

<b>I am a field.</b> <i>Yay!</i>
```
Define your own header!

I am a field. Yay!
Define your own header!

Request URL: http://localhost:8000/
Request method: GET
Status code: 200 OK

Response headers (0.063 KB)

Content-Type: "text/html"

GHub: "Sumanariffic"
Define your own header!

```
sumanah@perspective ~ $ nc -l 8000 <<<
> HTTP/1.1 200 OK
> Glub: Sumanariffic
> Content-Type: text/html
>
> <b>I am a field.</b> <i>Yay!</i>
> !
GET / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US, en; q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Cache-Control: max-age=0
```
Defining your own headers – good idea?
Status codes

100 & 101: Informational

2xx: Successful

3xx: Redirection

4xx: Client error

5xx: Server error
Status (response) codes

404  Not Found

Code  Reason-phrase
Status (response) codes

“A client SHOULD ignore the reason–phrase content.”
Heard of these?

- **410 Gone**
  
  It was here, but now it’s not.

- **304 Not Modified**
  
  You said, ‘**GET** this, if it’s been modified since [date]’. It hasn’t been.
451 Unavailable For Legal Reasons

Server is legally required to reject client’s request
451 Unavailable For Legal Reasons

Can’t let you see that; it’s censored.
451 Unavailable For Legal Reasons

“This is considered a client-side error even though the request is well formed and the legal requirement exists on the server side. After all, that representation was censored for a reason. There must be something wrong with you, citizen.”

—RESTful Web APIs, Leonard Richardson & Mike Amundsen
451 Unavailable For Legal Reasons

not quite a standard yet

451 – good idea?
WTF responses

All of these were found in the wild
WTF responses

Code: 126

Reason: Incorrect key file for table '/tmp/mysqltmp/#sql_13fb_2.MYI'; try to repair it
SQL=SHOW FULL COLUMNS FROM `y4dnu_extensions`
WTF responses

Code: 301

Reason: explicit_header_response_code
WTF responses

Code: 403

Reason: You've got to ask yourself one question: Do I feel lucky?
WTF responses

Code: 403

Reason: can't put wasabi in bed
WTF responses

Code: 404

Reason: HTTP/1.1 404
WTF responses

Code: 404

Reason: Not Found

<?php Header("cache-Control: no-store, no-cache, must-revalidate") ?>
WTF responses

Code: 200

Reason: Forbidden
WTF responses

Code: 404

Reason: Apple WebObjects
WTF responses

Code: 404

Reason: forbidden
WTF responses

Code: 434

Reason: HTTP/1.1 434
WTF responses

Code: 451

Reason: Unknown Reason-Phrase
WTF responses

Code: 503

Reason: Backend is unhealthy
WTF responses

Code: 520

Reason: Origin Error
WTF responses

Code: 525

Reason: Origin SSL Handshake Error
WTF responses

Code: 533

Reason: mtd::http: Unknown: Banned
WTF responses

Code: 732

Reason:
http://www.[hostname].com/intro/copyright.php
WTF responses

Code: 999

Reason: Request denied
Changing Reason-phrases

```python
import http.server

class OddHeaderHTTPRequestHandler(http.server.SimpleHTTPRequestHandler):
    responses = dict(http.server.SimpleHTTPRequestHandler.responses)
    responses[200] = ('Oll Korrect', 'Oh Kay')
    responses[404] = ('I Know Nothing', 'Nothing here by that name')
```

more at https://gitlab.com/brainwane/http-can-do-that/
Rockin'

This is a pretty rockin' site. I'm glad you decided to visit.

Wheee!
Request URL: http://localhost:8000/
Request method: GET
Status code: 200 Oll Korrect
Bespoke status codes/reasons – good idea?
There’s so much more

- “Don’t cache this”
- **Pragma** – pass instructions to server/client
- **CONNECT, TRACE, LINK, & UNLINK** methods
- 409 **Conflict**
- Look-before-you-leap requests
- Resources at HTTPS vs. HTTP URLs can differ
- “q” and preference ranking in the Accept header
- **Content-Disposition** (e.g. “attachment”)
The feeling of power

The sense of wonder
What might the web have been?

What might it still be?
Read & play

- RFCs 7230–7235
- `netcat, wget, netstat, telnet`
- basic HTTP servers (in your favorite language)
- https://gitlab.com/brainwane/http-can-do-that/
Thanks

Leonard Richardson
Greg Hendershott
Zack Weinberg
The Recurse Center
Clay Hallock
Paul Tagliamonte

Julia Evans, Allison Kaptur, Amy Hanlon, and Katie Silverio
Thank you

Sumana Harihareswara

http://brainwane.net

@brainwane

https://gitlab.com/brainwane/http-can-do-that/

sumanah@panix.com