CMPUT 410: HTTP Part I

Abram Hindle
abram.hindle@ualberta.ca
Department of Computing Science
University of Alberta
http://softwareprocess.es/

Context: FTP vs HTTP

- Transfers Files
- Directory oriented
- Out of band communication
- Some firewall issues with server connecting back to client (optional)
- 200 OK
- Anonymous by convention – must login everytime.

- Transfers content
- Request oriented
- GET/POST/DELETE/PUT/HEAD/etc.
- Allows arguments to accompany commands.
- 200 OK
- Anonymous by default
- Custom headers
- Custom Arguments and Bodies

Context: Gopher vs HTTP

- Transfers Files
- Directory oriented
- Simple
- Hypertext
- Rigid HTML
- Death by Licensing and Adoption

- Transfers content
- Request oriented
- GET/POST/DELETE/PUT/HEAD/etc.
- Allows arguments to accompany commands.
- 200 OK
- Anonymous by default
- Custom headers
- Custom Arguments and Bodies

HTTP

- Hypertext – “over” text
- Transport – Move it/Communicate it
- Protocol – A method of communication
- Accepted custom headers – allowing for extension
- Allowed for a more request/command oriented protocol (remember the command pattern)
- Relied of the pairing of web clients and web servers
- Relies on URIs to describe resources, allows more than 1 resource to be hosted on 1 server

If you don't listen to me

- Read this:

  http://tools.ietf.org/html/rfc2616

- Request for Comments:
  - Hypertext Transfer Protocol – HTTP/1.1
  - IETF's definition of HTTP/1.1
- No matter what I say about HTTP, that's the word.

HTTP Basics

- HTTP uses TCP (usually)
- HTTP uses TCP Port 80 (usually)
- HTTPS allows for ENCRYPTED HTTP
- HTTPS uses port TCP 443 (usually)
- HTTP can work over IPV4 and IPV6
- HTTP requests are made to addresses called URIs

HTTP Commands made to URIs

- GET – Retrieve information from that URI
- POST – Post data, append data, change data
- HEAD – GET without a message body (for caching)
- PUT – Store the entity at the that URI
- DELETE – Delete the resource at that URI
- OPTIONS – What options a resource can accomidate
- TRACE – Debugging / Echo Request
- CONNECT – Tunneling over HTTP

Toe-mate-oh/Toe-mot-oh

- URI
  - Universal Resource Identifier
  - Some URIs (most) are URLs
  - Scheme: http, ftp, mailto, crid, file
  - String identifies a resource
  - Absolute and relative

    http://softwareprocess.es/static/SoftwareProcess.es.html

    http://softwareprocess.es/static/../static/SoftwareProcess.es.html

    http://softwareprocess.es/static/SoftwareProcess.es.html#someAnchor

  - https://tools.ietf.org/html/rfc3986

Example HTTP URI

- http://
- username:password@ (optional)
- hostname
- :port (optional)
- /path/to/resource/resource.html
- http://username:password@hostname :port/path/to/resource/resource.html
- Password syntax not used anymore

Examples URIs

ftp://ftp.is.co.za/rfc/rfc1808.txt

http://www.ietf.org/rfc/rfc2396.txt

ldap://[2001:db8::7]/c=GB?objectClass?one

mailto:John.Doe@example.com

news:comp.infosystems.www.servers.unix

tel:+1-816-555-1212

telnet://192.0.2.16:80/

urn:oasis:names:specification:docbook:dtd:xml:4.1.2

From https://tools.ietf.org/html/rfc3986

URIs can have a query portion.

- http://geocities.com/SoHo/yourwebpage.html?que
- Example URL with a query portion that has 1 argument.
- URI queries are separated from the path by a question mark: ?
- Often parameters are seperated by & or ;

  https://tools.ietf.org/html/rfc3986#page-23

URIs and URLs

- Why are URIs and URLs important to the web?

URIs versus Fantasy Literature

- True Names
  - Rumpelstiltskin
- The Laws of Magic
  - The LAW of NAMES – Knowing the complete true name of an entity gives one control over it. http://deoxy.org/lawsofmagic.htm
- URI
  - Knowing the true URI lets one request it.
    - Like that URI for the weather!
    - http://weather.noaa.gov/pub/data/observations/metar/decoded/CYEG.TXT

URIs and encoding

- Universal URIs have to reference anything
- Even paths with spaces and other characters!
- For HTTP assume our URIs are Unicode UTF-8encoded
- For characters that aren't in [-._~0-9a-zA-Z] we use % encoding.
  - %20 is space
  - %E3%82%A2 is ア　KATAKANA LETTER A in UTF-8
  - %e2%98%83 is ☃ (&#x2603; in HTML)
- Domain names can be unicode
  - http://☃.net/ which is converted to http://xn--n3h.net/

Example GET http://slashdot.org

- Request http://slashdot.org
- We see HTTP so we know it'll be the http protocol.
  - No port specified so assume TCP port 80

Example GET http://slashdot.org

- Open up a connection to port 80 slashdot.org
- Send

```
> GET / HTTP/1.1\r\n
> User-Agent: curl/7.29.0\r\n
> Host: slashdot.org\r\n
> Accept: */*\r\n
> \r\n
```

Example GET http://slashdot.org

- Open up a connection to port 80 slashdot.org
- Send

```
> GET / HTTP/1.1\r\n
> User-Agent: curl/7.29.0\r\n
> Host: slashdot.org\r\n
> Accept: */*\r\n
> \r\n
```

The Root Of Slashdot .org

Specify the host slashdot. org

Example GET http://slashdot.org

- Receive headers

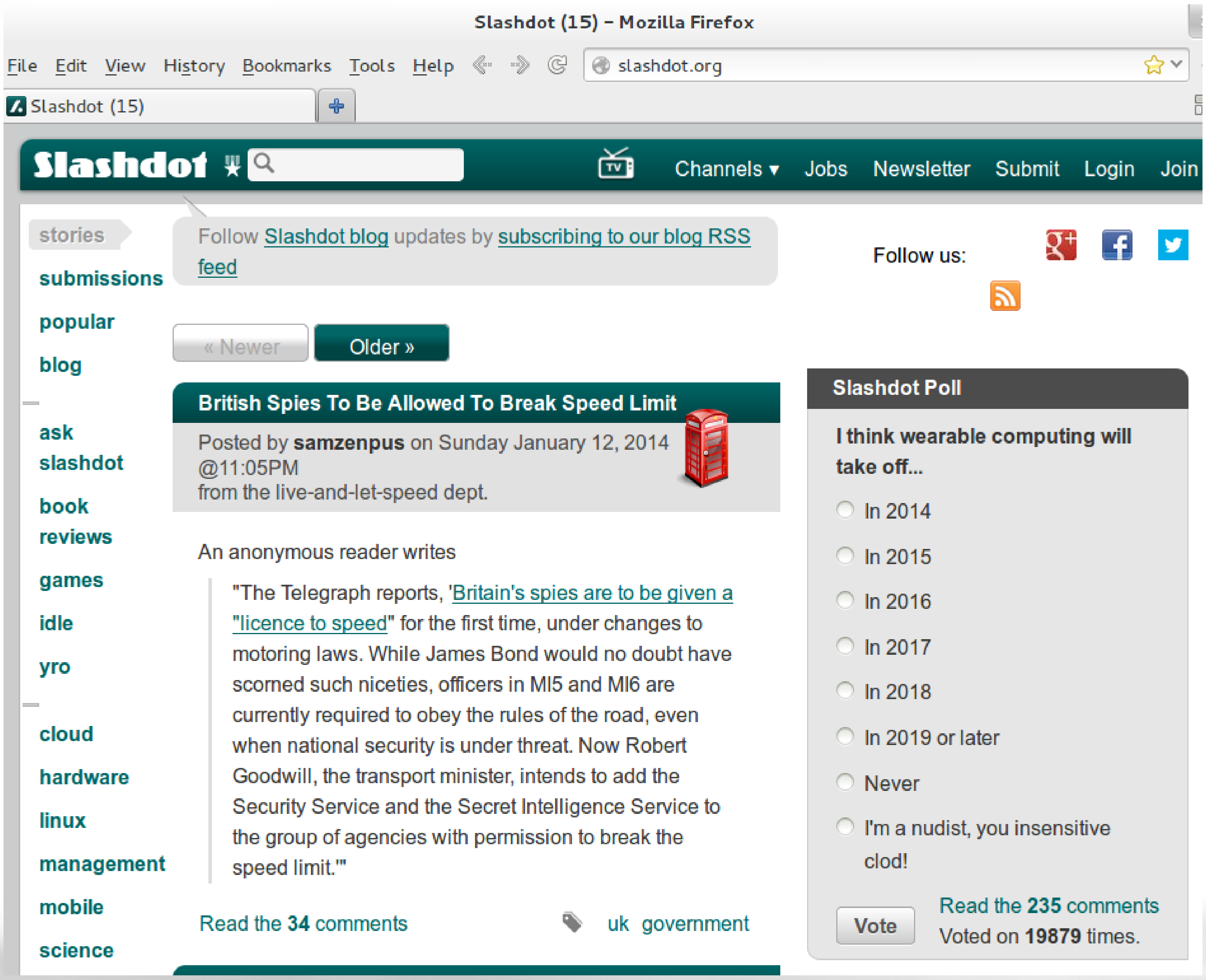
```
< HTTP/1.1 200 OK\r\n
< Server: Apache/2.2.3 (CentOS)\r\n
< SLASH_LOG_DATA: shtml\r\n
< Set-Cookie: betagroup=42; path=/; expires=Wed, 13-Jan-20
< Set-Cookie: betagroup=42; path=/; domain=.slashdot.org;
< Cache-Control: no-cache\r\n
< Pragma: no-cache\r\n
< X-XRDS-Location: http://slashdot.org/slashdot.xrds\r\n
< Content-Type: text/html; charset=utf-8\r\n
< Content-Length: 116473\r\n
< Date: Mon, 13 Jan 2014 04:56:37 GMT\r\n
< X-Varnish: 1432339936\r\n
< Age: 0\r\n
< Connection: keep-alive\r\n
< Vary: Accept-Encoding, User-Agent\r\n
< \r\n
```

Example GET http://slashdot.org

- Receive HTML

```
<!DOCTYPE html>\n
<html lang="en">\n
<head>\n
\n
\n
\n
\n
<script id="before-content" type="text/javascr
```

How it might look in a browser



Slashdot (15) – Mozilla Firefox

File Edit View History Bookmarks Tools Help slashdot.org

Slashdot (15)

Slashdot Channels ▾ Jobs Newsletter Submit Login Join

stories
submissions
popular
blog
ask slashdot
book reviews
games
idle
yro
cloud
hardware
linux
management
mobile
science

Follow Slashdot blog updates by subscribing to our blog RSS feed

Follow us:

« Newer Older »

British Spies To Be Allowed To Break Speed Limit

Posted by **samzenpus** on Sunday January 12, 2014 @11:05PM
from the live-and-let-speed dept.

An anonymous reader writes

> "The Telegraph reports, 'Britain's spies are to be given a "licence to speed" for the first time, under changes to motoring laws. While James Bond would no doubt have scorned such niceties, officers in MI5 and MI6 are currently required to obey the rules of the road, even when national security is under threat. Now Robert Goodwill, the transport minister, intends to add the Security Service and the Secret Intelligence Service to the group of agencies with permission to break the speed limit.'"

Read the **34** comments uk government

Slashdot Poll

I think wearable computing will take off...

- In 2014
- In 2015
- In 2016
- In 2017
- In 2018
- In 2019 or later
- Never
- I'm a nudist, you insensitive clod!

Vote

Read the **235** comments
Voted on **19879** times.

We used HTTP GET

- HTTP GET is a simple request to be sent that resource.
  - It might be dynamic (code)
  - It might be static (a file)
  - It might be a mixture
- We can send query parameters along with an HTTP get in the URI
- It is considered bad form to mutate data using a GET

HTTP POST

- Like a GET except the body of the HTTP Request contains data.
- Used for updating, creating, or general interaction with a URI
- Is not limited by URI length limits that impede HTTP GET
- Used to submit HTML Forms
- POSTs are expected to add or mutate data

HTTP POST

- Get parameters are url-encoded.
- So are POST parameters (usually) in a POST request body as
  - application/x-www-form-urlencoded
- They can also be sent following RFC 2388's format:
  - multipart/form-data
  - http://tools.ietf.org/html/rfc2388

HTTP POST Parameters

- If I want to send
  - Name: Abram Hindle
  - Occupation: Slide Maker
- I will encode it as:
  - Name=Abram+Hindle&Occuptation=Slide+Maker
  - Encoded using application/x-www-form-urlencoded

Example HTTP POST

```
hindle1@st-francis:~$ curl -X POST http://webdocs.cs.ualberta.ca/~hindle1/1.py --trace-ascii \
                      /dev/stdout -d 'What=1&Huh=2&Huh=3&args=4'
== Info: About to connect() to webdocs.cs.ualberta.ca port 80 (#0)
== Info:   Trying 129.128.184.6... == Info: connected
=> Send header, 257 bytes (0x101)
> POST /~hindle1/1.py HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
> Host: webdocs.cs.ualberta.ca
> Accept: */*
> Content-Length: 25
> Content-Type: application/x-www-form-urlencoded
>
=> Send data, 25 bytes (0x19)
> What=1&Huh=2&Huh=3&args=4
<= Recv headers
< HTTP/1.1 200 OK
< Date: Mon, 13 Jan 2014 23:41:45 GMT
< Server: Apache/2.2.3 (Red Hat)
< Connection: close
< Transfer-Encoding: chunked
< Content-Type: text/html; charset=UTF-8

<H3>Current Working Directory:</H3>
/compsci/webdocs/hindle1/web_docs


<H3>Command Line Arguments:</H3>

['/compsci/webdocs/hindle1/web_docs/1.py']


<H3>Form Contents:</H3>
<DL>
<DT>Huh: <i>&lt;type 'list'&gt;</i>
<DD>[MiniFieldStorage('Huh', '2'), MiniFieldStorage('Huh', '3')]
<DT>What: <i>&lt;type 'instance'&gt;</i>
<DD>MiniFieldStorage('What', '1')
<DT>args: <i>&lt;type 'instance'&gt;</i>
<DD>MiniFieldStorage('args', '4')
```

multipart/form-data

- http://tools.ietf.org/html/rfc2388
- Use mime to send form data
- Mostly used to upload files as binary
- Can be used for any forms.
- Sends the content-size first and then asks the server if that's OK.
  - Server responds HTTP/1.1 100 Continue if it can handle that data.
  - Then send the body
- Because of this interaction you can argue this is a slower method of posting since it requires the server to respond to the initial header before it sends the body.

Example HTTP POST

```
hindle1@st-francis:~$ curl  -F 'what=1' -F 'suzie=q' -X POST http://webdocs.cs.ualberta.ca/~hindle1/1.py
                      --trace-ascii /dev/stdout
== Info: About to connect() to webdocs.cs.ualberta.ca port 80 (#0)
== Info:   Trying 129.128.184.6... == Info: connected
=> Send header
> POST /~hindle1/1.py HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0 Ope
> nSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
> Host: webdocs.cs.ualberta.ca
> Accept: */*
> Content-Length: 235
> Expect: 100-continue
> Content-Type: multipart/form-data; boundary=--------------------
> --------9edfbc1fb1b0
>
<= Recv header
< HTTP/1.1 100 Continue
=> Send data
> ------------------------------9edfbc1fb1b0
> Content-Disposition: form-data; name="what"
>
> 1
> ------------------------------9edfbc1fb1b0
> Content-Disposition: form-data; name="suzie"
>
> q
> ------------------------------9edfbc1fb1b0--
<= Recv
< HTTP/1.1 200 OK
< Date: Mon, 13 Jan 2014 23:37:30 GMT
< Server: Apache/2.2.3 (Red Hat)
< Connection: close
< Transfer-Encoding: chunked
< Content-Type: text/html; charset=UTF-8
<
<H3>Form Contents:</H3>
<DL>
<DT>suzie: <i>&lt;type 'instance'&gt;</i>
<DD>FieldStorage('suzie', None, 'q')
<DT>what: <i>&lt;type 'instance'&gt;</i>
<DD>FieldStorage('what', None, '1')
```

Note the use of mime
And multipart/form-data and
The random boundary.

Resources: RFCs

- URIs https://tools.ietf.org/html/rfc3986
- HTTP http://tools.ietf.org/html/rfc2616

Resources: Encoding

- UCS versus UTF-8
  - http://lucumr.pocoo.org/2014/1/9/ucs-vs-utf8/
- UCS-2 is now UTF-16
  - http://en.wikipedia.org/wiki/UTF-16

Resources: DNS

- DNS
  - Domain Names
    - http://tools.ietf.org/html/rfc1035
    - http://tools.ietf.org/html/rfc1123
    - http://tools.ietf.org/html/rfc2181
  - Paul Vixie on DNS
    - http://queue.acm.org/detail.cfm?id=1242499
  - Tools
    - On Unix: nslookup and dig and whois and pwhois
    - http://network-tools.com/nslook/
  - IDN
    - http://www.unicode.org/faq/idn.html