CMPUT 410: HTTP Part II

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

HTTP PUT

- Like HTTP POST except the URI does not handle the request, it is the request.
  - The URI in a POST request identifies the resource that will handle the enclosed entity. That resource might be a data-accepting process, a gateway to some other protocol, or a separate entity that accepts annotations. In contrast, the URI in a PUT request identifies the entity enclosed with the request -- the user agent knows what URI is intended and the server MUST NOT attempt to apply the request to some other resource.

Fielding, et al. ,RFC2616: http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.6

HTTP POST verus PUT

- A URI identifies the handler.
- Arguments are stored in HTTP Request body/data section
- The HTTP Request body is interpreted and handled.

- A URI identifies the ENTITY
- Arguments are stored in HTTP Request body/data section
- The HTTP Request body/data is often or often contains the ENTITY

HTTP POST verus PUT

- Login
- Logout
- Search
- Query
- Summarize
- Entities being changed could be arguments
- “Post this request and handle it!”

- Create a new entity
- Update an existing entity
- Entity is referred to by the URI
- “Put this entity into that URI, or update that URI to be this entity I am sending”

HTTP DELETE

- Like HTTP POST except the URI does not handle the request, it is the request. And the action is deletion
  - The DELETE method requests that the origin server delete the resource identified by the Request-URI. This method MAY be overridden by human intervention (or other means) on the origin server. The client cannot be guaranteed that the operation has been carried out, even if the status code returned from the origin server indicates that the action has been completed successfully. However, the server SHOULD NOT indicate success unless, at the time the response is given, it intends to delete the resource or move it to an inaccessible location.
  - A successful response SHOULD be 200 (OK) if the response includes an entity describing the status, 202 (Accepted) if the action has not yet been enacted, or 204 (No Content) if the action has been enacted but the response does not include an entity.
  -

Fielding, et al. ,RFC2616: http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.7

HTTP POST verus DELETE

- A URI identifies the handler.
- Arguments are stored in HTTP Request body/data section
- The HTTP Request body is interpreted and handled.

- A URI identifies the ENTITY
- Arguments are stored in HTTP Request body/data section
- The HTTP Response is usually empty.
- The intent is to delete whatever entity resides at the request URI

HTTP POST verus DELETE

- Login
- Logout
- Search
- Query
- Summarize
- Entities being changed could be arguments
- “Post this request and handle it!”

- Delete an Entity
- Remove an Entity
- Entity is referred to by the URI
- “Delete this entity that resides at this URI”

HTTP POST/GET/DELETE/GET

- In the following example we use elastic search and we PUT, GET, DELETE and GET an entity from the service.
- PUT Stores the entity
- GET retrieves it
- DELETE deletes it

```
hindle1@piggy:~$ curl -v --trace-ascii /dev/stdout -H 'Content-type: application/json' \
    -X PUT http://cmput301.softwareprocess.es:8080/testing/junk/1 -d '{"name":"one"}'
== Info: About to connect() to cmput301.softwareprocess.es port 8080 (#0)
== Info:   Trying 192.30.35.214...
== Info: Connected to cmput301.softwareprocess.es (192.30.35.214) port 8080 (#0)
=> Send header, 162 bytes (0xa2)
0000: PUT /testing/junk/1 HTTP/1.1
001e: User-Agent: curl/7.29.0
0037: Host: cmput301.softwareprocess.es:8080
005f: Accept: */*
006c: Content-type: application/json
008c: Content-Length: 14
00a0:
=> Send data, 14 bytes (0xe)
0000: {"name":"one"}
== Info: upload completely sent off: 14 out of 14 bytes
<= Recv header, 17 bytes (0x11)
0000: HTTP/1.1 200 OK
<= Recv header, 47 bytes (0x2f)
0000: Content-Type: application/json; charset=UTF-8
<= Recv header, 20 bytes (0x14)
0000: Content-Length: 68
<= Recv header, 2 bytes (0x2)
0000:
<= Recv data, 68 bytes (0x44)
0000: {"ok":true,"_index":"testing","_type":"junk","_id":"1","_version
0040: ":5}
== Info: Connection #0 to host cmput301.softwareprocess.es left intact
{"ok":true,"_index":"testing","_type":"junk","_id":"1","_version":5}
```

PUT the entity {“name”:”one”}
To URI: http://host/testing/junk/1

```
hindle1@piggy:~$ curl -v --trace-ascii /dev/stdout -H 'Content-type: application/json' -X \
GET http://cmput301.softwareprocess.es:8080/testing/junk/1
== Info: About to connect() to cmput301.softwareprocess.es port 8080 (#0)
== Info:   Trying 192.30.35.214...
== Info: Connected to cmput301.softwareprocess.es (192.30.35.214) port 8080 (#0)
=> Send header, 142 bytes (0x8e)
0000: GET /testing/junk/1 HTTP/1.1
001e: User-Agent: curl/7.29.0
0037: Host: cmput301.softwareprocess.es:8080
005f: Accept: */*
006c: Content-type: application/json
008c:
<= Recv header, 17 bytes (0x11)
0000: HTTP/1.1 200 OK
<= Recv header, 47 bytes (0x2f)
0000: Content-Type: application/json; charset=UTF-8
<= Recv header, 21 bytes (0x15)
0000: Content-Length: 100
<= Recv header, 2 bytes (0x2)
0000:
<= Recv data, 100 bytes (0x64)
0000: {"_index":"testing","_type":"junk","_id":"1","_version":5,"exist
0040: s":true, "_source" : {"name":"one"}}
== Info: Connection #0 to host cmput301.softwareprocess.es left intact
{"_index":"testing","_type":"junk","_id":"1","_version":5,"exists":true, "_source" :
{"name":"one"}}
```

GET the entity at URI:
http://host/testing/junk/1

```
hindle1@piggy:~$ curl -v --trace-ascii /dev/stdout -H 'Content-type: application/json' -X   \
  DELETE http://cmput301.softwareprocess.es:8080/testing/junk/1
== Info: About to connect() to cmput301.softwareprocess.es port 8080 (#0)
== Info:   Trying 192.30.35.214...
== Info: Connected to cmput301.softwareprocess.es (192.30.35.214) port 8080 (#0)
=> Send header, 145 bytes (0x91)
0000: DELETE /testing/junk/1 HTTP/1.1
0021: User-Agent: curl/7.29.0
003a: Host: cmput301.softwareprocess.es:8080
0062: Accept: */*
006f: Content-type: application/json
008f:
<= Recv header, 17 bytes (0x11)
0000: HTTP/1.1 200 OK
<= Recv header, 47 bytes (0x2f)
0000: Content-Type: application/json; charset=UTF-8
<= Recv header, 20 bytes (0x14)
0000: Content-Length: 81
<= Recv header, 2 bytes (0x2)
0000:
<= Recv data, 81 bytes (0x51)
0000: {"ok":true,"found":true,"_index":"testing","_type":"junk","_id":
0040: "1","_version":6}
== Info: Connection #0 to host cmput301.softwareprocess.es left intact
{"ok":true,"found":true,"_index":"testing","_type":"junk","_id": "1","_version":6}
```

DELETE the entity at URI:
http://host/testing/junk/1

```
hindle1@piggy:~$ curl -v --trace-ascii /dev/stdout -H 'Content-type: application/json' \
                    -X GET http://cmput301.softwareprocess.es:8080/testing/junk/1
== Info: About to connect() to cmput301.softwareprocess.es port 8080 (#0)
== Info:   Trying 192.30.35.214...
== Info: Connected to cmput301.softwareprocess.es (192.30.35.214) port 8080 (#0)
=> Send header, 142 bytes (0x8e)
0000: GET /testing/junk/1 HTTP/1.1
001e: User-Agent: curl/7.29.0
0037: Host: cmput301.softwareprocess.es:8080
005f: Accept: */*
006c: Content-type: application/json
008c:
<= Recv header, 24 bytes (0x18)
0000: HTTP/1.1 404 Not Found
<= Recv header, 47 bytes (0x2f)
0000: Content-Type: application/json; charset=UTF-8
<= Recv header, 20 bytes (0x14)
0000: Content-Length: 60
<= Recv header, 2 bytes (0x2)
0000:
<= Recv data, 60 bytes (0x3c)
0000: {"_index":"testing","_type":"junk","_id":"1","exists":false}
== Info: Connection #0 to host cmput301.softwareprocess.es left intact
{"_index":"testing","_type":"junk","_id":"1","exists":false}
```

GET the entity at URI:
http://host/testing/junk/1
But it has been
DELETED!

WEBDAV and GET/PUT/DELETE

- WebDav http://tools.ietf.org/html/rfc4918
- Like FTP but for the web.
  - Lets you upload to a URI using HTTP PUT
  - Download from a URI using HTTP GET
  - Delete an entity at a URI using HTTP DELETE
  - And make directory URIs with a new verb:
    - HTTP MKCOL

Why HTTP PUT/DELETE and WebDav?

- Why would we bother with HTTP PUT and DELETE when we have POST which can do both?

HTTP User Agent

- A client/browser
  - in RFCs it usually means the HTTP Client and often it means a browser.
- Represented in HTTP Requests by
  - User-agent:
    - Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:26.0) Gecko/20100101 Firefox/26.0
    - Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.63 Safari/537.36
    - Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.73.11 (KHTML, like Gecko) Version/7.0.1 Safari/537.73.11
    - Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko

Examples stolen from: http://techblog.willshouse.com/2012/01/03/most-common-user-agents/

HTTP Status Codes

- HTTP/1.1 200 OK – the 200 is the status code!
- Informational codes 1XX e.g. 100 CONTINUE
- Success codes 2XX e.g. 200 OK
- Redirection 3XX e.g. 301 Moved Permanently
- Client Error 4XX e.g. 404 Not Found (your fault)
- Server Error 5XX e.g. 500 Internal Server Error

See RF2616 http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html#sec10

HTTP Status Codes: 1XX

- HTTP/1.1 100 Continue
  - Used in mulipart and uploads
  - Tells the client to send the request body/data
  - The server had a choice to accept the request or not and it has decided to accept it.

HTTP Status Codes 2XX

- HTTP/1.1 **200** OK
  - The request has succeeded
- HTTP/1.1 201 Created
  - The request succeeded and a new resource was created and exists. Maybe a good response to a PUT request.
- HTTP/1.1 202 Accepted
  - The request was accepted but it might not be done yet. E.g. you submitted some work to be done and it might take a while.

HTTP Status Codes 2XX

- HTTP/1.1 204 No Content (rare)
  - Request fufilled but don't update your view/page.
- HTTP/1.1 205 Reset Content (rare)
  - Reset the view/page/form but the request was successful.
- HTTP/1.1 **206** Partial Content
  - The HTTP GET had a range header (e.g. continue a download) so only the range requested will be returned.

HTTP Status Codes: 3XX redirection

- There's always a reason why you want to redirect something.
  - HTTP allows for redirection.
  - Redirection is basically a cheap abstraction built into the protocol
  - Watch out for infinite redirects.

HTTP Status Codes: 3XX redirection

- HTTP/1.1 300 Multiple Choices (rare)
  - Responds with a list of choices, where the user agent choses.
- HTTP/1.1 **301** Moved Permanently
  - The resource has moved to the URI in the Location: header of the HTTP Response. Clients should update the location and go to the new URL if the request was an HTTP GET.
  - e.g.

    HTTP/1.1 301 Moved Permanently

    Location: http://yournewdomain.com/SEO-Tips
- HTTP/1.1 **302** Found
  - A temporary or alterable redirect. Clients shouldn't change their URIs. Good for load balancing.

HTTP Status Codes: 3XX redirection

- HTTP/1.1 303 See Other
  - Just like a 302, but you can redirect POSTs and other actions beyond a GET. Confuses many User Agents.
- HTTP/1.1 **304** Not Modified
  - If the HTTP was conditional, respond that the GET is allowed, but the document didn't change. So the message body won't be sent unless you GET again explicitly.
  - Basically, you already cached it and the server said nothing changed so don't bother.
- HTTP/1.1 305 Use Proxy (rare)
  - Repeat your request with this proxy in the Location header.
- HTTP/1.1 **307** Temporary Redirect
  - Like 302, but cacheable if according to headers. Furthermore cannot redirect POST, PUT, DELETE etc. automatically.

Redirection? Why?

- Why would we use HTTP Redirection?

Redirection? Why?

- Why would we use HTTP Redirection?
  - Performance
    - Load balancing
    - Redirecting dynamic or static content to servers meant for it.
  - Keeping a consistent name but using multiple servers.
  - Linking to other websites
  - Restructuring a website but keeping old links alive
  - …

HTTP Client Error Status Codes: 4xx

- It's all your fault
  - You're not allowed
  - You're wrong
  - You owe us money
  - You can't handle it
  - You're taking too long
  - You're in conflict
  - It ain't here and it ain't never coming back.
  - Your headers are bad
  - We can't meet your expectations

HTTP Client Error Status Codes: 4xx

- HTTP/1.1 400 Bad Request
  - Hey buddy, I can't read this garbage DO NOT SEND IT AGAIN.
- HTTP/1.1 **401** Unauthorized
  - You're not authorized to see this information.
- HTTP/1.1 402 Payment Required
  - Currently reserved, but you can return it. Supposedly MobileMe returns this if you didn't pay up!

HTTP Client Error Status Codes: 4xx

- HTTP/1.1 **403** Forbidden (Common)
  - The server understands the request, logging in or changing authorization will not help. Maybe it could answer your request but an administer disabled that ability.
- HTTP/1.1 **404** Not Found
  - You've got the wrong resource or path. Can't find what you're looking for. Droids? What droids?
- HTTP/1.1 405 Method not allowed
  - We don't accept that method (GET/HEAD/POST/PUT/DELETE/etc.) here

Page not found · GitHub – Mozilla Firefox

Edit View History Bookmarks Tools Help

GitHub, Inc. (US) https://github.com/abramhindle/dreamcodethatworks

age not found · GitHub

Username or Email Password Sign in

404

This is not the web page you are looking for.

Find code, projects, and people on GitHub:

Search

Contact Support — GitHub Status — @githubstatus

HTTP Client Error Status Codes: 4xx

- HTTP/1.1 406 Not Acceptable
  - The sever cannot respond in way that matches your request's accept header line. E.g. you asked for JSON and we can only serve XML, but it's your fault.
- HTTP/1.1 407 Proxy Authentication Required
  - We're not going to proxy your request till you authenticate.
- HTTP/1.1 408 Request Timeout
  - You took too long to send your request, we're not going to service you. Try again but faster next time.

HTTP Client Error Status Codes: 4xx

- HTTP/1.1 409 Conflict
  - The request is in conflict. Often used with PUT requests. Maybe you PUT an old version or there were 2 PUTs at once and the server can't resolve it.
- HTTP/1.1 **410** Gone
  - It ain't coming back. Don't even try. Remove it from archive.org too!
- HTTP/1.1 411 Length Required
  - I can't service a request without a length header!

HTTP Client Error Status Codes: 4xx

- HTTP/1.1 412 Precondition Failed
  - We can't fufill the request based on preconditions given in the HTTP headers.
- HTTP/1.1 **413** Request Entity Too Large
  - Hey we only wanted 64kb, you are trying to send us 64mb!
- HTTP/1.1 **414** Request-URI too long
  - This server has a limit for the URI length so it won't service your 5mb long URI. Sorry.

HTTP Client Error Status Codes: 4xx

- HTTP/1.1 **415** Unsupported Media Type
  - The format of the request is not understandable to the server. Could be used to ban MP3s from being uploaded.
- HTTP/1.1 **416** Request Range Not Satisfiable
  - You asked for a range of a file in a GET and it doesn't make sense. E.g. you ask to continue a 1mb file from the 1.1mb mark.
- HTTP/1.1 417 Expectation Failed
  - The server cannot meet the expect header requirements
  - Expect is a header saying I expect a HTTP 100 Continue from the server for this request.

HTTP Server Error Status 5xx

- It's the server's fault because:
  - The web dev messed up
  - The sysadmin messed up
  - Something is broken
  - We didn't implement that
  - The server is down
  - The request took too long
  - I don't support that version of HTTP

HTTP Server Error Status 5xx

- HTTP/1.1 **500** Internal Server Error
  - The server encountered an error fufilling this request. AKA your PHP/CGI script broke.
- HTTP/1.1 501 Not Implemented
  - The server can't fufill that request (like an HTTP PUT) because it was never addressed so it can't make a coherent 4XX error code.
- HTTP/1.1 **502** Bad Gateway
  - The server talks to another process to fufill this request and that other process isn't working so well. It's probably down. But the server is getting rejected.
  - See this on websites when they get too busy and the web server cannot open new connections or new requests to the web application.

HTTP Server Error Status 5xx

- HTTP/1.1 **503** Service Unavailable
  - The service is temporarily down. Something's broken and we'll bring it back up eventually.
- HTTP/1.1 **504** Gateway Timeout
  - The server talks to another process to fufill this request and that other process isn't responding fast enough. Very common when a webapp is overloaded.
- HTTP/1.1 505 HTTP Version Not Supported
  - Your request used the wrong HTTP version. A version the server no longer supports.
  - Twitter doesn't let you do HTTP/1.0 requests anymore

HTTP Errors: Client, Server, or Application?

- If there is a problem in your web application how should you respond?
  - 4XX codes?
  - 5XX codes?
  - 4XX or 5XX + HTML response?

HTTP Errors: Client, Server, or Application?

- If there is a problem in your web application how should you respond?
  - 4XX codes?
  - 5XX codes?
  - 4XX or 5XX + HTML response?
- Some suggest that business logic errors (software error reporting) should be done with the client without codes
- Some suggest that you should use HTTP status codes.
- Decide if the application is user facing and how you should handle it for your audience.
- Warning: IE only shows a few kb of HTML for 404 pages.

HTTP Headers

- There are 47+ standard HTTP Headers
- These headers have an effect on:
  - Authentication
  - Caching
  - Encoding
  - Partial Downloading
  - Content Type
- There are so many in the RFC that I won't go over all of them
- http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

HTTP Request Headers

- Accept: */*
  - Accept: video/ogg,video/*;
  - Accept header specifies the kind of media the client can handle.
- Accept-*:
  - Headers for encodings and language
    - e.g. English and UTF-8, German and ASCII
- Authorization:
  - Provides credentials, like a hashed challenge response.

HTTP Request Headers

- Connection
  - Keep-alive or close the connection
- Content-Length
  - The length in bytes of the body of the HTTP Request
- Content-Type
  - Mimetype and encoding of the data/body

HTTP Request Headers

- Host
  - The host and port number of the original URI.
  - Needed for virtualhosts and HTTP/1.1
- If-Modified-Since
  - Used in a GET or HEAD to determine if the content should be downloaded again because it was updated.
- Referer
  - Tells the server what was the last page you were at. Where did the client come from. Leaks information!

HTTP Request Headers

- User-Agent
  - What is the user-agent of the client

HTTP Response Headers

- Age: seconds
  - Age of a cached response – used for caching
- Allow: Method
  - The URI allows HTTP Method
- Cache-Control
  - How the client should cache or not cache the response
- Connection
  - Keep-alive or close the connection

HTTP Response Headers

- Content-Encoding: gzip
  - Usually the response has been compressed.
- Content-Length
  - The length in bytes of the body of the HTTP Response
- Content-Range
  - What bytes are returned (if partial)
- Content-Type
  - Mimetype and encoding of the data/body

HTTP Response Headers

- Date
  - Time of the message
- Expires
  - For cache reasons when does this content expire.
- Last-Modified
  - When the server thinks the resource was last modified
- Location
  - Used to redirect clients on 3XX responses

HTTP Response Headers

- Server
  - The “user-agent” field of the server.
- WWW-Authenticate
  - The challenge given with a 401 response when you aren't authenticated.

Custom HTTP Headers

- You used to use X- headers
- This is now deprecated
- Send whatever headers you want as long as they don't conflict with current interpretations.
- Deprecating the "X-" Prefix and Similar Constructs in Application Protocols
  - http://tools.ietf.org/html/rfc6648

Missing Topics

- Authetnication
- Caching

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