Unsafe Methods

• Safe defined in 4.2.1, RFC 7231
• Safe methods: read operations that do not change the status of the server
  – GET, HEAD, OPTIONS, TRACE
  – n.b.: in practice, GET can have side effects: http://www.foo.com/a/b/c.php?var1=foo&var2=bar

• Unsafe methods: write operations; change the state of a resource
  – PUT, POST, DELETE
Idempotent Methods

• Idempotent defined in 4.2.2 of RFC 7231
• Safe & Idempotent:
  – GET (no side effects), HEAD, OPTIONS, TRACE
• Unsafe & Idempotent
  – PUT, DELETE
• Unsafe & ~Idempotent
  – POST, GET (w/ side effects)
    • e.g. http://foo.edu/counter.cgi?action=increment&variable=x
PUT vs. POST

• PUT tells the server to use the uploaded entity to create a resource at the specified URI
  – Unix semantic equivalent:
    ```bash
echo "hello world" > /tmp/hw.txt
    ```

• POST tells the server to submit the uploaded entity to the existing resource at the specified URI
  – Unix semantic equivalent:
    ```bash
echo "hello world" | /usr/bin/spell
    ```
REST Idiom

• PUT / DELETE for existing URIs
  – http://example.org/staff/nelson

• POST to a collection to create a new resource
  – http://example.org/staff/
POST

• If the request does not result in a resource that can be identified with a URI, then the response codes should be:
  – 200 OK
    • An entity describing the result
  – 204 No Content
    • No description; user agent does not navigate to a new page/URI

• If the result does produce a URI identifiable resource, the result should be:
  – 201 Created, and:
    – “Location” header specifying the new URI
PUT

• If a new resource is created:
  – 201 Created
    • Response code is returned

• If an existing resource is modified:
  – 200 OK
    • If there is an entity describing the results
  – 204 No Content
    • If there is no entity describing the results
DELETE

• If the URI is successfully deleted, then valid response codes are:
  – 200 OK
    • If there is an entity describing the results
  – 204 No Content
    • If there is no entity describing the results
  – 202 Accepted
    • The request was understood, queued and might be successful in the future
    • An entity is returned with this response, but there is no provision for the server to relay the eventual success or failure of the original request
Failure Response Codes

• 403 Forbidden
  – Server understood the request, but will not honor it
  – Authentication will not help; do not repeat
• 405 Method Not Allowed
  – Method/URI combination not valid
  – cf. "501 Not Implemented"!
• 411 Length Required
  – “Content-Length” header is missing on client upload
• 413 Request Entity Too Large
  – Configurable server value; prevent DOS attacks
    • Note the “Content-Length” header may lie!
• 414 Request-URI Too Long
  – Configurable server value; prevent DOS attacks
• 415 Unsupported Media Type
  – E.g., server wants “application/json” but received “image/jpeg”
Reality…

• PUT and DELETE are rarely (never?) implemented as specified in the RFC
  – Security considerations, limited client support, incomplete semantics
  – PUT sometimes implemented by redirecting to a CGI script:
    • http://httpd.apache.org/docs/current/mod/mod_actions.html
  – Web Distributed Authoring and Versioning (WebDAV) is the preferred implementation for “write” operations
    • http://www.webdav.org/
• We will do neither approach; we’ll implement native support for unsafe methods
Allowing PUT and DELETE

• Recursively allow PUT / DELETE in a directory via these directives in *WeMustProtectThisHouse!* file:
  – ALLOW-PUT
  – ALLOW-DELETE

• Orthogonal to the uid/passwd info:

```bash
# ALLOW-PUT
ALLOW-DELETE
#
authorization-type=Basic
#
realm="Fried Twice"
#
 bda:9177d249338e2b2394f65faa17a46a29
 jbollen:6c4bea736ded1341eb8c507d4b0baa5b
 mln:ae33d20c70e59a4c734d9f2c19c0df56
 vaona:81e5a6b538844ed0c494149a96310a85
```
PUT Example

PUT /~mln/fairlane.txt HTTP/1.1
Host: www.cs.odu.edu
Connection: close
User-Agent: CS 595-s07 Automatic Testing Program
Content-type: text/plain
Content-length: 193
DELETE Example

DELETE /~mln/fairlane.txt HTTP/1.1
Host: www.cs.odu.edu
Connection: close
User-Agent: CS531 Automated Tester
Reminder: OPTIONS

• Be sure to give the correct values for the OPTIONS method
  – PUT, DELETE depend on the values in “WeMustProtectThisHouse!”
  – POSTing to URI that is not an executable file?
    • Apache seems to allow it…
      – But not to directories
      – 2018-11-07 update: Apache allows POST to both now
    • We will not (status 405)
POST

• Typically the result of HTML “Forms”
  – http://www.w3.org/TR/REC-html40/interact/forms.html#h-17.13.4

• Two types of values in the client’s “Content-type” request header:
  – application/x-www-form-urlencoded
    • (original & default)
  – multipart/form-data
    • Introduced in RFC-1867; allows file upload
      – http://www.ietf.org/rfc/rfc1867.txt
HTML Examples

<Form action="http://server.com/cgi/handle"
    enctype="application/x-www-form-urlencoded"
    method="post">
    <P>
    What is your name? <Input type="text" name="submit-name"><br>
    <Input type="submit" value="Send"> <Input type="reset">
</form>

<Form action="http://server.com/cgi/handle"
    enctype="multipart/form-data"
    method="post">
    <P>
    What is your name? <Input type="text" name="submit-name"><br>
    What files are you sending? <Input type="file" name="files"><br>
    <Input type="submit" value="Send"> <Input type="reset">
</form>

Based on examples from: http://www.w3.org/TR/REC-html40/interact/forms.html#h-17.13.4
The "encoding" in "enctype" refers to "urlencoded", not "Content-Encoding"
POST /~mln/foo.cgi HTTP/1.1
Host: www.cs.odu.edu
Connection: close
Referer: http://www.cs.odu.edu/~mln/bar.html
User-Agent: CS 595-s06 Automatic Testing Program
Content-type: application/x-www-form-urlencoded
Content-Length: 134

action=restore&manufacturer=ford&model=fairlane+500XL&year=1966
&status=modified&engine=427+sideoiler&transmission=4+speed+toploader

*Functionally the same as (modulo a possible 414 response):*

GET /~mln/foo.cgi?action=restore&manufacturer=ford&model=fairlane+500XL&year=1966
&status=modified&engine=427+sideoiler&transmission=4+speed+toploader HTTP/1.1
Host: www.cs.odu.edu
Connection: close
Referer: http://www.cs.odu.edu/~mln/bar.html
User-Agent: CS 595-s06 Automatic Testing Program

This has obvious limitations for sending 1) a lot of data, 2) non-ascii/binary data
POST /~mln/foo.cgi HTTP/1.1
Host: www.cs.odu.edu
Connection: close
Referer: http://www.cs.odu.edu/~mln/bar.html
User-Agent: CS 595-s06 Automatic Testing Program
Content-type: multipart/form-data; boundary=----------0xKhTmLbOuNdArY
Content-Length: 698

----------0xKhTmLbOuNdArY
Content-Disposition: form-data; name="action"
restore
----------0xKhTmLbOuNdArY
Content-Disposition: form-data; name="manufacturer"
ford
----------0xKhTmLbOuNdArY
Content-Disposition: form-data; name="model"
fairlane 500xl
----------0xKhTmLbOuNdArY
Content-Disposition: form-data; name="year"
1966
----------0xKhTmLbOuNdArY
Content-Disposition: form-data; name="picture"; filename="fairlane.txt"
Content-Type: text/plain

Note the “--” to indicate the end