Distributed Systems
Lecture 5

HTTP: Hypertext Transfer Protocol

Joseph Phillips

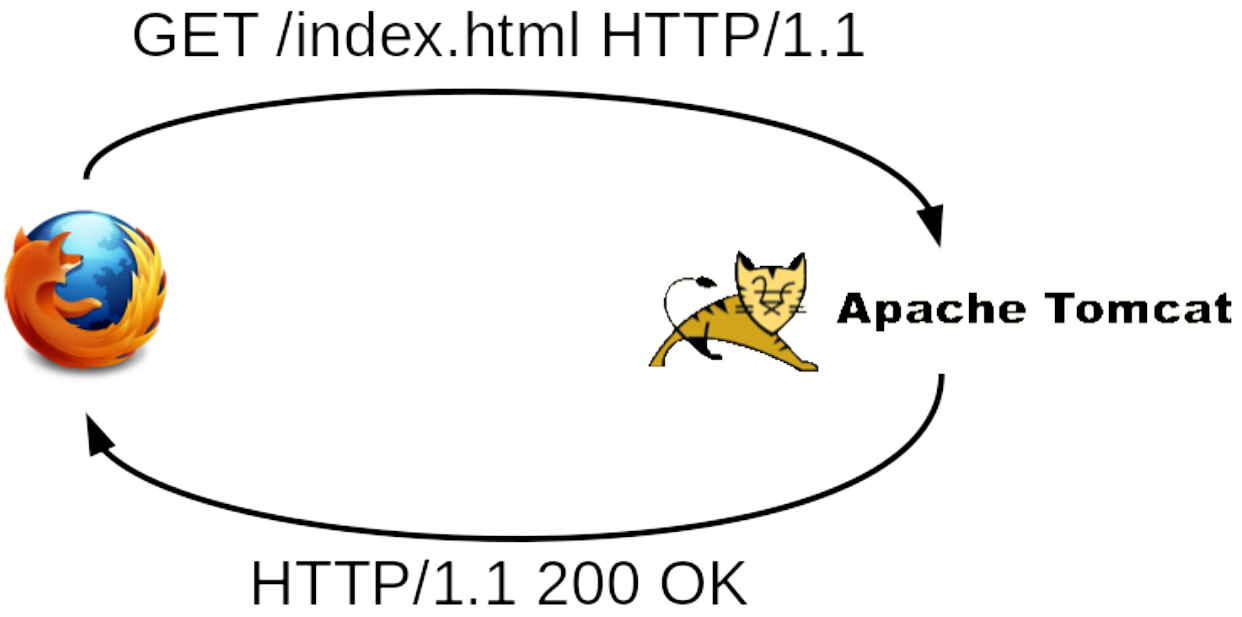
Last modified 2019 April 29

Topics

- Client to Server
- HTTP commands
  - GET
  - PUT
  - DELETE
  - POST
  - HEAD
  - OPTIONS and TRACE
- Server to Client
- Inspecting the communication
- Cookies

Motivation

Curious student "*Hmm, I wonder what **really goes on** when a **web browser** talks with a **web server**?*"



Excellent Question!

- HTTP = HyperText Transfer Protocol
- Tells what client sends to server
- Tells what server sends back to client
- What is replied?
  - Yes: HTML (HyperText Markup Language)
  - Also: Images, PDF files, Video, etc.

- HTTP/1.0
  1. Client connects to TCP port 80 on server
  2. Client sends message with header (and optional contents)
  3. Server sends back response
  4. Server close connection

Client to server:

```
GET https://www.eclipse.org/ HTTP/1.1
Host: www.eclipse.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
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, br
Cookie: PHPSESSID=qcop813fn2oad2p2tj53nka6kmbkbm0f
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Sat, 27 Apr 2019 23:25:24 GMT
Cache-Control: max-age=0
```

- Request line
- “Keyword: value” lines
  - Keywords are NOT case sensitive
  - Values may or may not be case sensitive
  - ASCII only for both
  - Continue long values on next line with space or tab on next line
- Blank line
- Optional message body (none for GET)
- Lines end with carriage-return/line-feed (\r\n)

Client to server: request line

GET https://www.eclipse.org/ HTTP/1.1

Host: www.eclipse.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
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, br
Cookie: PHPSESSID=qcop813fn2oad2p2tj53nka6kmbkbm0f
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Sat, 27 Apr 2019 23:25:24 GMT
Cache-Control: max-age=0

- Command + Path + Version
- Command
  - GET, PUT, POST and DELETE most common
- Path
- Version
  - HTTP/1.0: Closes connection after every call
  - HTTP/1.1: Can leave socket open, may make multiple requests

Client to server: Host

GET https://www.eclipse.org/ HTTP/1.1

Host: www.eclipse.org

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
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, br
Cookie: PHPSESSID=qcop813fn2oad2p2tj53nka6kmbkbm0f
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Sat, 27 Apr 2019 23:25:24 GMT
Cache-Control: max-age=0

- Tell server host client connects to
- (Allows server to handle multiple hosts from same IP address)

Client to server: User-Agent

GET https://www.eclipse.org/ HTTP/1.1
Host: www.eclipse.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
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, br
Cookie: PHPSESSID=qcop813fn2oad2p2tj53nka6kmbkbm0f
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Sat, 27 Apr 2019 23:25:24 GMT
Cache-Control: max-age=0

- Tells server which browser
  - And OS, machine architecture, etc.
- (Lets server optimize for this particular browser)

Client to server: Accept

```
GET https://www.eclipse.org/ HTTP/1.1
Host: www.eclipse.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
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, br
Cookie: PHPSESSID=qcop813fn2oad2p2tj53nka6kmbkbm0f
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Sat, 27 Apr 2019 23:25:24 GMT
Cache-Control: max-age=0
```

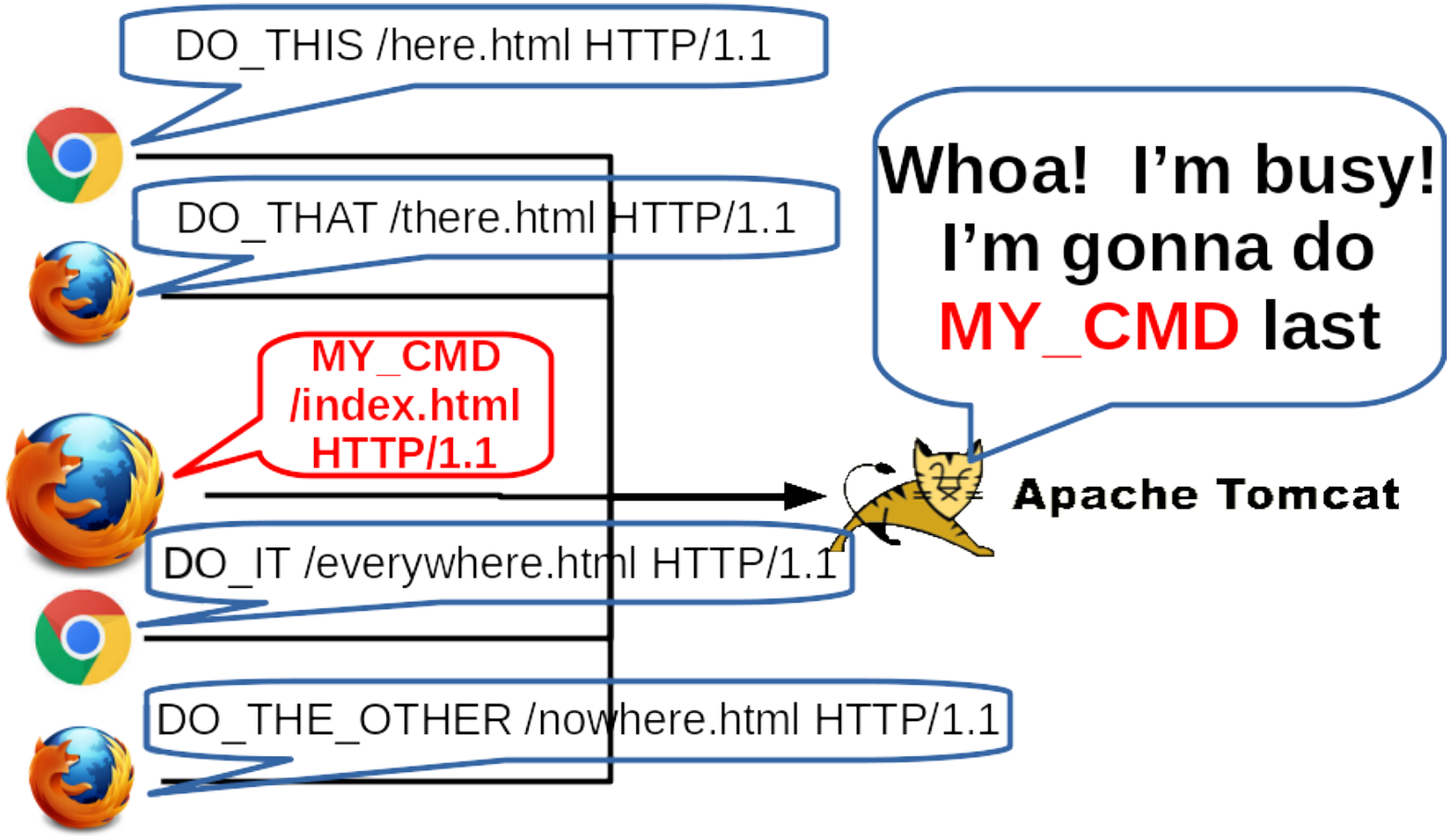
- Tells server which what types of data client can handle
  - Servers can ignore this
- MIME = Multipurpose Internet Mail Extensions (format: ***type/subtype***)
  - text/* : Human readable (e.g. text/html, text/plain)
  - image/* : Images (e.g. image/gif)
  - audio/* , video/* : Self-explanatory
  - application/* : binary
  - multipart/* : containers for documents and resources

Client to server: Connection

GET https://www.eclipse.org/ HTTP/1.1
Host: www.eclipse.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
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, br
Cookie: PHPSESSID=qcop813fn2oad2p2tj53nka6kmbkbm0f
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Sat, 27 Apr 2019 23:25:24 GMT
Cache-Control: max-age=0

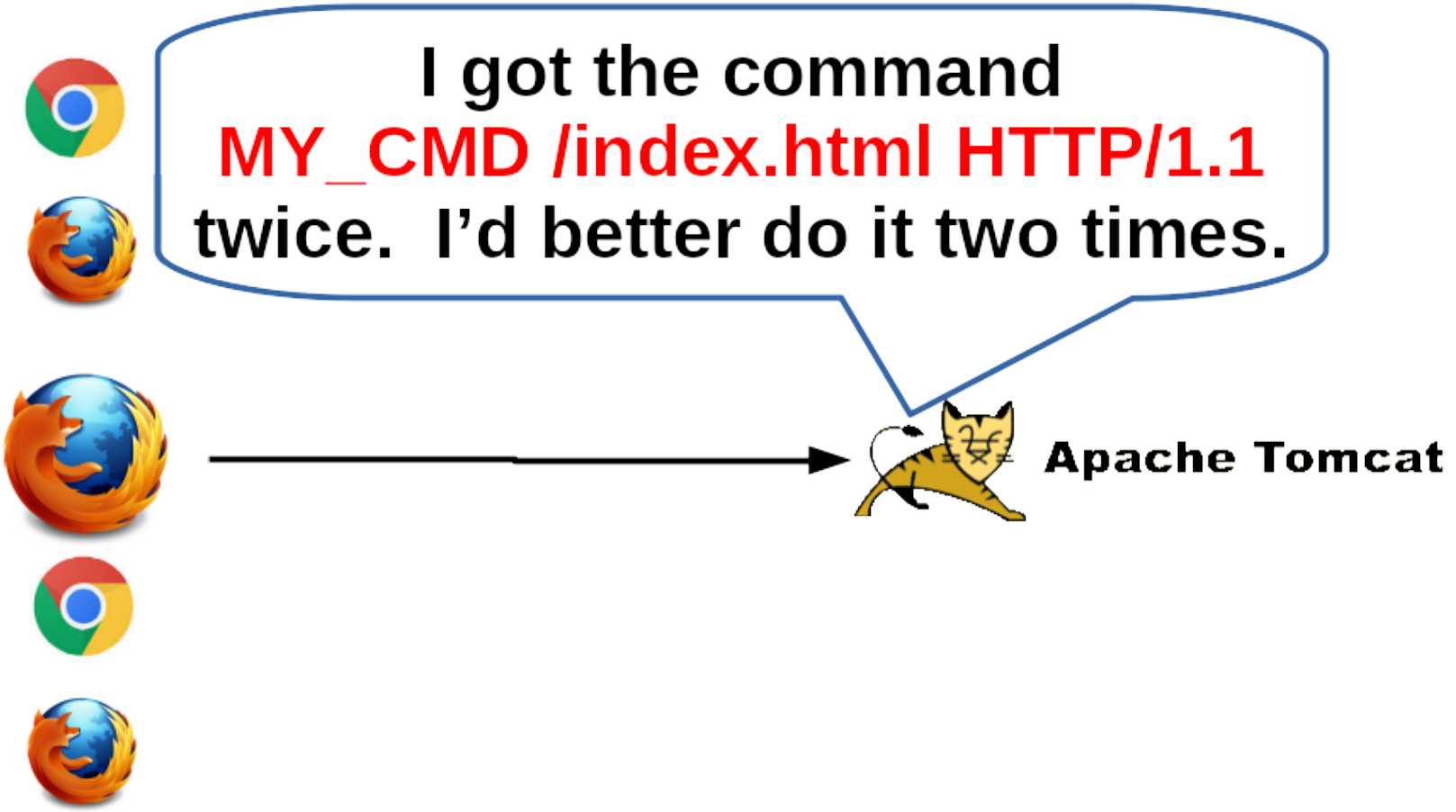
- Tells server is socket should be left open
  - close: no
  - keep-alive: yes
- HTTP/1.0
  - Automatically closed
- HTTP/1.1
  - Can leave open
  - Overcomes overhead of HTTPS connection using SSL or TLS

Consider a busy server . . .



. . . and an impatient client

Is it wrong to do MY_CMD twice?



Introducing “Idempotence!”

- *If **MY_CMD** is **idempotent**, doing it **one time** is the same as doing it **ten times**!*
  - impatient clients!
  - laggy networks!
  - overworked servers!
  - *linear combinations thereof!*
- HTTP cmds defined to be idempotent:
  - GET, PUT, DELETE

HTTP: GET

- Retrieves resource in header line
- Side-effect free
  - Can be redone if fails
  - If accidentally done multiple times, no problem!
- Can be:
  - Bookmarked
  - Pre-fetched
  - Cached

HTTP: PUT

- Uploads resource to server at given URL
- Not side-effect free, but . . .
- Is “idemponent”
  - Can be redone if fails
  - If accidentally done multiple times, no problem! (You just overwrite an exact copy of what was already there)

HTTP: DELETE

- Deletes resource from server at given URL
- Not side-effect free, but . . .
- Is “idemponent”
  - Can be redone if fails
  - If accidentally done multiple times, no problem!
    (You just tried to delete what was already gone)

HTTP: POST

- Uploads resource to server
- Not side-effect free
- ***Not*** “idemponent”
- Server can do what it wants
  - Change URL of resource
  - Update some other resource
  - Used to commit to some action (e.g. buy what is in shopping cart)
- Should be used for “unsafe” operations that should not be repeated without user permission

HTTP: HEAD

- Like GET
  - Only gets header
- Used to check modification date/time
  - To see if browser can rely on resource in its cache

HTTP: OPTIONS and TRACE

- OPTIONS:
  - Client browser "***Hey Server! What can I do with this resource?***"
- TRACE:
  - For debugging

GET vs PUT vs POST

- Elliotte Harold "***In practice, POST is vastly overused on the Web today***"
  - GET can handle URL lines of up to 2000 chars
  - For uploading large resources want to use PUT or POST anyway
- Joe Phillips "*I'm not so sure. POST is the safest*" "*That said, try do design for idempotence, so then can use GET, PUT and DELETE.*"

Request body

- Can be arbitrary bytes
- BUT browser should say what it is uploading
  - `Content-type`: "***What is it?***"
  - `Content-length`: "***Length in bytes***" (not including header)
- Example (notice blank line):

```
. . .
Content-type: application/x-www-form-urlencoded
Content-length: 54

username=Elliotte+Harold&email=elharo
%40ibiblio.org
```

Server to client

```
HTTP/1.1 200 OK
Server: nginx
Date: Sat, 27 Apr 2019 23:34:48 GMT
Content-Type: text/html
Content-Length: 6353
Connection: keep-alive
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Last-Modified: Sat, 27 Apr 2019 23:34:53 GMT
Expires: Sat, 27 Apr 2019 23:34:53 GMT
Vary: Accept-Encoding
Content-Encoding: gzip
X-NodeID: www-vm1
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=15552000;
includeSubDomains; preload
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Proxy-Cache: MISS
```

- Status line + header + blank line + requested resource

Server to client: Status line

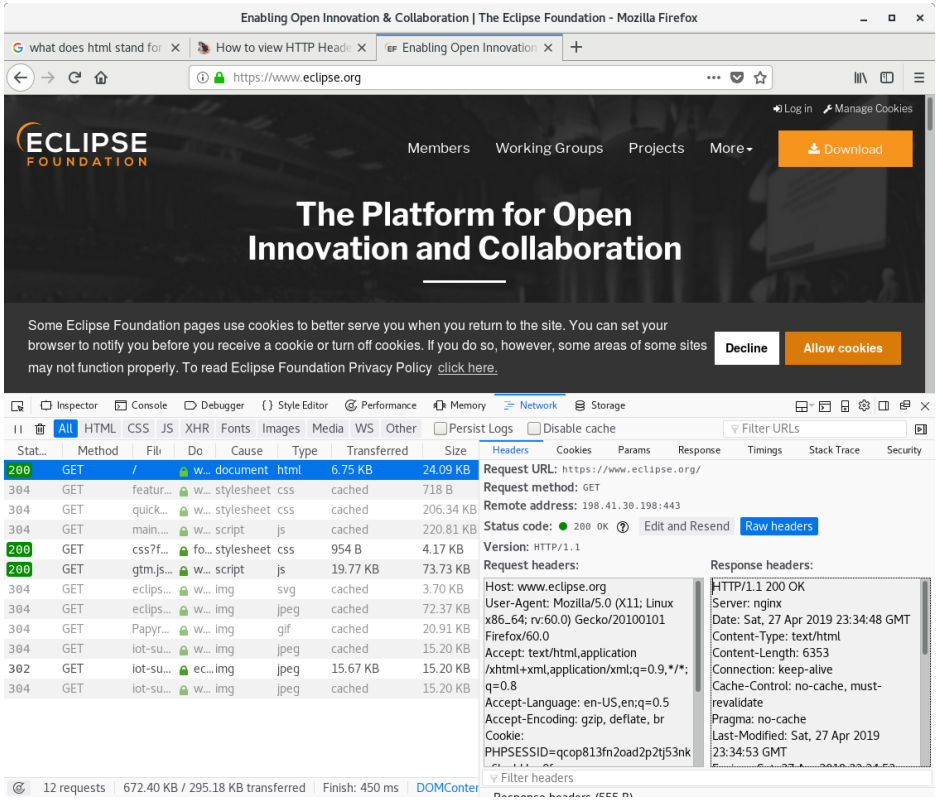
HTTP/1.1 200 OK

Server: nginx
Date: Sat, 27 Apr 2019 23:34:48 GMT
Content-Type: text/html
Content-Length: 6353
Connection: keep-alive
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Last-Modified: Sat, 27 Apr 2019 23:34:53 GMT
Expires: Sat, 27 Apr 2019 23:34:53 GMT
Vary: Accept-Encoding
Content-Encoding: gzip
X-NodeID: www-vm1
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=15552000; includeSubDomains; preload
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Proxy-Cache: MISS

- HTTP version
- Response code (a sample)
  - 1xx: Informational
  - 100: Continue “sending your big-ass resource”
  - 2xx: Request succeeded
  - 200: Ok
  - 201: Created
  - 202: Accepted
  - 203: Non-authoritative Information (e.g. from caching proxy)
  - 300: Multiple choices (e.g. PS and PDF)
  - 301: Moved permanently
  - 302: Moved temporarily
  - 4xx: Client error
  - 401: Unauthorized
  - 402: Payment required
  - 403: Forbidden
  - 404: Not found
  - 5xx: Server error

Seeing all this (firefox)

- Go to desired site
- Web Developer → Inspector → Network
- Refresh to reload
- Click “Headers”



Cookies

- Data that server gives to client browser to hold on to
  - ***name=value*** pairs
  - given back to server by client
- ***Could*** be
  - ITEM=Blue+suede+shoes&price=$99.95
  - Way more likely to be key to server db
- Limitations
  - Non-whitespace ASCII text
  - No commas or semicolons
- Server gives to client
  - Set-cookie: cart=ATVPDKIKX0DER
- Client gives back to client
  - Cookie: cart=ATVPDKIKX0DER

Cookie etiquette

- `www.mysite.com` can only set cookies for `www.mysite.com`
  - Not for `www.spammers.com`, or `.com`
- Therefore, `www.mysite.com` could embed an image from `www.spammers.com`,
  - Forces browser to go to `www.spammers.com`,
  - Allows `www.spammers.com` to install its cookies on browser, even though user did not go there
- Therefore, some browsers allow blocking ***third-party cookies***

Cookie attributes

- Domain (e.g. `Domain=.google.com`)
  - Browser will send cookie back to *.google.com, not just www.google.com
- Path (e.g. `Path=/restricted`)
  - Browser will only send cookie back to that directory and its subdirectories
- Expiration, several formats:
  - Absolute time:
    `expires=Wed, 21-Dec-2015 15:30:00 GMT`
  - Relative time (in seconds):
    `Max-Age=3600`
- Security:
  - “Never send over insecure channels” (attribute only, no value)
    `secure`
  - “Never give to Javascript, just the server” (attribute only, no value)
    `httponly`

Java HttpCookie class

```
public class HttpCookie implements
Cloneable {
  public HttpCookie (String name, String
value)
  public boolean hasExpired()
  public void setComment (String com)
  public String getComment ()
  public void setCommentURL (String
url)
  public String getCommentURL ()
  public void setDiscard (boolean disc)
  public boolean getDiscard ()
  public void setPortList (String ports)
  public String getPortList()
  public void setDomain (String domain)
  public String getDomain ()
  public void setMaxAge (long entry)
  public long getMaxAge ()
  public void setPath (String path)
  public String getPath ()
  public void setSecure (boolean flag)
  public boolean getSecure ()
  public String getName()
  public void setValue (String value)
  public String getValue()
  public int getVersion()
  public void setVersion (int v)
  public static boolean domainMatches
(String domain, String host)
  public static List<HttpCookie> parse
(String header)
  public String to String()
  public boolean equals (Object obj)
  public int hashCode ()
  public Object clone()
} //  obsolete in red
```

Java: CookieManager

```
CookieManager  manager = new CookieManager();

manager.setCookiePolicy(CookiePolicy.ACCEPT_ORIGINAL_SERVER);
    //  CookiePolicy.ACCEPT_ALL
    //  CookiePolicy.ACCEPT_NONE
    //  CookiePolicy.ACCEPT_ORIGINAL_SERVER
CookieHandler.setDefault(manager);
```

Finer-grained with `public boolean shouldAccept (URI uri, HttpCookie cookie)`

```
  public class NoGovernmentCookies implements CookiePolicy  {
    @Override
    public boolean shouldAccept (URI uri, HttpCookie cookie) {
      return( !
   (uri.getAuthority().toLowercase().endswith("gov")
                    ||
                  uri.getDomain().toLowercase().endswith("gov"))
             );
    }
  }
```

Java: CookieStore

- Allows saving cookie between sessions
  - CookieStore store = manager.getCookieStore();

- Methods:
  - public void add (URI uri, HttpCookie cookie)
  - public List<HttpCookie> get(URI uri)
  - public List<HttpCookie> getCookies()
  - public List<URI> getURIs()
  - public boolean remove (URI uri, HttpCookie cookie)
  - public boolean removeAll()

Example Java Cookie program

```
//  By Rishabh Mahrsee
//  From https://www.geeksforgeeks.org/java-net-httpcookie-java/
//  Downloaded 2019 Apr 28
import java.io.IOException;
import java.net.CookieHandler;
import java.net.CookieManager;
import java.net.CookieStore;
import java.net.HttpCookie;
import java.net.URL;
import java.net.URLConnection;
import java.util.List;

public class httpcookie1
{

    public static void main(String[] args) throws IOException
    {

        if  (args.length < 1)
        {
          System.err.println("Usage:\tjava httpcookie1 <url>");
          System.exit(1);
        }

        String urlString = args[0];

        if  ( !urlString.substring(0,4).equals("http") )
        {
          urlString = "http://" + urlString;
        }

        // Create a default system-wide CookieManager
        CookieManager cookieManager = new CookieManager();

        CookieHandler.setDefault(cookieManager);

        // Open a connection for the given URL
        URL url = new URL(urlString);
        URLConnection urlConnection = url.openConnection();
        urlConnection.getContent();

        // Get CookieStore which is the default internal in-memory
        CookieStore cookieStore = cookieManager.getCookieStore();

        // Retrieve all stored HttpCookies from CookieStore
        List<HttpCookie> cookies = cookieStore.getCookies();

        int cookieIdx = 0;

        // Iterate HttpCookie object
        for (HttpCookie ck : cookies) {

            System.out.println("------ Cookie." + ++cookieIdx + " -------");

            // Get the cookie name
            System.out.println("Cookie name: " + ck.getName());

            // Get the domain set for the cookie
            System.out.println("Domain: " + ck.getDomain());

            // Get the max age of the cookie
            System.out.println("Max age: " + ck.getMaxAge());
```

Example Java Cookie program, cont’d

```
        // Get the path of the server
        System.out.println("Server path: " + ck.getPath());

        // Get boolean if the cookie is being restricted to a secure
        // protocol
        System.out.println("Is secured: " + ck.getSecure());

        // Gets the value of the cookie
        System.out.println("Cookie value: " + ck.getValue());

        // Gets the version of the protocol with which the given cookie is
        // related.
        System.out.println("Cookie protocol version: " + ck.getVersion());

    }
  }

}
```

Your Turn

Write a program that lets the user find and edit a cookie

C/libcurl

```
//  Like iterating over a list of struct addrinfo given by getaddrinfo()
CURLcode res;
struct curl_slist *cookies;

//  get linked list
res = curl_easy_getinfo(curl, CURLINFO_COOKIELIST, &cookies);
if(res != CURLE_OK)
{
    fprintf(stderr, "Curl curl_easy_getinfo failed: %s\n",curl_easy_strerror(res));
    exit(1);
}

//  iterate over linked list
while(nc)
{
  printf("[%d]: %s\n", i, nc->data); // *all* the data is in the "data" member var
  nc = nc->next;
}

//  free() list
curl_slist_free_all(cookies);
```

C/libcurl

- "***So, how to I get the data out of the*** `data` ***member var?***"
- Uses an old Netscape format of tab-separated fields
  - domain (a char array)
  - isAccessible (a char array holding TRUE or FALSE)
  - path (a char array)
  - isSecure (a char array holding TRUE or FALSE)
  - expiration (Unix `time_t`: an integer telling time since 1970 Jan 1)
  - name (a char array)
  - value (a char array)

C/libcurl

```
//  To delete all cookies:
curl_easy_setopt(curl, CURLOPT_COOKIELIST, "ALL");

//  To add cookies (note old Netscape format)
snprintf(nline, sizeof(nline), "%s\t%s\t%s\t%s\t%lu\t%s\t%s",
        ".example.com", "TRUE", "/", "FALSE",
        (unsigned long)time(NULL) + 31337UL,
        "PREF", "hello");
res = curl_easy_setopt(curl, CURLOPT_COOKIELIST, nline);
if(res != CURLE_OK) {
  fprintf(stderr, "Curl curl_easy_setopt failed: %s\n",
          curl_easy_strerror(res));
  return 1;
}
```

Example program

```
//  Modified from cookie_interface.c
/***************************************************************************
 *
 *  Project                     ___| | | |  _ \| |
 *                             / __| | | | |_) | |
 *                            | (__| |_| |  _ <| |___
 *                             \___|\___/|_| \_\_____|
 *
 * Copyright (C) 1998 - 2018, Daniel Stenberg, <daniel@haxx.se>, et al.
 *
 * This software is licensed as described in the file COPYING, which
 * you should have received as part of this distribution. The terms
 * are also available at https://curl.haxx.se/docs/copyright.html.
 *
 * You may opt to use, copy, modify, merge, publish, distribute and/or sell
 * copies of the Software, and permit persons to whom the Software is
 * furnished to do so, under the terms of the COPYING file.
 *
 * This software is distributed on an "AS IS" basis, WITHOUT WARRANTY OF
ANY
 * KIND, either express or implied.
 *
 ***************************************************************************/
/* <DESC>
 * Import and export cookies with COOKIELIST.
 * </DESC>
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <time.h>
```

```
#include <curl/curl.h>

#define  VERBOSE_MODE0  "-v"
#define  VERBOSE_MODE1  "--verbose"

const int MAX_LEN = 4096;


void    pressEnter      ()
{
  printf("Press \"Enter\" to continue:");
  while  (fgetc(stdin) != '\n');
}


struct MemoryStruct {
  char *memory;
  size_t size;
};

static size_t
WriteMemoryCallback(void *contents, size_t size, size_t nmemb, void *userp)
{
  size_t realsize = size * nmemb;
  struct MemoryStruct *mem = (struct MemoryStruct *)userp;

  char *ptr = realloc(mem->memory, mem->size + realsize + 1);
  if(ptr == NULL) {
    /* out of memory! */
    printf("not enough memory (realloc returned NULL)\n");
    return 0;
  }
```

Example program, cont’d

```
  mem->memory = ptr;
  memcpy(&(mem->memory[mem->size]), contents, realsize);
  mem->size += realsize;
  mem->memory[mem->size] = 0;

  return realsize;
}

static void
print_cookies(CURL *curl)
{
  CURLcode res;
  struct curl_slist *cookies;
  struct curl_slist *nc;
  int i;
  char domain[MAX_LEN];
  char isAccessible[10];
  char path[MAX_LEN];
  char isSecure[10];
  time_t expiration;
  char name[MAX_LEN];
  char value[MAX_LEN];

  printf("Cookies, curl knows:\n");
  res = curl_easy_getinfo(curl, CURLINFO_COOKIELIST, &cookies);
  if(res != CURLE_OK) {
    fprintf(stderr, "Curl curl_easy_getinfo failed: %s\n",
            curl_easy_strerror(res));
    exit(1);
  }
  nc = cookies;
  i = 1;
  while(nc) {
    domain[0] = isAccessible[0] = path[0] = isSecure[0] =
      name[0] = value[0] = '\0';
    expiration = 0;
    sscanf(nc->data,"%s %s %s %s %ld %s %s",
           domain,isAccessible,path,isSecure,&expiration,name,value
          );
    printf("[%d]\tdomain:\t\t%s\n""\tisAccessible:\t%s\n\tpath:\t\t%s\n\t"
           "isSecure:\t%s\n\texpiration:\t%s"
           "\tname:\t\t%s\n\tvalue:\t\t%s\n\n",
           i,domain,isAccessible,path,
           isSecure,ctime(&expiration),
           name,value
          );
    nc = nc->next;
    i++;
  }
  if(i == 1) {
    printf("(none)\n");
  }
  curl_slist_free_all(cookies);
}


void    showUsage       ()
{
  fprintf(stderr,
          "Usage:\tcookie_interface <URL> [--verbose]\n"
          "Where:\n"
          "<URL>:\t\tWhich site to investigate\n"
          "--verbose:\tPrint header and other communication with server\n"
         );
}
```

Example program, cont'd

```
int main(int argc, char* argv[])
{
  CURL *curl;
  CURLcode res;

  if  (argc < 2)
  {
    showUsage();
    exit(EXIT_FAILURE);
  }

  long          verboseMode     = 0;
  const char*   urlCPtr         = NULL;

  if  ( (strcmp(argv[1],VERBOSE_MODE0) == 0)  ||
        (strcmp(argv[1],VERBOSE_MODE1) == 0)
      )
  {
    verboseMode = 1;
  }
  else
  {
    urlCPtr     = argv[1];
  }

  if  (argc >= 3)
  {

    if  ( (strcmp(argv[2],VERBOSE_MODE0) == 0)  ||
          (strcmp(argv[2],VERBOSE_MODE1) == 0)
        )
    {
      verboseMode       = 1;
    }
    else
    {

      if  (urlCPtr != NULL)
      {
        showUsage();
        exit(EXIT_FAILURE);
      }

      urlCPtr   = argv[2];
    }

  }

  if  (urlCPtr == NULL)
  {
    showUsage();
    exit(EXIT_FAILURE);
  }


  struct MemoryStruct chunk;

  chunk.memory = malloc(1);  /* will be grown as needed by the realloc above */
  chunk.size = 0;    /* no data at this point */

  curl_global_init(CURL_GLOBAL_ALL);
  curl = curl_easy_init();
  if(curl) {
    char nline[256];

    /* send all data to this function  */
    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteMemoryCallback);
```

Example program, cont’d

```
    /* we pass our 'chunk' struct to the callback function */
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, (void *)&chunk);


    curl_easy_setopt(curl, CURLOPT_URL, urlCPtr);
    curl_easy_setopt(curl, CURLOPT_VERBOSE, verboseMode);
    curl_easy_setopt(curl, CURLOPT_COOKIEFILE, ""); /* start cookie engine */
    res = curl_easy_perform(curl);
    if(res != CURLE_OK) {
      fprintf(stderr, "Curl perform failed: %s\n", curl_easy_strerror(res));
      return 1;
    }

    print_cookies(curl);
    pressEnter();
    free(chunk.memory);
    chunk.memory = malloc(1);
    chunk.size = 0;


    printf("Erasing curl's knowledge of cookies!\n");
    curl_easy_setopt(curl, CURLOPT_COOKIELIST, "ALL");

    print_cookies(curl);
    pressEnter();

    printf("----------------------------------------------\n"
           "Setting a cookie \"PREF\" via cookie interface:\n");
#ifdef WIN32
#define snprintf _snprintf
#endif
    /* Netscape format cookie */
    snprintf(nline, sizeof(nline), "%s\t%s\t%s\t%s\t%lu\t%s\t%s",
             ".example.com", "TRUE", "/", "FALSE",
             (unsigned long)time(NULL) + 31337UL,
             "PREF", "hello");
    res = curl_easy_setopt(curl, CURLOPT_COOKIELIST, nline);
    if(res != CURLE_OK) {
      fprintf(stderr, "Curl curl_easy_setopt failed: %s\n",
              curl_easy_strerror(res));
      return 1;
    }

    /* HTTP-header style cookie. If you use the Set-Cookie format and don't
    specify a domain then the cookie is sent for any domain and will not be
    modified, likely not what you intended. Starting in 7.43.0 any-domain
    cookies will not be exported either. For more information refer to the
    CURLOPT_COOKIELIST documentation.
    */
    snprintf(nline, sizeof(nline),
      "Set-Cookie: OLD_PREF=3d141414bf4209321; "
      "expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/;
domain=.example.com");
    res = curl_easy_setopt(curl, CURLOPT_COOKIELIST, nline);
    if(res != CURLE_OK) {
      fprintf(stderr, "Curl curl_easy_setopt failed: %s\n",
              curl_easy_strerror(res));
      return 1;
    }

    print_cookies(curl);
    pressEnter();
```

Example program, cont’d

```
    res = curl_easy_perform(curl);
    if(res != CURLE_OK) {
      fprintf(stderr, "Curl perform failed: %s\n", curl_easy_strerror(res));
      return 1;
    free(chunk.memory);
    }

    curl_easy_cleanup(curl);
  }
  else {
    fprintf(stderr, "Curl init failed!\n");
    return 1;
  }

  curl_global_cleanup();
  return 0;
}
```

Your Turn

Write a program that lets the user find and edit a cookie

References:

- Elliotte Rusty Harold "*Java Network Programming: 4th Ed.*"
- https://o7planning.org/en/11637/how-to-view-http-headers-in-firefox
- Rishabh Mahrsee, https://www.geeksforgeeks.org/java-net-httpcookie-java/
- Users "ETL" and "Venning", https://unix.stackexchange.com/questions/36531/format-of-cookies-when-using-wget