15-213
“The course that gives CMU its Zip!”

Web services
Nov 28, 2000

Topics
• HTTP
• Serving static content
• Serving dynamic content
Web history

1945:

  – Describes the idea of a distributed hypertext system.
  – a “memex” that mimics the “web of trails” in our minds.

1989:

• Tim Berners-Lee (CERN) writes internal proposal to develop a distributed hypertext system.
  – connects “a web of notes with links”.
  – intended to help CERN physicists in large projects share and manage information

1990:

• Tim BL writes a graphical browser for Next machines.
Web history (cont)

1992
• NCSA server released
• 26 WWW servers worldwide

1993
• Marc Andreessen releases first version of NCSA Mosaic browser
• Mosaic version released for (Windows, Mac, Unix).
• Web (port 80) traffic at 1% of NSFNET backbone traffic.
• Over 200 WWW servers worldwide.

1994
• Andreessen and colleagues leave NCSA to form "Mosaic Communications Corp" (now Netscape).
Internet Domain Survey
(www.isc.org)

Internet hosts


Mosaic and Netscape
Web servers

Clients and servers communicate using the HyperText Transfer Protocol (HTTP)

- client and server establish TCP connection
- Client requests content
- Server responds with requested content
- client and server close connection (usually)

Current version is HTTP/1.1
- RFC 2616, June, 1999.
Web server statistics

source: Netcraft Web Survey
www.netcraft.com/survey
Static and dynamic content

The content returned in HTTP responses can be either static or dynamic.

Static content:

- content stored in files and retrieved in response to an HTTP request
  - HTML files
  - images
  - audio clips

Dynamic content:

- content produced on-the-fly in response to an HTTP request
  - Example: content produced by a CGI process executed by the server on behalf of the client.
URIs and URLs

network resources are identified by Universal Resource Indicators (URIs)

The most familiar is the absolute URI known as the HTTP URL:

- \texttt{http-url} = "http:" "//" host ["":" port] [abs\_path]
- \texttt{port} defaults to "80"
- \texttt{abs\_path} defaults to "/"
- \texttt{abs\_path} ending in / defaults to …/index.html

Examples (all equivalent):

- \texttt{http://www.cs.cmu.edu:80/index.html}
- \texttt{http://www.cs.cmu.edu/index.html}
- \texttt{http://www.cs.cmu.edu}
HTTP/1.1 messages

An HTTP message is either a Request or a Response:

HTTP-message = Request | Response

Requests and responses have the same basic form:

generic-message = start-line
    *message-header
    CRLF
    [message body]

start-line = Request-line | Status line
message-header = field-name ":" [field value] CRLF
message-body = <e.g., HTML file>
HTTP/1.1 requests

Request = Method SP Request-URI SP HTTP-VERSION CRLF
*(general-header | request-header | entity header)*
CRLF
[ message-body ]

Method: tells the server what operation to perform, e.g.,
- GET: serve static or dynamic content
- POST: serve dynamic content
- OPTIONS: retrieve server and access capabilities

Request-URI: identifies the resource to manipulate
- data file (HTML), executable file (CGI)

headers: parameterize the method
- Accept-Language: en-us
- User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)

message-body: text characters
HTTP/1.1 responses

Response = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
        *(general-header | response-header | entity header) CRLF
        [ message-body ]

Status code: 3-digit number

Reason-Phrase: explanation of status code

headers: parameterize the response

• Date: Thu, 22 Jul 1999 23:42:18 GMT
• Server: Apache/1.2.5 BSDI3.0-PHP/FI-2.0
• Content-Type: text/html

message-body:

• file
How servers interpret Request-URIs

GET / HTTP/1.1
  • resolves to home/html/index.html
  • action: retrieves index.html

GET /index.html HTTP/1.1
  • resolves to home/html/index.html
  • action: retrieves index.html

GET /foo.html HTTP/1.1
  • resolves to home/html/foo.html
  • action: retrieves foo.html

GET /cgi-bin/test.pl HTTP/1.1
  • resolves to home/cgi-bin/test.pl
  • action: runs test.pl

GET http://euro.ecom.cmu.edu/index.html HTTP/1.1
  • resolves to home/html/index.html
  • action: retrieves index.html
Example HTTP/1.1 conversation

kittyhawk> telnet euro.ecom.cmu.edu 80
Connected to euro.ecom.cmu.edu.
Escape character is '
'.

Request
sent by client

| GET /test.html HTTP/1.1 ;request line |
| Host: euro.ecom.cmu.edu ;request hdr |
| CRLF |

Response
sent by server

| HTTP/1.1 200 OK ;status line |
| Date: Thu, 22 Jul 1999 03:37:04 GMT ;response hdr |
| Server: Apache/1.3.3 Ben-SSL/1.28 (Unix) |
| Last-Modified: Thu, 22 Jul 1999 03:33:21 GMT |
| ETag: "48bb2-4f-37969101" |
| Accept-Ranges: bytes |
| Content-Length: 79 |
| Content-Type: text/html |
| CRLF |
| <html> ;beginning of 79 byte message body (content) |
| <head><title>Test page</title></head> |
| <body><h1>Test page</h1> |
| </html> |
OPTIONS method

Retrieves information about the server in general or resources on that server, without actually retrieving the resource.

Request URIs:

• if request URI = “*”, then the request is about the server in general
  – Is the server up?
  – Is it HTTP/1.1 compliant?
  – What brand of server?
  – What OS is it running?

• if request URI != “*”, then the request applies to the options that available when accessing that resource:
  – what methods can the client use to access the resource?
OPTIONS (euro.ecom)

Host is a required header in HTTP/1.1 but not in HTTP/1.0

kittyhawk> telnet euro.ecom.cmu.edu 80
Trying 128.2.218.2...
Connected to euro.ecom.cmu.edu.
Escape character is '^]'.

OPTIONS * HTTP/1.1
Host: euro.ecom.cmu.edu
CRLF
HTTP/1.1 200 OK
Date: Thu, 22 Jul 1999 06:12:11 GMT
Server: Apache/1.3.3 Ben-SSL/1.28 (Unix)
Content-Length: 0
Allow: GET, HEAD, OPTIONS, TRACE
kittyhawk> telnet amazon.com 80
Trying 208.216.182.15...
Connected to amazon.com.
Escape character is '\'].

OPTIONS / HTTP/1.0
CRLF
HTTP/1.0 405 Because I felt like it.
Server: Netscape-Commerce/1.12
Date: Thursday, 22-Jul-99 04:17:32 GMT
Allow: GET, POST
Content-type: text/plain
GET method

Retrieves the information identified by the request URI.

- static content (HTML file)
- dynamic content produced by CGI program
  - passes arguments to CGI program in URI

Can also act as a conditional retrieve when certain request headers are present:

- If-Modified-Since
- If-Unmodified-Since
- If-Match
- If-None-Match
- If-Range

Conditional GETs useful for caching
GET (euro.ecom.cmu.edu)

kittyhawk> telnet euro.ecom.cmu.edu 80
Connected to euro.ecom.cmu.edu.
Escape character is '^[].

GET /test.html HTTP/1.1
Host: euro.ecom.cmu.edu
CRLF
HTTP/1.1 200 OK
Date: Thu, 22 Jul 1999 03:37:04 GMT
Server: Apache/1.3.3 Ben-SSL/1.28 (Unix)
Last-Modified: Thu, 22 Jul 1999 03:33:21 GMT
ETag: "48bb2-4f-37969101"
Accept-Ranges: bytes
Content-Length: 79
Content-Type: text/html
CRLF
<html>
<head><title>Test page</title></head>
<body><h1>Test page</h1>
</body>
</html>
GET request to euro.ecom (Internet Explorer browser)

GET /test.html HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)
Host: euro.ecom.cmu.edu
Connection: Keep-Alive
CRLF
GET response from euro.ecom

HTTP/1.1 200 OK
Date: Thu, 22 Jul 1999 04:02:15 GMT
Server: Apache/1.3.3 Ben-SSL/1.28 (Unix)
Last-Modified: Thu, 22 Jul 1999 03:33:21 GMT
ETag: "48bb2-4f-37969101"
Accept-Ranges: bytes
Content-Length: 79
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
CRLF
<html>
<head><title>Test page</title></head>
<body>
<h1>Test page</h1>
</html>
GET request to euro.ecom
(Netscape browser)

GET /test.html HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.06 [en] (Win98; I)
Host: euro.ecom.cmu.edu
Accept: image/gif, image/x-xbitmap, image/jpg, image/jpeg, image/pjpeg,
    image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
CRLF
GET response from euro.ecom

HTTP/1.1 200 OK
Date: Thu, 22 Jul 1999 06:34:42 GMT
Server: Apache/1.3.3 Ben-SSL/1.28 (Unix)
Last-Modified: Thu, 22 Jul 1999 03:33:21 GMT
ETag: "48bb2-4f-37969101"
Accept-Ranges: bytes
Content-Length: 79
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
CRLF
<html>
<head><title>Test page</title></head>
<body>
<h1>Test page</h1>
</html>
HEAD method

Returns same response header as a GET request would have...

But doesn’t actually carry out the request and returns no content
  • some servers don’t implement this properly
  • e.g., espn.com

Useful for applications that
  • check for valid and broken links in Web pages.
  • check Web pages for modifications.
HEAD (etrade.com)

kittyhawk> telnet etrade.com 80
Trying 198.93.32.75...
Connected to etrade.com.
Escape character is '^[']'.

HEAD / HTTP/1.1
Host: etrade.com
CRLF
HTTP/1.0 200 OK
Server: Netscape-Enterprise/2.01-p100
Date: Fri, 23 Jul 1999 03:18:57 GMT
RequestStartUsec: 780328
RequestStartSec: 932699937
Accept-ranges: bytes
Last-modified: Tue, 20 Jul 1999 00:59:26 GMT
Content-length: 15370
Content-type: text/html
kittyhawk> telnet espn.com 80
Trying 204.202.136.31...
Connected to espn.com.
Escape character is '^]'.

HEAD / HTTP/1.1
Host: espn.com

HTTP/1.1 301 Document Moved
Server: Microsoft-IIS/4.0
Date: Fri, 23 Jul 1999 03:22:32 GMT
Location: http://espn.go.com/
Content-Type: text/html

<html>
  Is now part of the http://espn.go.com service<br>
</html>
POST method

Another technique for producing dynamic content.
Executes program identified in request URI (the CGI program).
Passes arguments to CGI program in the message body
  • unlike GET, which passes the arguments in the URI itself.
Responds with output of the CGI program.

Advantage over GET method:
  • unlimited argument size

Disadvantages:
  • more cumbersome
  • can’t serve static content
POST request

POST /cgi-bin/post.pl HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg,
    image/pjpeg, application/vnd.ms-excel, application/msword,
    application/vnd.ms-powerpoint, */*
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows 98)
Host: kittyhawk.cmcl.cs.cmu.edu:8000
Content-Length: 25
CRLF
first=dave&last=ohallaron
POST response

HTTP/1.1 200 OK
Date: Fri, 23 Jul 1999 05:42:30 GMT
Server: Apache/1.3.4 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html
CRLF
<p>first=dave&last=ohallaron

Generated by
server

Generated by
CGI script
post.pl
TRACE, PUT, and DELETE methods

TRACE
  • Returns contents of request header in response message body.
  • HTTP’s version of an echo server.
  • Useful for debugging.

PUT:
  • add a URI to the server’s file system

DELETE
  • delete a URI from the server’s file system
Serving dynamic content

Client sends request to server.

If request URI contains the string “/cgi-bin”, then the server assumes that the request is for dynamic content.

GET /cgi-bin/env.pl HTTP/1.1
Serving dynamic content

The server creates a child process and runs the program identified by the URI in that process.
Serving dynamic content

The child runs and generates the dynamic content.

The server captures the content of the child and forwards it without modification to the client.
Serving dynamic content

The child terminates.
Server waits for the next client request.
Issues in serving dynamic content

How does the client pass program arguments to the server?

How does the server pass these arguments to the child?

How does the server pass other info relevant to the request to the child?

How does the server capture the content produced by the child?

These issues are addressed by the Common Gateway Interface (CGI) specification.
CGI

Because the children are written according to the CGI spec, they are often called CGI programs.

Because many CGI programs are written in Perl, they are often called CGI scripts.

However, CGI really defines a simple standard for transferring information between the client (browser), the server, and the child process.
add.com: 
THE Internet addition portal!

Ever need to add two numbers together and you just can’t find your calculator?

Try Dr. Dave’s addition service at add.com: THE Internet addition portal!
  • Takes as input the two numbers you want to add together.
  • Returns their sum in a tasteful personalized message.

After the IPO we’ll expand to multiplication!
The add.com experience

Welcome to add.com: THE Internet addition portal.

The answer is: 1 + 5 = 6

Thanks for visiting!
Serving dynamic content with GET

Question: How does the client pass arguments to the server?
Answer: The arguments are appended to the URI

Can be encoded directly in a URL typed to a browser or a URL in an HTML link
  - http://add.com/cgi-bin/adder?1&2
  - adder is the CGI program on the server that will do the addition.
  - argument list starts with “?”
  - arguments separated by “&”
  - spaces represented by “+” or “%20”

Can also be generated by an HTML form

<form method=get action="http://add.com/cgi-bin/postadder">
Serving dynamic content with GET

URL:
- http://add.com/cgi-bin/adder?1&2

Result displayed on browser:

Welcome to add.com: THE Internet addition portal.

The answer is: 1 + 2 = 3

Thanks for visiting! Tell your friends.
Serving dynamic content with GET

**Question**: How does the server pass these arguments to the child?

**Answer**: In environment variable `QUERY_STRING`
- a single string containing everything after the “?”
- for add.com: `QUERY_STRING = “1&2”`

```c
/* child code that accesses the argument list */
if ((buf = getenv("QUERY_STRING")) == NULL) {
    exit(1);
}

/* extract arg1 and arg2 from buf and convert */
...  
n1 = atoi(arg1);
n2 = atoi(arg2);
```
Serving dynamic content with GET

**Question:** How does the server pass other info relevant to the request to the child?

**Answer:** in a collection of environment variables defined by the CGI spec.
Some CGI environment variables

General

- SERVER_SOFTWARE
- SERVER_NAME
- GATEWAY_INTERFACE (CGI version)

Request-specific

- SERVER_PORT
- REQUEST_METHOD (GET, POST, etc)
- QUERY_STRING (contains GET args)
- REMOTE_HOST (domain name of client)
- REMOTE_ADDR (IP address of client)
- CONTENT_TYPE (for POST, type of data in message body, e.g., text/html)
- CONTENT_LENGTH (length in bytes)
Some CGI environment variables

In addition, the value of each header of type type received from the client is placed in environment variable HTTP_type

• Examples:
  – HTTP_ACCEPT
  – HTTP_HOST
  – HTTP_USER_AGENT (any “-” is changed to “_”)

Serving dynamic content with GET

**Question:** How does the server capture the content produced by the child?

**Answer:** The child writes its headers and content to stdout.

- Server maps socket descriptor to stdout (more on this later).
- Notice that only the child knows the type and size of the content. Thus the child (not the server) must generate the corresponding headers.

```c
/* child generates the result string */
sprintf(content, "Welcome to add.com: THE Internet addition portal\n    <p>The answer is: %d + %d = %d\n    <p>Thanks for visiting!\n", n1, n2, n1+n2);

/* child generates the headers and dynamic content */
printf("Content-length: %d\n", strlen(content));
printf("Content-type: text/html\n");
printf("\r\n");
printf("%s", content);
```
Serving dynamic content with GET

bass> tiny 8000
GET /cgi-bin/adder?1&2 HTTP/1.1
Host: bass.cmcl.cs.cmu.edu:8000
<CRLF>

kittyhawk> telnet bass 8000
Trying 128.2.222.85...
Connected to BASS.CMCL.CS.CMU.EDU.
Escape character is '^]'.
GET /cgi-bin/adder?1&2 HTTP/1.1
Host: bass.cmcl.cs.cmu.edu:8000
<CRLF>
HTTP/1.1 200 OK
Server: Tiny Web Server
Content-length: 102
Content-type: text/html
<CRLF>
Welcome to add.com: THE Internet addition portal.
<p>The answer is: 1 + 2 = 3
<p>Thanks for visiting!
Connection closed by foreign host.
kittyhawk>
class26.ppt

HTTP request received by server

HTTP request sent by client

HTTP response generated by the server

HTTP response generated by the CGI program