HTTP, HTTPS and TCP Networking in Java

Some reminders

Credits

Some material derived from:

- HTTP vs. HTTPS by Eng. T. Aldaldooh

RFC

Request for Comments (RFC) are a type of publication from the Internet Engineering Task Force (IETF) and the Internet Society (ISOC), the principal technical development and standards-setting bodies for the Internet.

- An RFC is authored by engineers and computer scientists in the form of a memorandum describing methods, behaviors, research, or innovations applicable to the working of the Internet and Internet-connected systems. I

Protocol

- Synonymous of **Etiquette**

a code of behavior that delineates expectations for social behavior according to contemporary conventional norms within a society, social class, or group.

Communications protocol, a set of rules and regulations that determine how data is transmitted in telecommunications and computer networking

Port

HTTP on port 80
- HTTP with SSL (HTTPS) on port 443
- FTP on port 21
- SMTP on port 25
- POP on port 110
- SSH on port 22

A port is an **endpoint of communication in an operating system**.

A **process** associates its input or output channels, via an Internet socket, with a transport protocol, a port number, and an IP address.

This process is known as binding,

PID	PORT	IP	Protocol
84	21	193.205.196.130	FTP
78	80	193.205.196.130	HTTP
321	8080	193.205.196.130	HTTP
541	25	193.205.196.130	SMTP

Mistranslated into Italian as “Porta” (door)

URL and URI

- URLs used early on by all Internet protocols, including various document retrieval protocols.
- More specifications (both from 1994):
  - URL : Uniform Resource Locators - RFC 1738.
  - URI : Universal Resource Identifiers - RFC 1630.

URL is just one type of a URI.

URLS and URIS

- URL (Uniform Resource Locators)
  - Provides single short string to identify network-accessible resource
  - <scheme>://<host>[:<port>]/<path>[?<query>]
  - http://www.w3.org/Icons/w3c_home.gif
- URI (Uniform Resource Identifier)
  - Identifies a resource either by location or name.
  - The selection of the representation can be determined by the web server through HTTP content negotiation.
  - A superset of URLs
  - http://www.w3.org/Icons/w3c_home.
  - http request line contains a non-URL URI

URL, URN, URC

- **URL**: identify resources by specifying their locations in the context of a particular access protocol, such as HTTP or FTP.
- **URN**: persistent, location-independent identifiers
- **URC**: standardized representation of document properties, such as owner, encoding, access restrictions or cost.

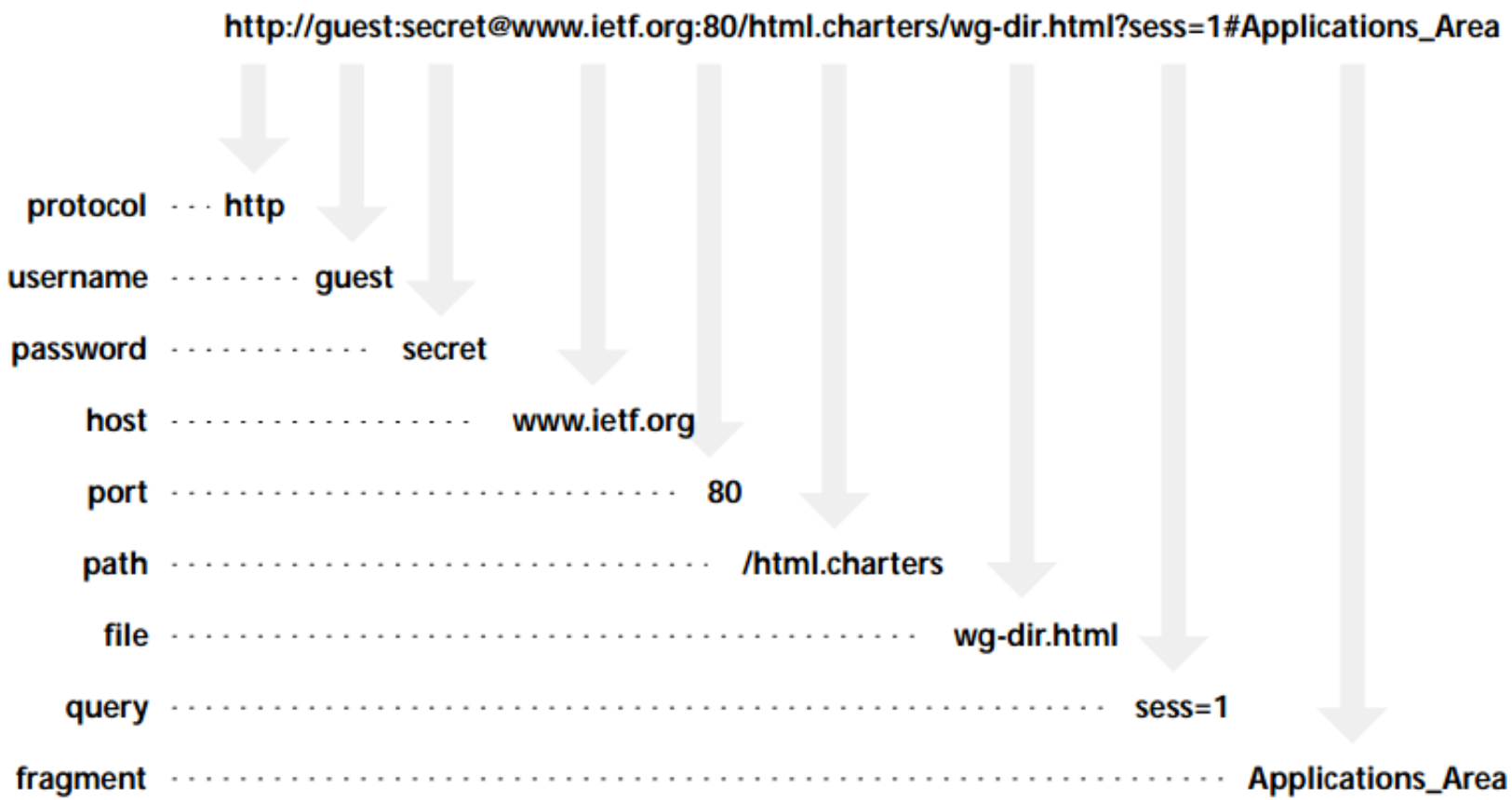
URN

- URN are not locators, are not required to be associated with a particular protocol or access method, and need not be resolvable.

- They should be assigned by a procedure which provides some assurance that they will **remain unique** and **identify the same resource** persistently over a prolonged period.

- A typical URN namespace is urn:isbn, for International Standard Book Numbers.

URLs cont.


http://guest:secret@www.ietf.org:80/html.charters/wg-dir.html?sess=1#Applications_Area
protocol
http
username
guest
password
secret
host
www.ietf.org
port
80
path
/html.charters
file
wg-dir.html
query
sess=1
fragment
Applications_Area

URLs cont.

- **Protocol**: Identifies the application protocol needed to access the resource, in this case HTTP.
- **Username** : If the protocol supports the concept of user names, this provides a user name that has access to the resource; in the example "guest."
- **Password**: The password associated with the user name, "secret" in the example.
- **Host** : The communication system that has the resource; for HTTP this is the Web server, www.ietf.org in the example.
- **Port** : The TCP port that the application protocols should use to access the resource; many protocols have an implied TCP port (for HTTP that port is 80
- **Path** : The path through a hierarchical organization under which the resource is located, often a file system's directory structure or equivalent.
- **File**: The resource itself.
- **Query**: Additional information about the resource or the client.
- **Fragment**: A particular location within a resource.

URL and MIME type

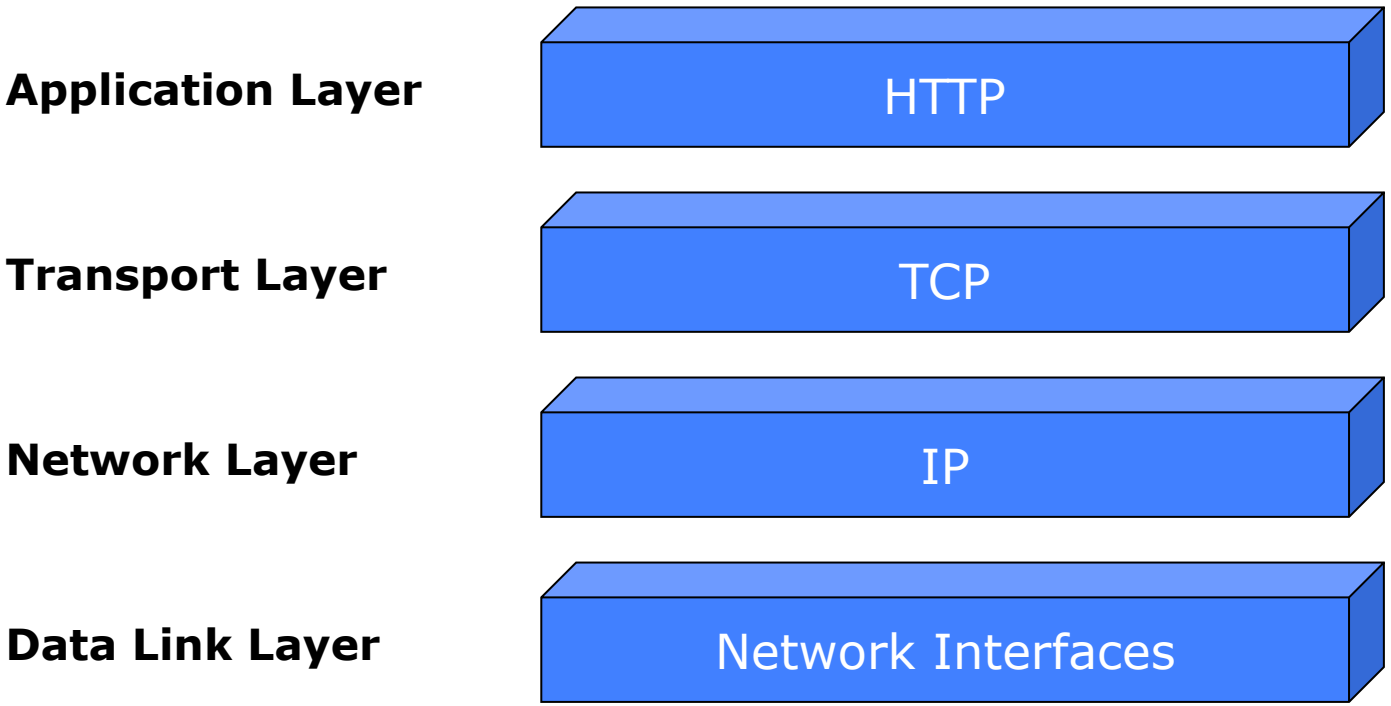
- **URLs point to resources ("content").**
- Resources are represented using different **Internet Media Types** (MIME Types)
  - Multipurpose Internet Mail Extensions RFC 2045,6
- MIME Type tells how content should be handled
  - File extensions are mapped to certain MIME Types
    - .html usually means a MIME Type of text/html
    - .jpg usually means a MIME Type of image/jpeg
- The most common MIME Types used on the Web come from the text, image and application top-level groups
    - text/html, text/css
    - image/gif, image/jpeg, image/png
    - application/pdf, application/octet-stream
    - application/x-javascript, application/x-shockwave-flash

An Introduction to HTTP

- Hyper Text Transfer Protocol
- One of the application layer protocols that make up the Internet
    - HTTP over TCP/IP
    - Like SMTP, POP, IMAP, NNTP, FTP, etc.
- The underlying language of the Web
- Three versions have been used, two are in common use and have been specified:
    - RFC 1945 HTTP 1.0 (1996)
    - RFC 2616 HTTP 1.1 (1999)

HTTP and TCP/IP

HTTP sits atop the TCP/IP Protocol Stack



HTTP

- The Hypertext Transfer Protocol
- distributed, collaborative, hypermedia information systems.
- HTTP functions as a request-response protocol in the client-server computing model.
- Actors:
  - Internet Engineering Task Force (IETF)
  - World Wide Web Consortium (W3C)

RFC

RFC 2616 (June 1999) defined HTTP/1.1

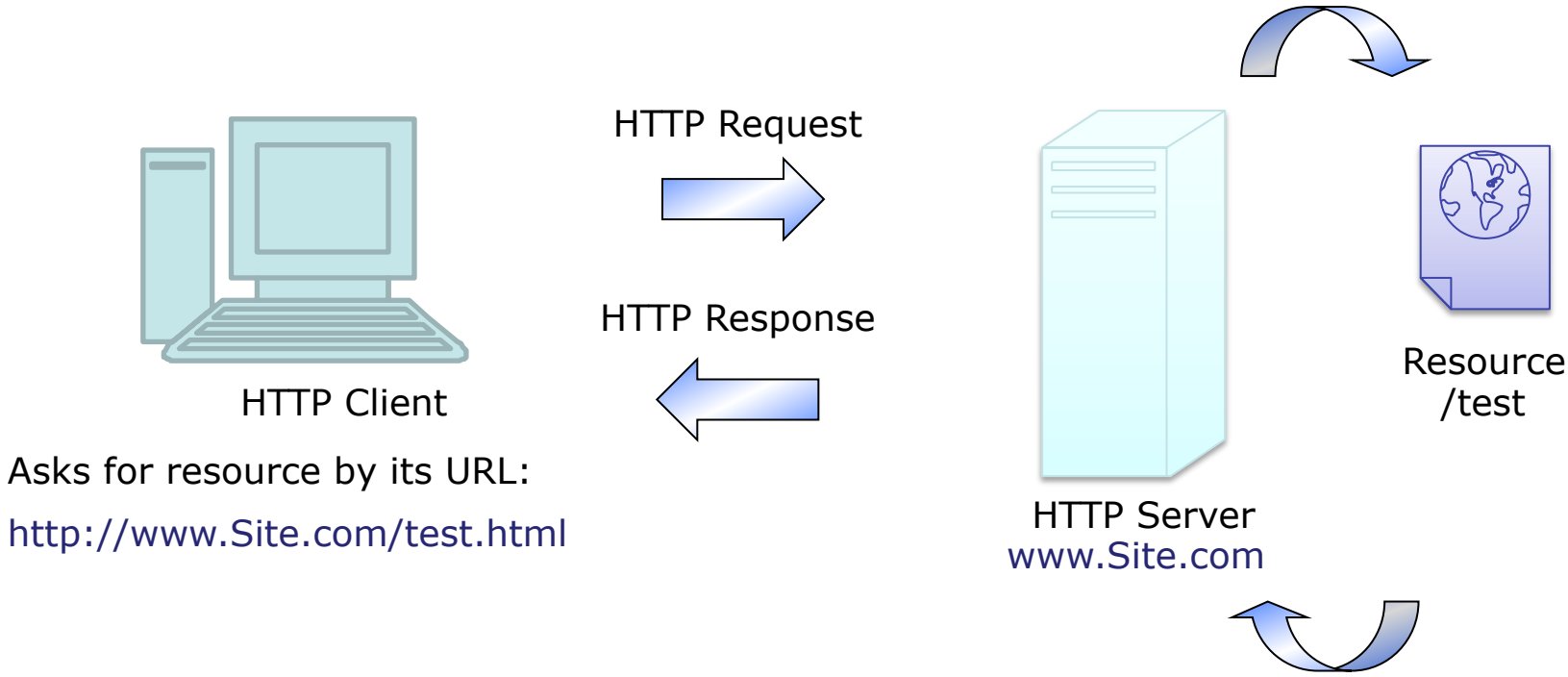
In June 2014, RFC 2616 was retired and HTTP/1.1 was redefined by

- RFC 7230 - HTTP/1.1: Message Syntax and Routing
- RFC 7231 - HTTP/1.1: Semantics and Content
- RFC 7232 - HTTP/1.1: Conditional Requests
- RFC 7233 - HTTP/1.1: Range Requests
- RFC 7234 - HTTP/1.1: Caching
- RFC 7235 - HTTP/1.1: Authentication

HTTP 2.0 Current Status

- May 2015 RFC 7540
- May 2015 RFC 7541 (HPACK)

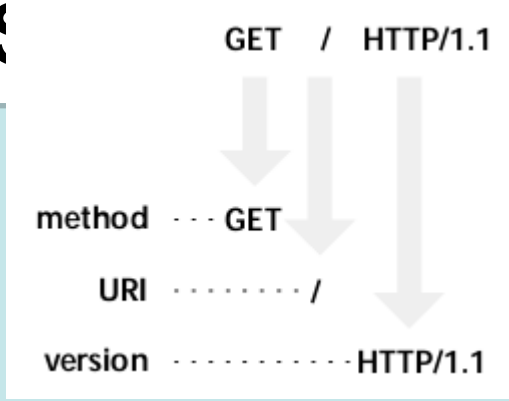
HTTP servers turn URLs into resources through a request-response cycle



HTTP requests and responses Messages

- HTTP requests and responses are both types of Internet Messages (RFC 822), and share a general format:
  - **A Start Line, followed by a CRLF**
    - Request Line for requests
    - Status Line for responses
  - **Zero or more Message Headers**
    - field-name ":" [field-value] CRLF
  - **An empty line**
    - Two CRLFs mark the end of the Headers
  - **An optional Message Body if there is a payload**
    - All or part of the "Entity Body" or "Entity"

HTTP Request



GET / HTTP/1.1[CRLF]
Host: www.iugaza.edu.ps[CRLF]
Connection: close[CRLF]
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)[CRLF]
Accept-Encoding: gzip[CRLF]
Accept-Charset: ISO-8859-1,UTF-8;q=0.7,*;q=0.7[CRLF]
Cache-Control: no-cache[CRLF]
Accept-Language: de,en;q=0.7,en-us;q=0.3[CRLF]
Referer: http://web-sniffer.net/[CRLF] [CRLF]

A Closer Look at the Request Methods

- **GET**
  - By far most common method
  - Retrieves a resource from the server
  - Supports passing of query string arguments
- **HEAD**
  - Retrieves only the Headers associated with a resource but not the entity itself
  - Highly useful for protocol analysis, diagnostics
- **POST**
  - Allows passing of data in entity rather than URL
  - Can transmit of far larger arguments that GET
  - Arguments not displayed on the URL

More Request Methods, cont.

- **OPTIONS**
  - Shows methods available for use on the resource (if given a path) or the host (if given a “*”)
- **TRACE**
  - Diagnostic method for assessing the impact of proxies along the request-response chain
- **PUT, DELETE**
  - Used in HTTP publishing (e.g., WebDav)
- **CONNECT**
  - A common extension method for Tunneling other protocols through HTTP

Web-based Distributed Authoring and Versioning (WebDAV) is a set of methods based on the Hypertext Transfer Protocol (HTTP) that facilitates collaboration between users in editing and managing documents and files stored on World Wide Web servers.

HTTP Responses

http://web-sniffer.net/

HTTP Response Header

Name	Value
Status: HTTP/1.1 200 OK	
Cache-Control:	public, max-age=1
Content-Type:	text/html; charset=utf-8
Expires:	Sat, 24 Dec 2011 19:04:54 GMT
Last-Modified:	Sat, 24 Dec 2011 19:04:24 GMT
Server:	Microsoft-IIS/7.0
X-AspNet-Version:	2.0.50727
X-Powered-By:	ASP.NET
Date:	Sat, 24 Dec 2011 19:04:52 GMT
Connection:	close
Content-Length:	70241

Content (50.58 KiB)

```
[CRLF]
<html xmlns="http://www.w3.org/1999/xhtml" dir="rtl">[CRLF]
<head>[CRLF]
[CRLF]
<META name="y_key" content="5c35482f6a363179" >[CRLF]
<meta http-equiv="Content-Type" content="text/html; charset=windows-1256" />[CRLF]
<meta name="description" content="Islamic University Of Gaza - Palestine , Gaza" />[CRLF]
<meta name="keywords" content="Islamic University Of Gaza - Palestine , Gaza - palestine, Gaza, universit
<meta name="subject" content="Islamic University Of Gaza - Palestine" />[CRLF]
[CRLF]
[CRLF]
<title>غزة - الاسلامية الجامعة</title>[CRLF]
```

A Closer Look at the Status Line

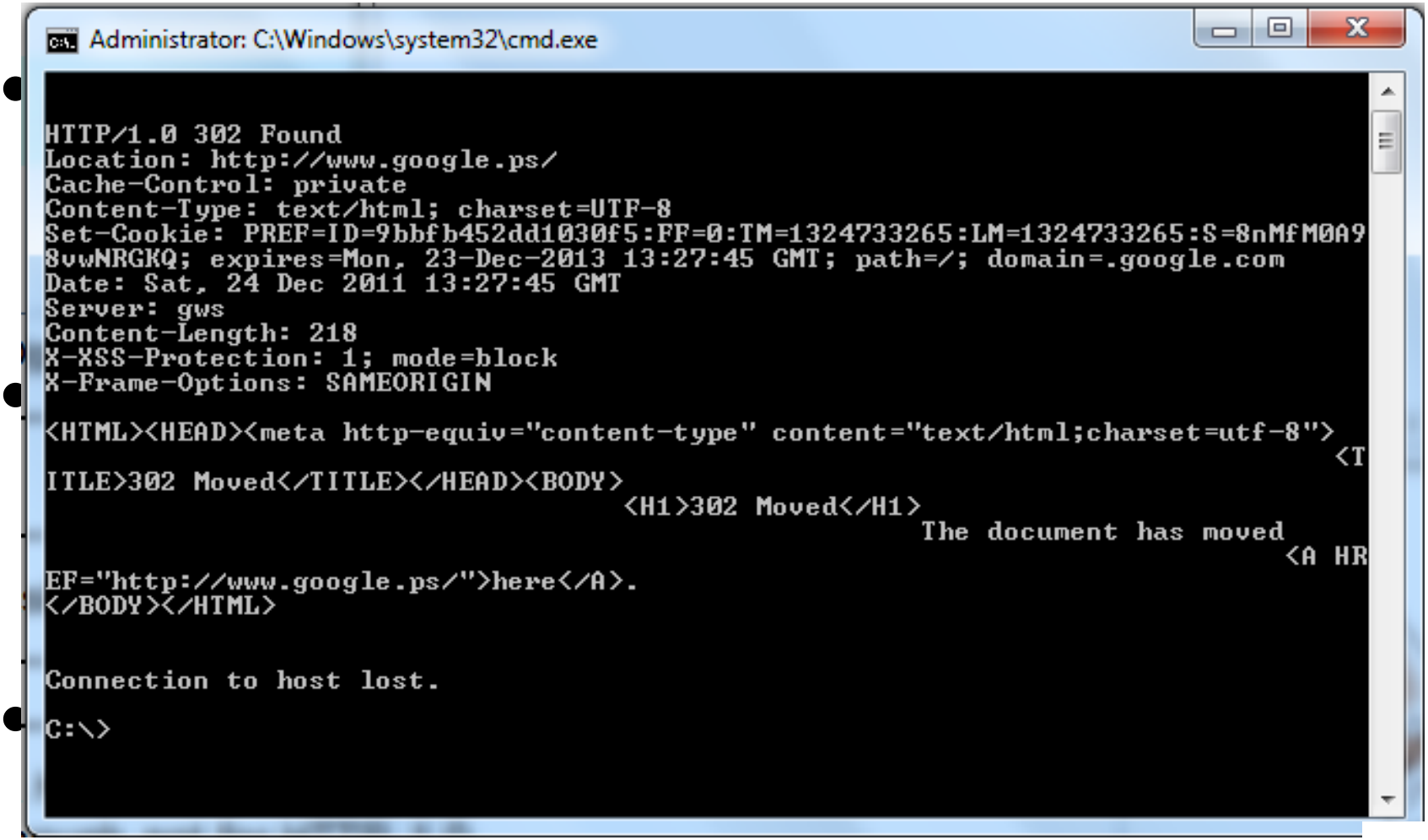
- **Consists of three major parts:**
- The HTTP Version
  - Just like third part of Request Line
- Status Code
  - **5 groups of 3 digit integers indicating the result of the attempt to satisfy the request:**
  - 1xx are informational
  - 2xx are success codes
  - 3xx are for alternate resource locations (redirects)
  - 4xx indicate client side errors
  - 5xx indicate server side errors
- The Reason Phrase followed by the CRLF
  - Short textual description of the status code

A Closer Look at the Status Line

Table 3.2 HTTP Status Code Categories

Status Code	Meaning
100-199	Informational; the server received the request but a final result is not yet available.
200-299	Success; the server was able to act on the request successfully.
300-399	Redirection; the client should redirect the request to a different server or resource.
400-499	Client error; the request contained an error that prevented the server from acting on it successfully.
500-599	Server error; the server failed to act on a request even though the request appears to be valid.

Making a simple HTTP request using Telnet



A Closer Look at HTTP Headers

Headers come in four major types, some for requests, some for responses, some for both:

- **General Headers**
  - Provide info about messages of both kinds
- **Request Headers**
  - Provide request-specific info
- **Response Headers**
  - Provide response-specific info
- **Entity Headers**
  - Provide info about request and response entities
- Extension headers are also possible

General Headers

- **Connection** – lets clients and servers manage connection state
  - Connection: Keep-Alive
  - Connection: close
- **Date** – when the message was created
  - Date: Sat, 31-May-03 15:00:00 GMT
- **Via** – shows proxies that handled message
  - Via: 1.1 www.myproxy.com (Squid/1.4)
- **Cache-Control** – Among the most complex of headers, enables caching directives
  - Cache-Control: no-cache

Request Headers

- **Host** – The hostname (and optionally port) of server to which request is being sent
- **Referer** – The URL of the resource from which the current request URI came
  - Referer: http://www.host.com/login.asp
- **User-Agent** – Name of the requesting application, used in browser sensing
  - User-Agent: Mozilla/4.0 (Compatible; MSIE 6.0)
- **Accept** and its variants – Inform servers of client's capabilities and preferences
  - Enables content negotiation
  - Accept: image/gif, image/jpeg;q=0.5
  - Accept- variants for Language, Encoding, Charset
- **Cookie** How clients pass cookies back to the servers that set them
  - Cookie: id=23432;level=3

Response Headers

- **Server** – The server's name and version
  - Server: Microsoft-IIS/5.0
  - Can be problematic for security reasons

- **Set-Cookie** – This is how a server sets a cookie on a client
  - Set-Cookie: id=234; path=/shop; expires=Sat, 31-May-03 15:00:00 GMT; secure

Entity Headers

- **Allow** – Lists the request methods that can be used on the entity
  - Allow: GET, HEAD, POST
- **Location** – Gives the alternate or new location of the entity
  - Used with 3xx response codes (redirects)
  - Location: http://www.iugaza.edu.ps/ar/
- **Content-Encoding** – specifies encoding performed on the body of the response
  - Used with HTTP compression
  - Corresponds to Accept-Encoding request header
  - Content-Encoding: gzip
- **Content-Length** – The size of the entity body in bytes
- **Content-Location** – The actual if different than its request URL
- **Content-Type** – specifies Media (MIME) type of the entity body

HTTP Overview

HTTP Requests

An HTTP request consists of
a request method, (“subprotocol” specification)
a request URL, (location)
header fields, (metadata)
a body. (data)

HTTP 1.1 defines the following request methods:

- GET: Retrieves the resource identified by the request URL
- HEAD: Returns the headers identified by the request URL
- POST: Sends data of unlimited length to the Web server
- PUT: Stores a resource under the request URL
- DELETE: Removes the resource identified by the request URL
- OPTIONS: Returns the HTTP methods the server supports
- TRACE: Returns the header fields sent with the TRACE request
- CONNECT request connection to a transparent TCP/IP tunnel,
- PATCH apply partial modifications to a resource.

HTTP 1.0 includes only the GET, HEAD, and POST methods.