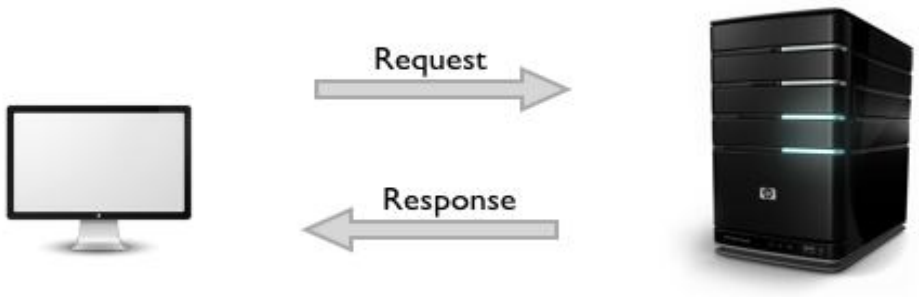
HTTP PROTOCOL

HTTP Request and Response



HTTP Request and Response

HTTP (Hypertext Transfer Protocol)

- generic, stateless protocol
- governs the transfer of files across a network

HTTP Request and Response

HTTP (Hypertext Transfer Protocol)

- supports access to SMTP,FTP and other protocols
- was designed to support hypertext

HTTP Request and Response

HTTP (Hypertext Transfer Protocol)

- Exchanged information can be static or dynamic
- Every resource, accessible over the Web has a URL(Uniform resource locator)
- HTTP mechanism is based on client/server model typically using TCP/IP sockets

HTTP Request and Response

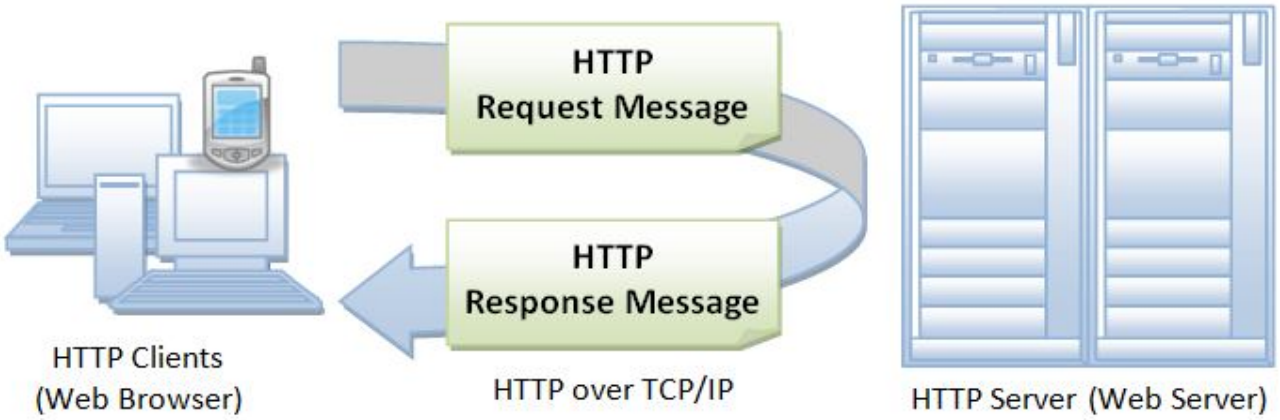


Figure illustrates request-response client-server protocol

HTTP Request

Request = Request-Line
***((general-header**
| request-header
| entity-header) CRLF)
CRLF
[message-body]

Example :

URL: http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

Request Headers | Request Body | Response Headers | Response Body | Cookies | Initiator | Timings

Key	Value
Request	GET /Protocols/rfc2616/rfc2616-sec14.html HTTP/1.1
Accept	text/html, application/xhtml+xml, */*
Referer	http://www.google.com/url?sa=t&source=web&cd=3&ved=0CC4QFjAC8
Accept-Language	en-US
User-Agent	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5
Accept-Encoding	gzip, deflate
Host	www.w3.org
If-Modified-Since	Wed, 01 Sep 2004 13:24:52 GMT
If-None-Match	"1edec-3e3073913b100"
Connection	Keep-Alive

Source : http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5

HTTP Request

- Request-Line
  - begins with a method token,
  - followed by the Request-URI & protocol version & ending with CRLF
  - elements are separated by SP characters
  - No CR or LF is allowed except in the final CRLF sequence

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Request-Line

Method

```
Method =          "OPTIONS"
                  | "GET"
                  | "HEAD"
                  | "POST"
                  | "PUT"
                  | "DELETE"
                  | "TRACE"
                  | "CONNECT"
                  | extension-method
extension-method = token
```

Request-Line

Request-URI

Request-URI = "*" | absoluteURI | abs_path | authority

HTTP Request

General header

```
general-header = Cache-Control
                 | Connection
                 | Date
                 | Pragma
                 | Trailer
                 | Transfer-Encoding
                 | Upgrade
                 | Via
                 | Warning
```

HTTP Request

Request header

```
request-header =    Accept
                  | Accept-Charset
                  | Accept-Encoding
                  | Accept-Language
                  | Authorization
                  | Expect
                  | From
                  | Host
                  | If-Match
                  | If-Modified-Since
                  | If-None-Match
                  | If-Range
                  | If-Unmodified-Since
                  | Max-Forwards
                  | Proxy-Authorization
                  | Range
                  | Referer
                  | TE
                  | User-Agent
```

HTTP Request

Entity header

```
entity-header  = Allow
               | Content-Encoding
               | Content-Language
               | Content-Length
               | Content-Location
               | Content-MD5
               | Content-Range
               | Content-Type
               | Expires
               | Last-Modified
               | extension-header
      extension-header = message-header
```

HTTP Request

Message body

```
message-body = entity-body
             | <entity-body encoded as per Transfer-Encoding>
```

HTTP Response

```
Response      = Status-Line
                *(( general-header
                 | response-header
                 | entity-header ) CRLF)
                CRLF
                [ message-body ]
```

Example :

URL: http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

Request Headers | Request Body | Response Headers | Response Body | Cookies | Initiator | Timings

Key	Value
Response	HTTP/1.1 200 OK
Date	Sun, 05 Dec 2010 01:27:11 GMT
Server	Apache/2
Last-Modified	Wed, 01 Sep 2004 13:24:52 GMT
ETag	"1edec-3e3073913b100"
Accept-Ranges	bytes
Content-Length	126444
Cache-Control	max-age=21600
Expires	Sun, 05 Dec 2010 07:27:11 GMT
P3P	policyref="http://www.w3.org/2001/05/P3P/p3p.xml"
Connection	close
Content-Type	text/html; charset=iso-8859-1

HTTP Response

Status Line

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

Status Code and Reason Phrase

- 1xx: Informational - Request received, continuing process –
- 2xx: Success - The action was successfully received, understood, and accepted
- 3xx: Redirection - Further action must be taken in order to complete the request
- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
- 5xx: Server Error - The server failed to fulfill an apparently valid request

For a detailed list of status codes refer : http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6

Source : http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6

HTTP Response

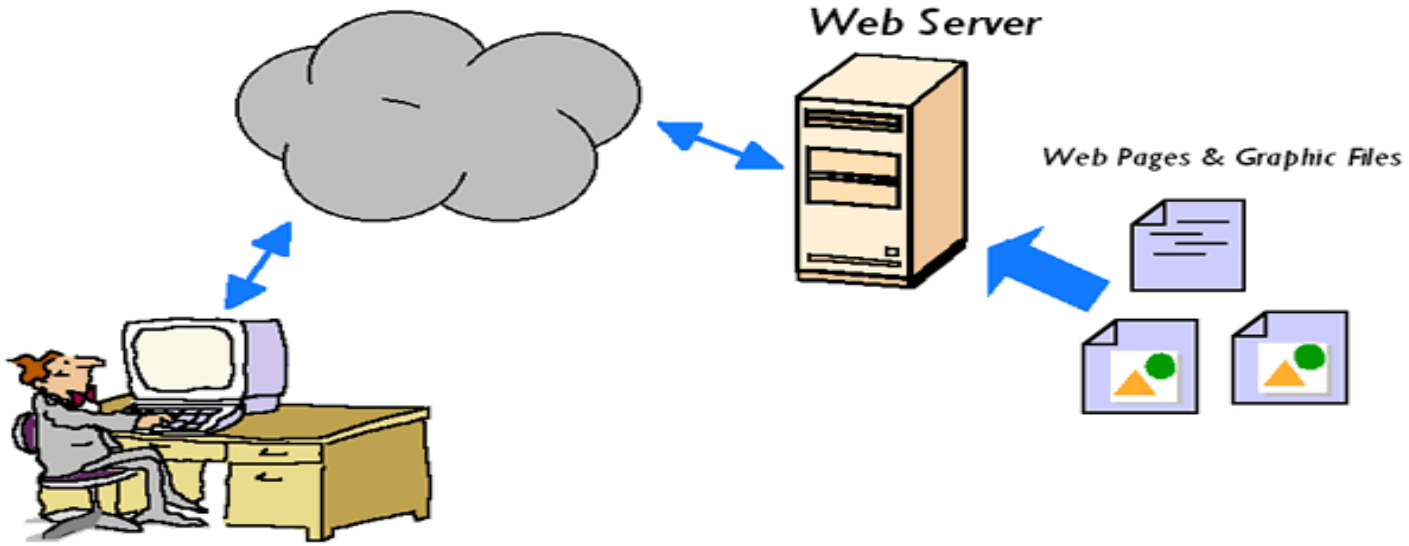
Response Header

```
response-header = Accept-Ranges
                | Age
                | ETag
                | Location
                | Proxy-Authenticate
                | Retry-After
                | Server
                | Vary
                | WWW-Authenticate
```

Source : http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6

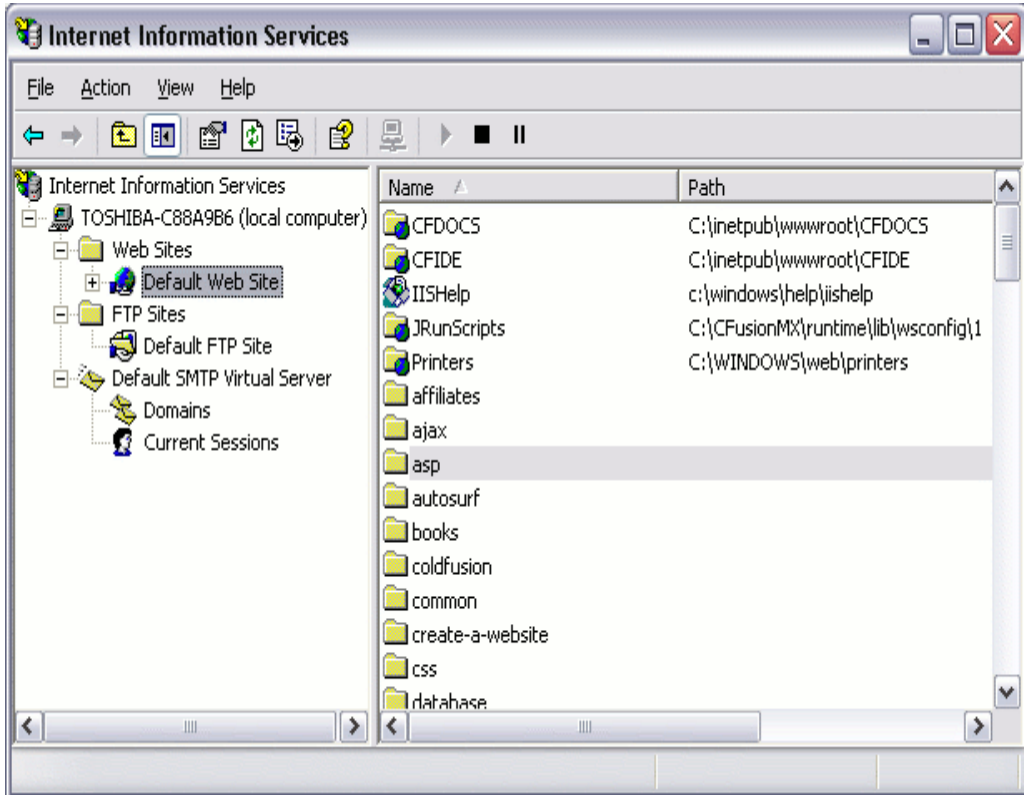
Web Server

- a piece of software that enables a website to be viewed using HTTP
- Server(computer) that serves Web pages
- Has an IP address & domain name



Web Server

- Example Web server – IIS 5.1 (Microsoft Internet Information Services)
- left pane represents the various websites, FTP sites, and SMTP virtual servers



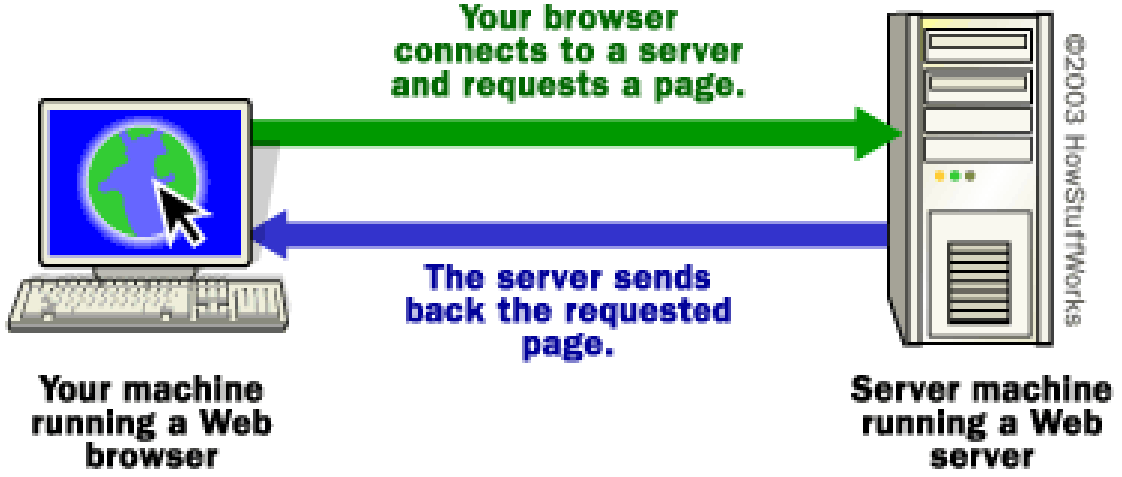
Screenshot has 1 Website,1 FTP site and one SMTP virtual server

Web Server

- Some common features
  - Create one or more websites
  - Configure log file settings
  - Configure website/directory security
  - Create an FTP site

Web Server

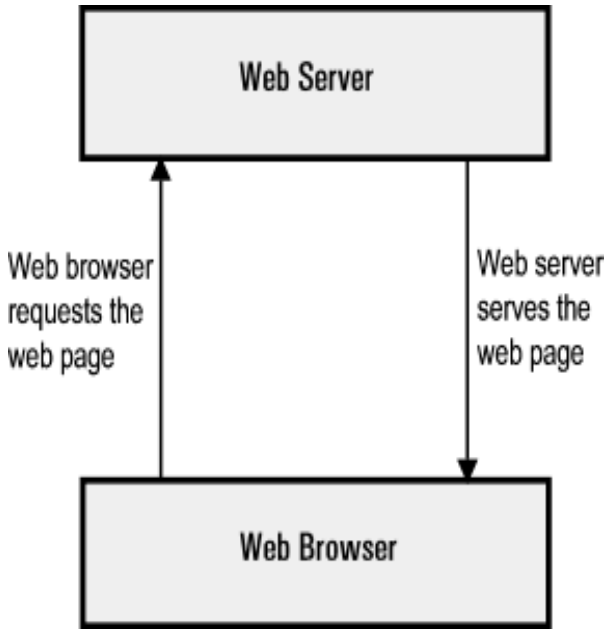
- Some common features
  - Create virtual directories, and map them to physical directories
  - Configure/nominate custom error pages
  - Specify default documents



Web Server – How it works

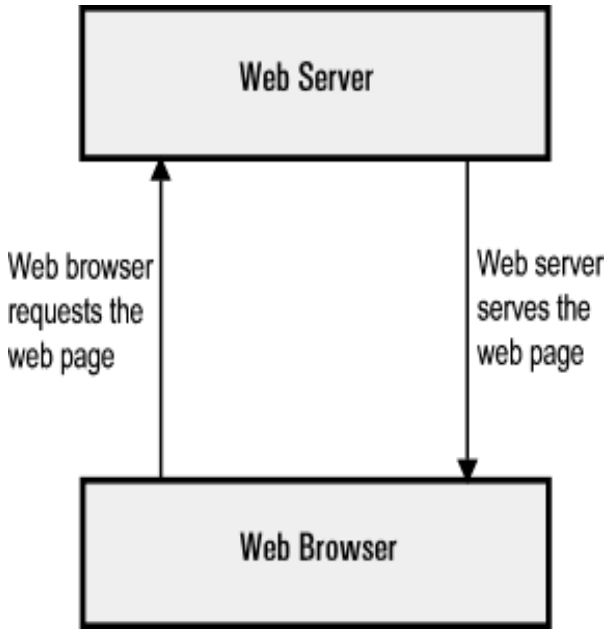
- Step 1 : Web browser gets IP address of website from cache or requests it from DNS server

- Step 2 : Web browser requests full URL from the Web server



Web Server – How it works

- Step 3 : Web server sends back requested page. Error page is sent if requested page is not found

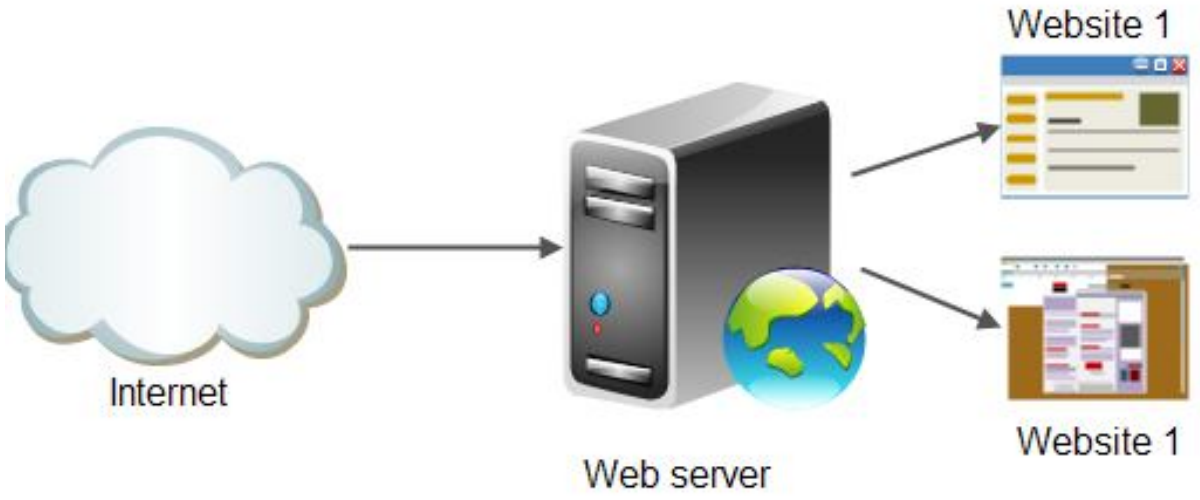


- Step 4 : Browser receives page and renders it

Web Server

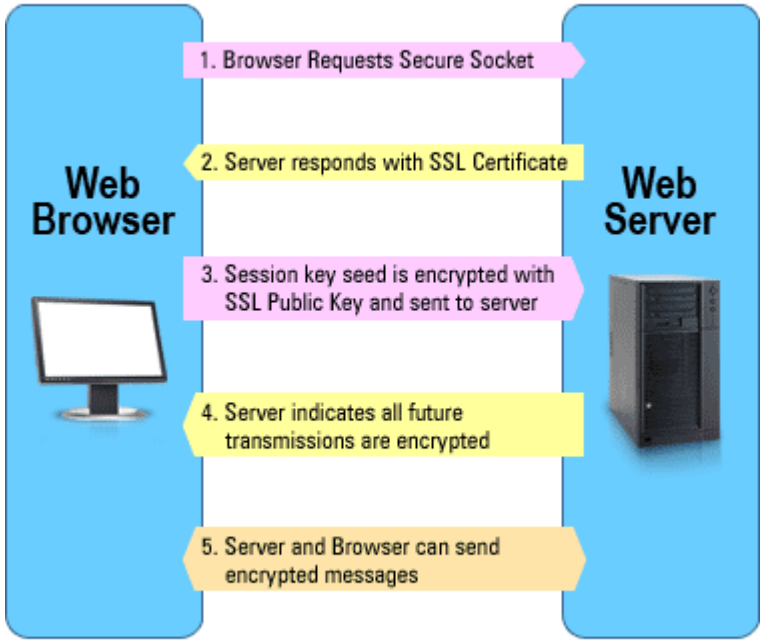
- Multiple Websites

  One web server could be configured to host multiple sites with unique IP address



Web Server

- SSL (Secure Socket Layer) Certificates
  - SSL can be applied to the website using a web server
  - Navigation to the website thus becomes secure
  - Uses https:// instead of http://



Web server Examples

- Apache HTTP Server - http://httpd.apache.org
- Apache Tomcat - http://tomcat.apache.org/
- Microsoft Internet Information Services (IIS) - http://www.microsoft.com/iis