An Internet application is an interactive, compiled application that can be accessed through a communication line. Internet applications can perform complex business processes on either the client or the server. In a server-based Internet application, the application uses the Internet protocol to receive requests from a client, typically a Web browser, process associated code, and return data to the browser.

Tier Technology: Definition

- Tier: refers to the layer.
- Tier Technology refers to the Network application architecture.
- The concept of tier provides a convenient way to group different classes of architecture.

Types of Tier Technology:

- There exists 4 different types of tier technology:

1) 1-Tier Technology
2) 2-Tier Technology
3) 3-Tier Technology
4) N-Tier Technology

1-Tier Technology:

- All software contains codes that can be broken down into the following categories:
  - **Presentation Logic:** User Interface, displaying data to the user, accepting input from the user.
  - **Business Logic:** Data validation, ensuring the data is correct before being added into database.
  - **Data Access Logic:** Database communication, accessing tables and indices, packing and unpacking data
- If the 3 categories of logic are contained in a single component within a single computer then the component is said to be a 1-tier Structure.

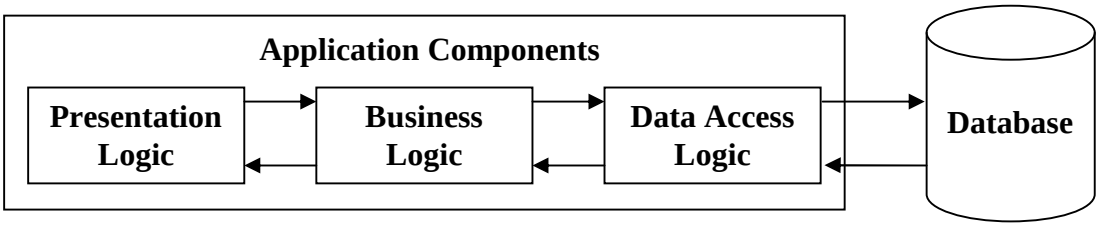


Fig: *1- Tier Technology*

2-Tier Technology:

- This technology is also known as Client/Server Technology.
- This consists of a primary tier which incorporates all presentation and business logic, and a secondary tier which contains all data access logic.
- is basically a client server model.
- There are basically two types of client:

a. Thin Client:

- The clients do not have too much work as most of the functionality is hosted on the server.
- Thin Clients were due to the fact that computing power was expensive.

b. Fat Client:

- A name given such because some of the processing has been passed onto it.
- Typical Example – if such a client were accessing a database, the client application would need to be capable of accessing the database and manipulating the data which resides on the server.

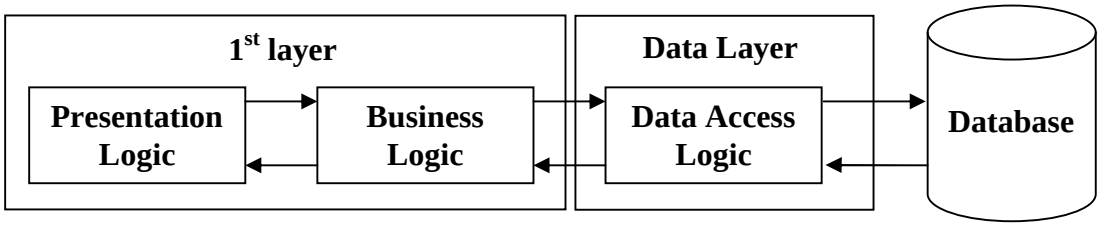


Fig: 2- *Tier Technology*

3-Tier Technology:

- If each category of logic is contained within a separate component, known as 3-tier structure.
- In this structure the presentation layer is not in direct communication with the database.
- In order to send or receive data it must communicate with the business layer, which in turn communicates with the data layer.
- Contains 3 physical tiers: client tier, business or middle tier, data tier.

a. Client Tier:

- Front –End tools with which the end user interacts.
- Not concerned with inner workings of the applications.
- E.g: Web Browser, or VB Forms.

b. Middle Tier:

- This is where the ASP pages resides upon the web-server.
- Contains rules that determine what, how & when to manipulate & access data.

- This logic is splitted into:

I. Client Side Business Logic or workflow

- Controls the input given from user.

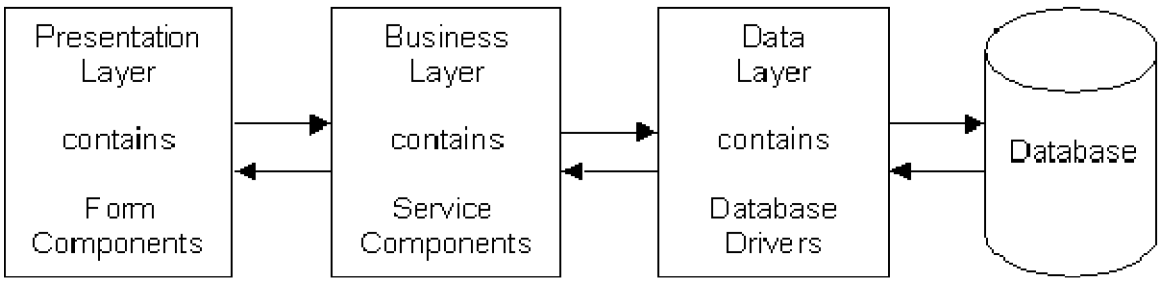
II. Server Side Business Logic

- Handles data manipulation & flow of data on server.

c. Data Tier

- Represents the storage mechanism used to hold persistent data.
- This could be a relational database, text based files, directories, mail server etc.

Fig: *3 – Tier Technology*



N-Tier Technology:

- N-Tier technology is the latest refinement of client/Server model.
- It refers to applications that have more than three tiers.
- An n-tier system is scalable because it can service a large number of clients by distributing each logical tier across one or more physical machines.
- is usually considered the most effective approach because it can provide integration of current information technology into this new, more flexible model.
- In this structure middle tier is splitted to incorporate user-centric tier and data-centric tier.

a. User-centric Tier:

- User-centric tier of an ASP applications contains the ASP pages and environment-dependent ASP components that help render HTML pages to the presentation layer. E.g. – generating HTML, accessing and creating cookies.

b. Data-centric Tier–

- Data-centric tier contains the components that do not depend upon the ASP environment but is responsible for database manipulation, such as adding, deleting, querying and updating etc.

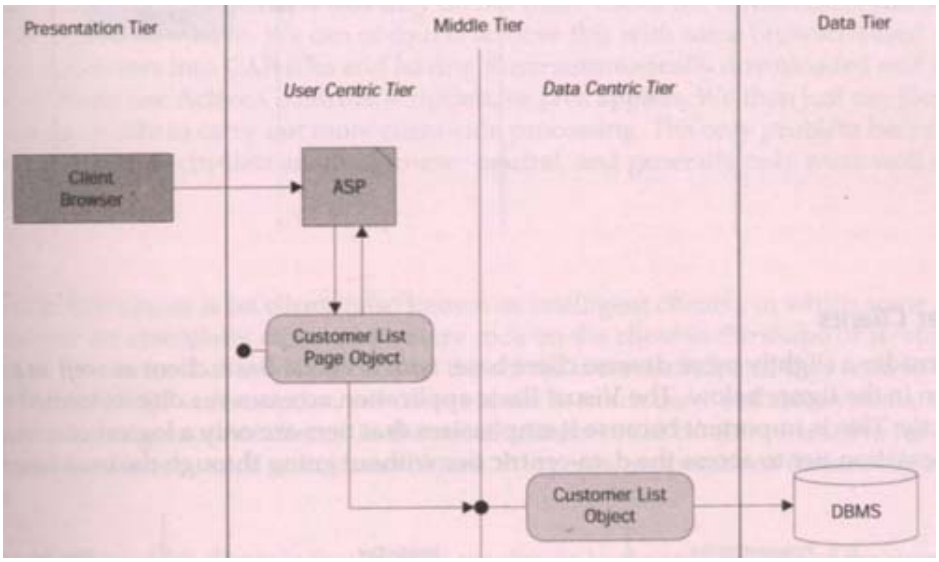
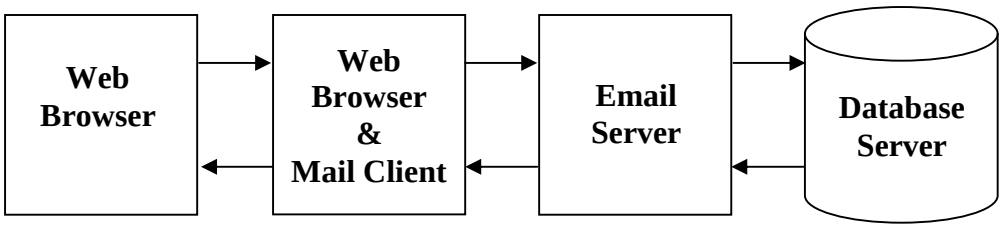
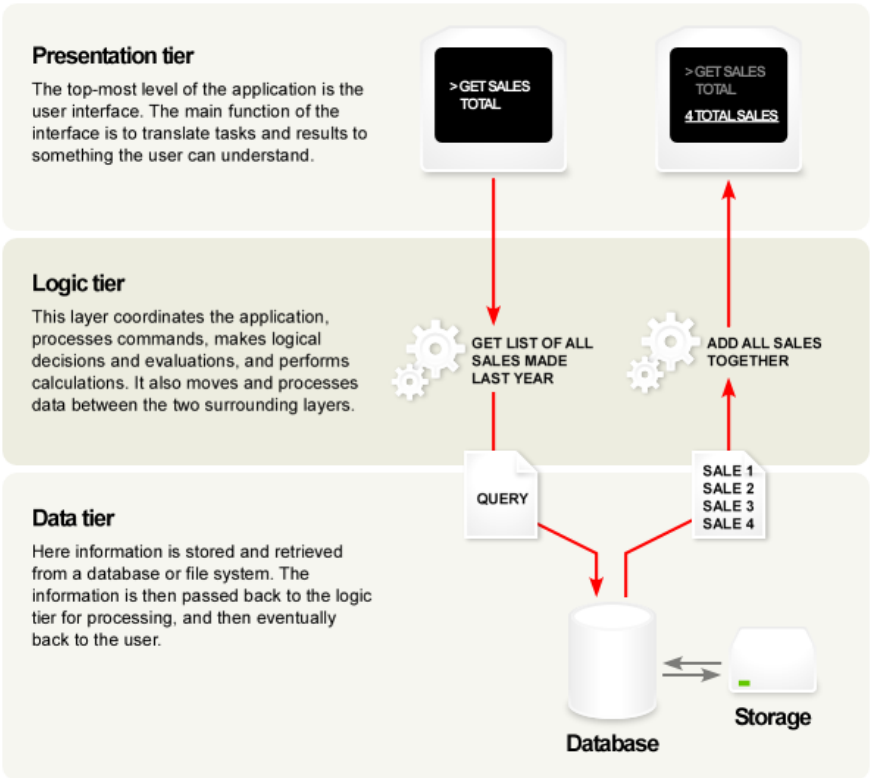


Fig: *N-Tier Technology*





A **web browser** or **Internet browser** is a software application for retrieving, presenting, and traversing information resources on the World Wide Web. An *information resource* is identified by a Uniform Resource Identifier (URI) and may be a web page, image, video, or other piece of content Hyperlinks present in resources enable users to easily navigate their browsers to related resources.

In order of release:

- WorldWideWeb, February 26, 1991
- Mosaic, April 22, 1993
- Netscape Navigator and Netscape Communicator, October 13, 1994
- Internet Explorer 1, August 16, 1995
- Opera, 1996
- Mozilla Navigator, June 5, 2002
- Safari, January 7, 2003
- Mozilla Firefox, November 9, 2004
- Google Chrome, September 2, 2008

Web Servers

Web servers are computers that deliver (*serves up*) Web pages. Every Web server has an IP address and possibly a domain name. Web servers are computers on the Internet that host websites, serving pages to viewers upon request. This service is referred to as web hosting. Every web server has a unique address so that other computers connected to the internet know where to find it on the vast network. The Internet Protocol (IP) address looks something like this: 69.93.141.146. This address maps to a more human friendly address, such as http://www.sarojpandey.com.np. Web hosts rent out space on their web servers to people or businesses to set up their own websites. The web server allocates a unique website address to each website it hosts.

For example, if you enter the URL *http://www.sarojpandey.com.np/index.html* in your browser, this sends a request to the Web server whose domain name is *sarojpandey.com.np*. The server then fetches the page named *index.html* and sends it to your browser.

Any computer can be turned into a Web server by installing server software and connecting the machine to the Internet. Two leading Web servers are Apache, the most widely-installed Web

server, and Microsoft's Internet Information Server (IIS). Other Web servers include Novell's Web Server for users of its NetWare operating system and IBM's family of Lotus Domino servers, primarily for IBM's OS/390 and AS/400 customers.

Apache HTTP Server

Apache HTTP Server (also referred to as simply "Apache") has, at the time of writing, been the most popular web server on the web since 1996. Apache is developed and maintained by the Apache Software Foundation, which consists of a decentralized team of developers. The software is produced under the Apache license, which makes it free and open source.

Apache is available for a range of operating systems, including Unix, Linux, Novell Netware, Windows, Mac OS X, Solaris, and FreeBSD.

Microsoft Personal Web Server (PWS) is a scaled-down web server software for Windows operating systems. It has fewer features than Microsoft's Internet Information Services (IIS) and its functions have been superseded by IIS and Visual Studio. Microsoft officially supports PWS on Windows 95, Windows 98, Windows 98 SE, and Windows NT 4.0. Prior to the release of Windows 2000, PWS was available as a free download as well as included on the Windows distribution CDs. PWS 4.0 was the last version and it can be found on the Windows 98 CD and the Windows NT 4.0 Option Pack.

Microsoft Internet Information Services (IIS)

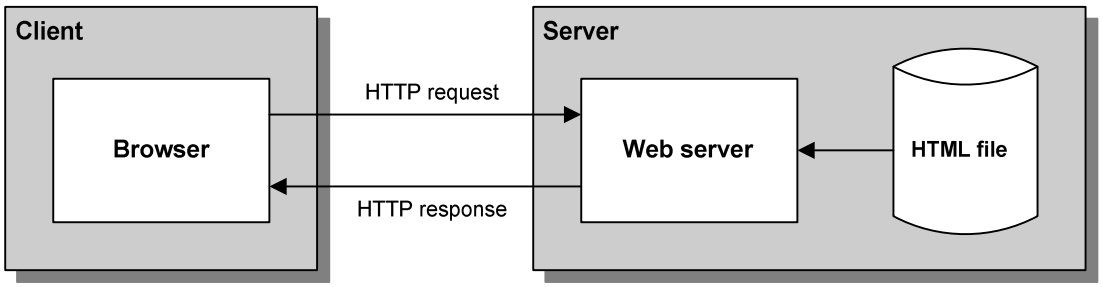
IIS is, at the time of writing, the second most popular web server on the web. It is however, gaining market share, and if the current trend continues, it won't be long before it overtakes Apache.

IIS comes as an optional component of most Windows operating systems. You can install IIS by using *Add/Remove Windows Components* from *Add or Remove Programs* in the Control Panel.

Sun Java System Web Server

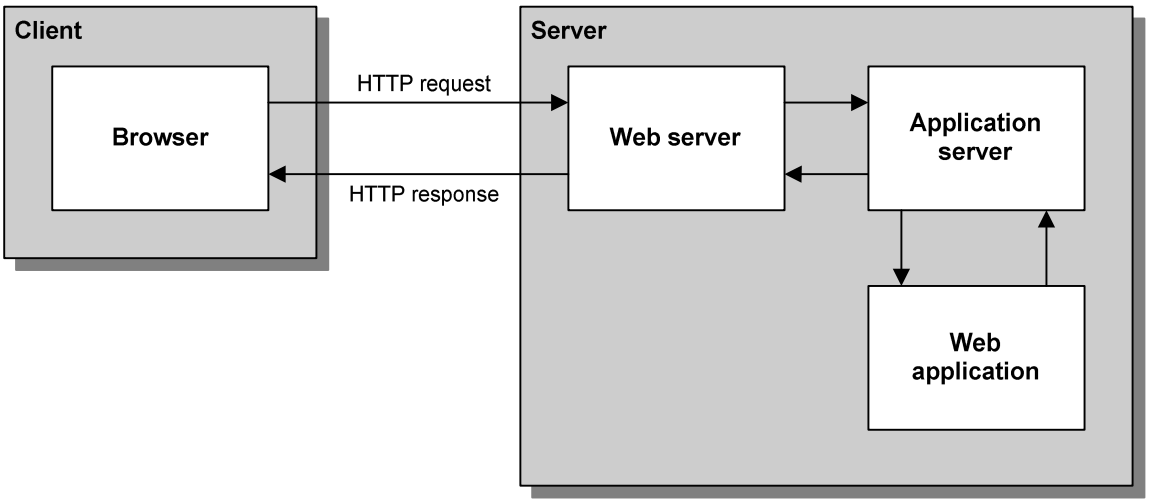
Based on the Sun One Web Server, the Sun Java System Web Server is designed for medium to large business applications. Sun Java System Web Server is available for most operating systems.

Static web page



- A *static web page* is an HTML document that is the same each time it's viewed. It doesn't change in response to user input.
- Static web pages are usually simple HTML files that are stored on the web server with a file extension of .htm or .html. When a browser requests a static web page, the web server retrieves the file from disk and sends it back to the browser.
- A web browser requests a page from a web server by sending the server an HTTP message known as an HTTP *request*.
- The HTTP request includes, among other things, the name of the HTML file being requested and the Internet address of both the browser and the web server.
- A user working with a browser can initiate an HTTP request in several ways. One way is to type the address of a web page, called a *URL*, or *Uniform Resource Locator*, into the browser's address area and press Enter. Another way is to click a link that refers to a web page.
- A web server replies to an HTTP request by sending a message known as an *HTTP response* back to the browser.
- The HTTP response contains the addresses of the browser and the server as well as the HTML document that's being returned.

Dynamic web page



- A *dynamic web page* is an HTML document that's generated by a web application. Often, the web page changes according to information that's sent to the web application by the browser.
- When a web server receives a request for a dynamic web page, the server passes the request to an *application server*.
- The application server executes the web application, which generates an HTML document. This document is returned to the application server, which passes it back to the web server. The web server, in turn, sends the document back to the browser.
- After the page is displayed, the user can interact with it using its controls. Some of those controls let the user *post* the page back to the server, so it's processed again using the data the user entered.
- To determine what application server is used to process a request, the web server looks up the extension of the requested file in a list of *application mappings*.
- Each application mapping specifies which application should be run to process files with that extension.
- If the file extension is aspx, the request is passed to ASP.NET.
- If the file extension isn't in the list of application mappings, the requested file is returned to the browser without any processing.

- The process that begins with the user requesting a web page and ends with the server sending a response back to the client is called a *round trip*.
- After a web application generates an HTML document, it ends. Then, unless the data the application contains is specifically saved, that data is lost.

HTTP Overview

- HTTP is the standard Web transfer protocol.
- The HTTP is the language that Web clients and Web servers use to communicate with each other.
- It is essentially the backbone of the web.
- The common protocol used by HTTP at transport layer is TCP which is not formally required by the standard.
- It is constantly evolving protocol with several versions in use and others are still under development.
- This protocol has two items: the set of requests from browsers to servers and the set of responses going back the other way.
- It is a stateless protocol and does not maintain any information from one transaction to the next, so the next transaction needs to start all over again
- The advantage is that an HTTP server can serve a lot more clients in a given period of time, since there's no additional overhead for tracking sessions from one connection to the next.
- Default Port used by HTTP is 80.
- ***Read More… (HTTPS)***

HTTP Transaction

All HTTP transactions follow the same general format. Each client request and server response has three parts: the request or response line, a header section, and the entity body. ***The client initiates a transaction as follows:***

1. The client contacts the server at a designated port number (by default, 80). Then it sends a document request by specifying an HTTP command called a *method*, followed by a document address, and an HTTP version number. For example:

```
GET /index.html HTTP/1.0
```

Uses the GET method to request the document *index.html* using version 1.0 of HTTP.

2. Next, the client sends optional header information to inform the server of its configuration and the document formats it will accept. All header information is given line by line, each with a header name and value. For example, this header information sent by the client indicates its name and version number and specifies several document preferences:

    User-Agent: Mozilla/2.02Gold (WinNT; I)
    Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*

    The client sends a blank line to end the header.

3. After sending the request and headers, the client may send additional data. This data is mostly used by CGI programs using the POST method. It may also be used by clients like Netscape Navigator-Gold, to publish an edited page back onto the Web server.

The server responds in the following way to the client's request:

1. The server replies with a status line containing three fields: HTTP version, status code, and description. The HTTP version indicates the version of HTTP that the server is using to respond. The status code is a three digit number that indicates the server's result of the client's request. The description following the status code is just human-readable text that describes the status code. For example, this status line:

    HTTP/1.0 200 OK

- Indicates that the server uses version 1.0 of HTTP in its response. A status code of 200 means that the client's request was successful and the requested data will be supplied after the headers.

2. After the status line, the server sends header information to the client about itself and the requested document. For example:

    Date: Fri, 20 Sep 1996 08:17:58 GMT
    Server: NCSA/1.5.2
    Last-modified: Mon, 17 Jun 1996 21:53:08 GMT
    Content-type: text/html
    Content-length: 2482

A blank line ends the header.

3. If the client's request is successful, the requested data is sent. This data may be a copy of a file, or the response from a CGI program. If the client's request could not be fulfilled, additional data may be a human-readable explanation of why the server could not fulfill the request.

   In HTTP 1.0, after the server has finished sending the requested data, it disconnects from the client and the transaction is over unless a *Connection: Key Admin* header is sent. In HTTP 1.1, however, the default is for the server to maintain the connection and allow the client to make additional requests.

   Being a stateless protocol, HTTP does not maintain any information from one transaction to the next, so the next transaction needs to start all over again. The advantage is that an HTTP server can serve a lot more clients in a given period of time, since there's no additional overhead for tracking sessions from one connection to the next.

HTTP Request methods

HTTP defines nine methods (sometimes referred to as "verbs") indicating the desired action to be performed on the identified **resource**. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

HEAD

Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.

GET

Requests a representation of the specified resource. Requests using GET (and a few other HTTP methods) "SHOULD NOT have the significance of taking an action other than retrieval".

POST

Submits data to be processed (e.g., from an HTML form) to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.

PUT

Uploads a representation of the specified resource.

DELETE

Deletes the specified resource.

TRACE

Echoes back the received request, so that a client can see what (if any) changes or additions have been made by intermediate servers.

OPTIONS

Returns the HTTP methods that the server supports for specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.

CONNECT

Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.

PATCH

Is used to apply partial modifications to a resource.

HTTP servers are required to implement at least GET and HEAD methods and, whenever possible, also the OPTIONS method.

HTTP status codes

1xx Informational

Request received, continuing process.

This class of status code indicates a provisional response, consisting only of the Status-Line and optional headers, and is terminated by an empty line. Since HTTP/1.0 did not define any 1xx status codes, servers *must not* send a 1xx response to an HTTP/1.0 client except under experimental conditions.

100 Continue

This means that the server has received the request headers, and that the client should proceed to send the request body (in the case of a request for which a body needs to be

sent; for example, a POST request). If the request body is large, sending it to a server when a request has already been rejected based upon inappropriate headers is inefficient. To have a server check if the request could be accepted based on the request's headers alone, a client must send `Expect: 100-continue` as a header in its initial request and check if a `100 Continue` status code is received in response before continuing (or receive `417 Expectation Failed` and not continue).

101 Switching Protocols

This means the requester has asked the server to switch protocols and the server is acknowledging that it will do so.

102 Processing (WebDAV) (RFC 2518)

As a WebDAV request may contain many sub-requests involving file operations, it may take a long time to complete the request. This code indicates that the server has received and is processing the request, but no response is available yet. This prevents the client from timing out and assuming the request was lost.

2xx Success

This class of status codes indicates the action requested by the client was received, understood, accepted and processed successfully.

200 OK

Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request the response will contain an entity describing or containing the result of the action.

201 Created

The request has been fulfilled and resulted in a new resource being created.

202 Accepted

The request has been accepted for processing, but the processing has not been completed. The request might or might not eventually be acted upon, as it might be disallowed when processing actually takes place.

203 Non-Authoritative Information (since HTTP/1.1)

The server successfully processed the request, but is returning information that may be from another source.

204 No Content

The server successfully processed the request, but is not returning any content.

205 Reset Content

The server successfully processed the request, but is not returning any content. Unlike a 204 response, this response requires that the requester reset the document view.

206 Partial Content

The server is delivering only part of the resource due to a range header sent by the client. The range header is used by tools like **wget** to enable resuming of interrupted downloads, or split a download into multiple simultaneous streams.

207 Multi-Status (WebDAV) (RFC 4918)

The message body that follows is an XML message and can contain a number of separate response codes, depending on how many sub-requests were made.

3xx Redirection

The client must take additional action to complete the request.

This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request. The action required *may* be carried out by the user agent without interaction with the user if and only if the method used in the second request is GET or HEAD. A user agent *should not* automatically redirect a request more than five times, since such redirections usually indicate an infinite loop.

300 Multiple Choices

Indicates multiple options for the resource that the client may follow. It, for instance, could be used to present different format options for video, list files with different extensions, or word sense disambiguation.

301 Moved Permanently

This and all future requests should be directed to the given URI.

302 Found

This is the most popular redirect cod, but also an example of industrial practice contradicting the standard. HTTP/1.0 specification (RFC 1945) required the client to perform a temporary redirect (the original describing phrase was "Moved Temporarily"), but popular browsers implemented 302 with the functionality of a 303 See Other. Therefore, HTTP/1.1 added status codes 303 and 307 to distinguish between the two behaviors. However, the majority of Web applications and frameworks still use the 302 status code as if it were the 303.

303 See Other (since HTTP/1.1)

The response to the request can be found under another URI using a GET method. When received in response to a PUT, it should be assumed that the server has received the data and the redirect should be issued with a separate GET message.

304 Not Modified

Indicates the resource has not been modified since last requested. Typically, the HTTP client provides a header like the If-Modified-Since header to provide a time against which to compare. Using this saves bandwidth and reprocessing on both the server and client, as only the header data must be sent and received in comparison to the entirety of the page being re-processed by the server, then sent again using more bandwidth of the server and client.

305 Use Proxy (since HTTP/1.1)

Many HTTP clients (such as Mozilla and Internet Explorer) do not correctly handle responses with this status code, primarily for security reasons.

306 Switch Proxy

No longer used.

307 Temporary Redirect (since HTTP/1.1)

In this occasion, the request should be repeated with another URI, but future requests can still use the original URI. In contrast to 303, the request method should not be changed when reissuing the original request. For instance, a POST request must be repeated using another POST request.

4xx Client Error

The 4xx class of status code is intended for cases in which the client seems to have erred. Except when responding to a HEAD request, the server *should* include an entity containing an explanation of the error situation, and whether it is a temporary or permanent condition. These status codes are applicable to any request method. User agents *should* display any included entity to the user. These are typically the most common error codes encountered while online.

400 Bad Request

The request cannot be fulfilled due to bad syntax.

401 Unauthorized

Similar to *403 Forbidden*, but specifically for use when authentication is possible but has failed or not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource.

402 Payment Required

Reserved for future use. The original intention was that this code might be used as part of some form of digital cash or micropayment scheme, but that has not happened, and this code is not usually used. As an example of its use, however, Apple's MobileMe service generates a 402 error "httpStatusCode:402" in the Mac OS X Console log) if the MobileMe account is delinquent.

403 Forbidden

The request was a legal request, but the server is refusing to respond to it. Unlike a *401 Unauthorized* response, authenticating will make no difference.

404 Not Found

The requested resource could not be found but may be available again in the future Subsequent requests by the client are permissible.

405 Method Not Allowed

A request was made of a resource using a request method not supported by that resource; for example, using GET on a form which requires data to be presented via POST, or using PUT on a read-only resource.

406 Not Acceptable

The requested resource is only capable of generating content not acceptable according to the Accept headers sent in the request.

407 Proxy Authentication Required

408 Request Timeout

The server timed out waiting for the request. According to W3 HTTP specifications: "The client did not produce a request within the time that the server was prepared to wait. The client MAY repeat the request without modifications at any later time."

409 Conflict

Indicates that the request could not be processed because of conflict in the request, such as an edit conflict.

410 Gone

Indicates that the resource requested is no longer available and will not be available again. This should be used when a resource has been intentionally removed and the resource should be purged. Upon receiving a 410 status code, the client should not request the resource again in the future. Clients such as search engines should remove the

resource from their indices. Most use cases do not require clients and search engines to purge the resource, and a "404 Not Found" may be used instead.

411 Length Required

The request did not specify the length of its content, which is required by the requested resource.

412 Precondition Failed

The server does not meet one of the preconditions that the requester put on the request.

413 Request Entity Too Large

The request is larger than the server is willing or able to process.

414 Request-URI Too Long

The URI provided was too long for the server to process.

415 Unsupported Media Type

The request entity has a media type which the server or resource does not support. For example the client uploads an image as image/svg+xml, but the server requires that images use a different format.

416 Requested Range Not Satisfiable

The client has asked for a portion of the file, but the server cannot supply that portion. For example, if the client asked for a part of the file that lies beyond the end of the file.

417 Expectation Failed

The server cannot meet the requirements of the Expect request-header field.

418 I'm a teapot

This code was defined in 1998 as one of the traditional IETF April Fools' jokes, in RFC 2324, *Hyper Text Coffee Pot Control Protocol*, and is not expected to be implemented by actual HTTP servers.

422 Unprocessable Entity (WebDAV) (RFC 4918)

The request was well-formed but was unable to be followed due to semantic errors.

423 Locked (WebDAV) (RFC 4918)

The resource that is being accessed is locked.

424 Failed Dependency (WebDAV) (RFC 4918)

The request failed due to failure of a previous request (e.g. a PROPPATCH).

425 Unordered Collection (RFC 3648)

Defined in drafts of "WebDAV Advanced Collections Protocol", but not present in "Web Distributed Authoring and Versioning (WebDAV) Ordered Collections Protocol".

444 No Response

An Nginx HTTP server extension. The server returns no information to the client and closes the connection (useful as a deterrent for malware).

426 Upgrade Required (RFC 2817)

The client should switch to a different protocol such as TLS/1.0.

449 Retry With

A Microsoft extension. The request should be retried after performing the appropriate action.

450 Blocked by Windows Parental Controls

A Microsoft extension. This error is given when Windows Parental Controls are turned on and are blocking access to the given webpage.

499 Client Closed Request

An Nginx HTTP server extension. This code is introduced to log the case when the connection is closed by client while HTTP server is processing its request, making server unable to send the HTTP header back.

5xx Server Error

The server failed to fulfill an apparently valid request. Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has encountered an error or is otherwise incapable of performing the request. Except when responding to a HEAD request, the server *should* include an entity containing an explanation of the error situation, and indicate whether it is a temporary or permanent condition. Likewise, user agents *should* display any included entity to the user. These response codes are applicable to any request method.

500 Internal Server Error

A generic error message, given when no more specific message is suitable.

501 Not Implemented

The server either does not recognise the request method, or it lacks the ability to fulfill the request.

502 Bad Gateway

The server was acting as a gateway or proxy and received an invalid response from the upstream server.

503 Service Unavailable

The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this is a temporary state.

504 Gateway Timeout

The server was acting as a gateway or proxy and did not receive a timely response from the upstream server.

505 HTTP Version Not Supported

The server does not support the HTTP protocol version used in the request.

506 Variant Also Negotiates (RFC 2295)

Transparent content negotiation for the request results in a circular reference.

507 Insufficient Storage (WebDAV) (RFC 4918)

509 Bandwidth Limit Exceeded (Apache bw/limited extension)

This status code, while used by many servers, is not specified in any RFCs.

510 Not Extended (RFC 2774)

Further extensions to the request are required for the server to fulfill it.

Client-side scripting is the class of computer programs on the web that are executed *client-side*, by the user's web browser, instead of *server-side* (on the web server). This type of computer programming is an important part of the Dynamic HTML(DHTML) concept, enabling web pages to be scripted; that is, to have different and changing content depending on user input, environmental conditions (such as the time of day), or other variables. Web authors write client-side scripts in languages such as JavaScript (Client-side JavaScript) and VBScript.

Client-side scripts are embedded within an HTML document (hence known as an "embedded script"), but they may also be contained in a separate file, which is referenced by the document (or documents) that use it (known as an "external script"). Upon request, the necessary files are sent to the user's computer by the web server (or servers) on whom they reside. The user's web browser executes the script, and then displays the document, including any visible output from the script. Client-side scripts may also contain instructions for the browser to follow in response to certain user actions, (e.g., clicking a button). Often, these instructions can be followed without further communication with the server. By viewing the file that contains the script, users may be able to see its source code. Many web authors learn how to write client-side scripts partly by examining the source code for other authors' scripts.

Server-side scripting is a web server technology in which a user's request is fulfilled by running a script directly on the web server to generate dynamic web pages. It is usually used to provide interactive web sites that interface to databases or other data stores. This is different from client-

side scripting where scripts are run by the viewing web browser, usually in JavaScript. The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores. Server side scripts are written in languages such as Perl, PHP, ASP.NET, JAVA, and server-side VBScript etc.

SSS produce output in a format understandable by web browsers (usually HTML), which is then sent to the user's computer. The user cannot see the script's source code (unless the author publishes the code separately), and may not even be aware that a script was executed. Documents produced by server-side scripts may, in turn, contain client-side scripts.

From security point of view, server-side scripts are never visible to the browser as these scripts are executed on the server and emit HTML corresponding to user's input to the page.

Programs that run on a user's local computer without ever sending or receiving data over a network are not considered clients, and so the operations of such programs would not be considered client-side operations.

Comparison: Client-side scripts have greater access to the information and functions available on the user's browser, whereas server-side scripts have greater access to the information and functions available on the server. Server-side scripts require that their languages interpreter be installed on the server, and produce the same output regardless of the client's browser, operating system, or other system details. Client-side scripts do not require additional software on the server (making them popular with authors who lack administrative access to their servers); however, they do require that the user's web browser understands the scripting language in which they are written. It is therefore impractical for an author to write scripts in a language that is not supported by popular web browsers.

FTP Commands & Replies

***F**ile **T**ransfer **P**rotocol,* the protocol for exchanging files over the Internet. FTP works in the same way as HTTP. FTP uses the Internet's TCP/IP protocols to enable data transfer. FTP is most commonly used to download a file from a server using the Internet or to upload a file to a server (e.g., uploading a Web page file to a server).

- File Transfer Protocol (FTP) is a method of transferring files from a client to a server or vice versa.
- Files are transferred over the Internet using TCP/IP protocol.
- FTP (RFC 959) is old-time protocol that maintains two simultaneous connections.
- The first connection uses the telnet remote login protocol to log the client into an account and process commands via the *protocol interpreter*.
- The second connection is used for the *data transfer process*.
- Whereas the first connection is maintained throughout the FTP session, the second connection is opened and closed for each file transfer.
- The FTP protocol also enables an FTP client to establish connections with two servers and to act as the third-party agent in transferring files between the two servers.
- FTP servers rarely change, but new FTP clients appear on a regular basis.
- These clients vary widely in the number of FTP commands they implement.
- Very few clients implement the third-party transfer feature, and most of the PC clients implement only a small subset of the FTP commands.
- Although FTP is a command-line oriented protocol, the new generation of FTP clients hides this orientation under a GUI environment.
- A client makes a TCP connection to the server's port 21. This connection, called the *control connection*, remains open for the duration of the session, with a second connection, called the *data connection*, either opened by the server **from** its port 20 to a negotiated client port (*active* mode) or opened by the client from an arbitrary port to a negotiated server port (*passive* mode) as required to transfer file data. The control connection is used for session administration (i.e., commands, identification, passwords) exchanged between the client and server using a telnet-like protocol. For example "RETR *filename*" would transfer the specified file from the server to the client. Due to this two-port structure, FTP is considered an *out-of-band*, as opposed to an *in-band* protocol such as HTTP.

    - ***Refer previous note for FTP Command and Replies.***