Internet Technology

Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture No #11
World Wide Web
Part – I

In the last class we are talking about electronic mail which is one of the most widely used applications on the internet. Today we will be starting our discussion on another very popular application which I am sure all of you are familiar with. You must have you must have used it in practice and it is the World Wide Web.

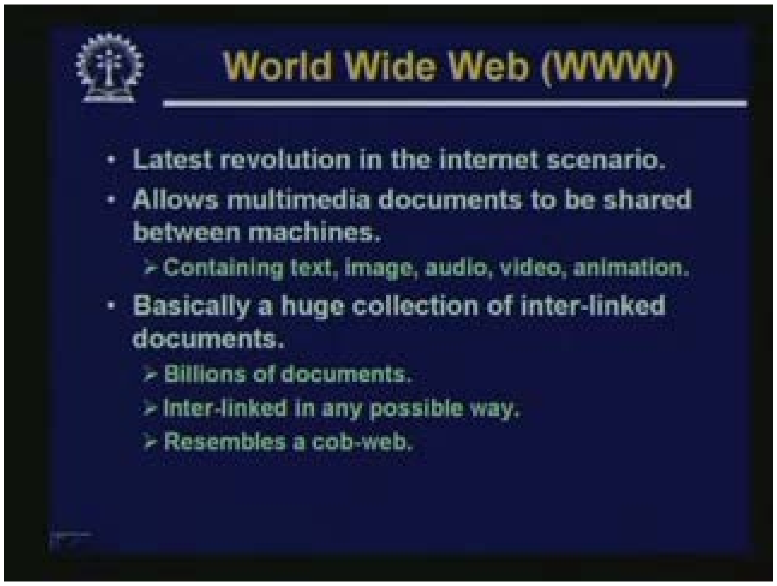
(Refer Slide Time: 01:16)

World Wide Web – Part I

Prof. Indranil Sen Gupta
Dept. of Computer Science & Engg.
I.I.T. Kharagpur, INDIA

Now this World Wide Web has become so popular that today World Wide Web and internet, they are used synonymously. Sometimes this World Wide Web or www is meant to be referring to the internet or vice versa. So this is an impact of the popularity of this www.

(Refer Slide Time: 01:43)



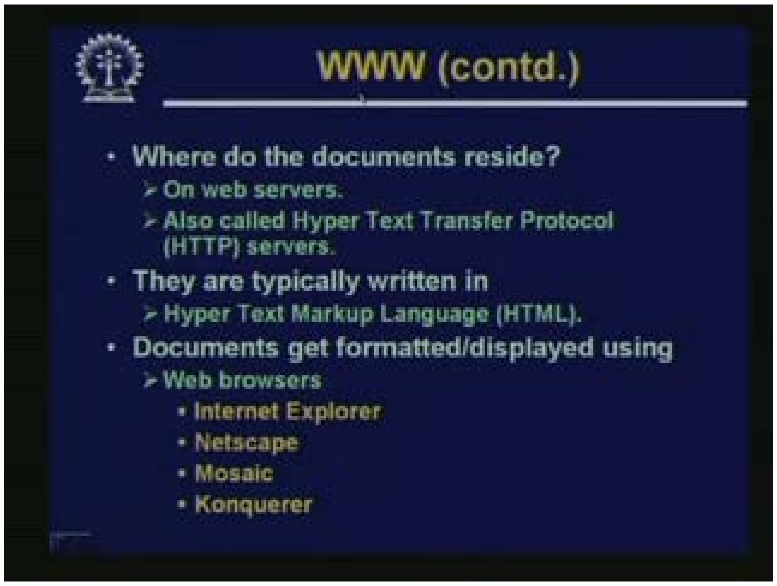
World Wide Web (WWW)

- Latest revolution in the internet scenario.
- Allows multimedia documents to be shared between machines.
  - Containing text, image, audio, video, animation.
- Basically a huge collection of inter-linked documents.
  - Billions of documents.
  - Inter-linked in any possible way.
  - Resembles a cob-web.

Now let us see what this World Wide Web actually is. This is of course the latest revolution in the internet scenario millions and millions of people or using this World Wide Web today. The main feature of WWW is that it allows us to share multimedia documents between machines. When you say multimedia we say document containing text, pictures, audio, clips, video, moving objects, animation. So all this things together can be part of the same document and we can have this kind of multimedia information which is available on a desk top which has been displayed on a browser which makes this application so very interesting and attractive. Now what this World Wide Web is actually is? This is basically a huge collection of documents which are interlinked.

Interlinked means one document is interlinked to some other document this some other document may be linked to some other document. So there can be links between one document and another you know that when you have a link you can click it on a mouse you go to some other document that is what link means. So these documents are all linked together. So there are billions of documents they are linked in all or which possible ways and since there is a very complicated interconnection of this document using this links. They resemble a cob web and from the cob web the World Wide Web this term as come. So the name World Wide Web has come from cob webs the interlinking of documents is so complex and unstructured it resembles a cob web.

(Refer Slide Time: 03:34)

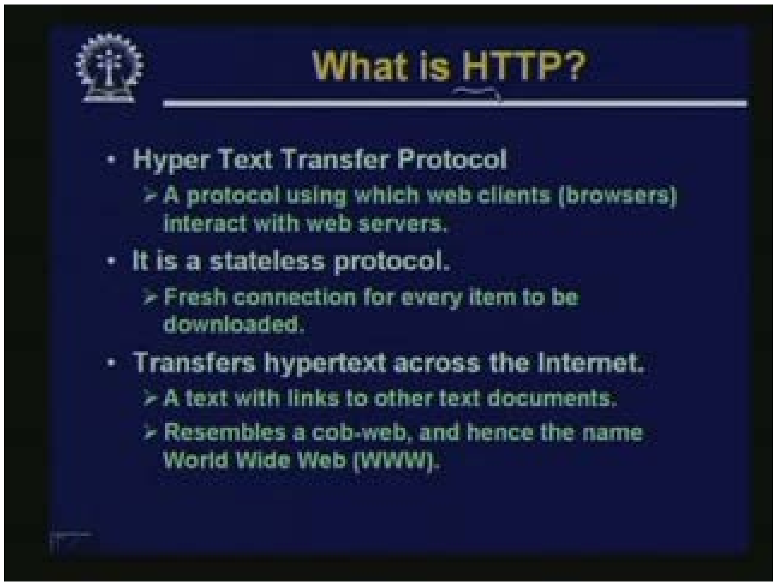


WWW (contd.)

- Where do the documents reside?
  - On web servers.
  - Also called Hyper Text Transfer Protocol (HTTP) servers.
- They are typically written in
  - Hyper Text Markup Language (HTML).
- Documents get formatted/displayed using
  - Web browsers
    - Internet Explorer
    - Netscape
    - Mosaic
    - Konquerer

Now in WWW the documents reside on web servers. Well what is a web server? Web server is a server which understands hypertext transfer protocol or http. So web servers are also sometimes known as http servers. And the documents which are residing in web servers they are typically written in a language called hypertext markup language. So they are either written in html or there are other scripts or programs which can generate html document as the output which is sent back to the browser. Finally the browser can receives html document and display it in a suitably formatted way. So the purpose of the browser the browser has the main responsibility to receive these documents and display it in a suitable formatted fashion. There are so many browsers available today I have listed some of them Explorer, Netscape, Mosaic and Konquerer there are many others are also available.

(Refer Slide Time: 4:48)



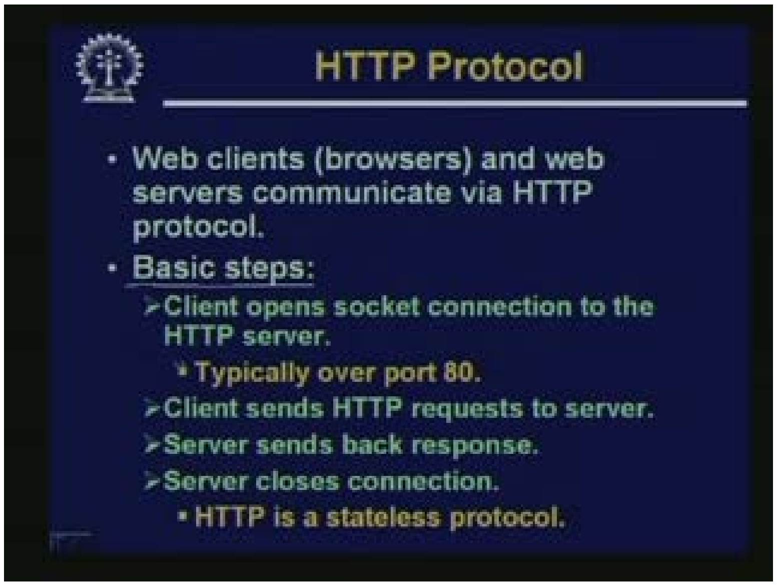
What is HTTP?

- Hyper Text Transfer Protocol
  - A protocol using which web clients (browsers) interact with web servers.
- It is a stateless protocol.
  - Fresh connection for every item to be downloaded.
- Transfers hypertext across the Internet.
  - A text with links to other text documents.
  - Resembles a cob-web, and hence the name World Wide Web (WWW).

Now this http is the driving force behind the World Wide Web. So when we talk about the World Wide Web; when you talk about web server it is the http protocol we are talking about. So the web server is running a program which is understanding the http protocol any client can send http request it will get back a http response or reply. So let us say what this http request and response is look like. So first let us try to understand http. The hypertext transport protocol transfer protocol these are protocol which the web clients which the browsers are typical example of web clients interact with web servers. This is the language using which a browser and a web server talk along with themselves. The hypertext transfer protocol the first version at least it is a stateless protocol stateless protocol means no history is maintained. Every time a request is sent a fresh connection has to be established.

If you are requesting for three documents from the same server you have to establish connection three times. Not that you are connecting only once and transferring three documents at the same time. There are of course some subsequent versions of http which allows the so called persistent connections where you can make multiple transactions over a single connection. This http hypertext across the internet sees the documents that are stored in the webpage. As I said they are some pages with links to other pages. These are called hypertext, so hypertexts are texts with links to other documents. So this http protocol allows transfer from one document to another using this hyper text by clicking a link you can go to some other place. There should be a mechanism in built in the http protocol which should support this kind of a thing.

(Refer Slide Time: 07:01)

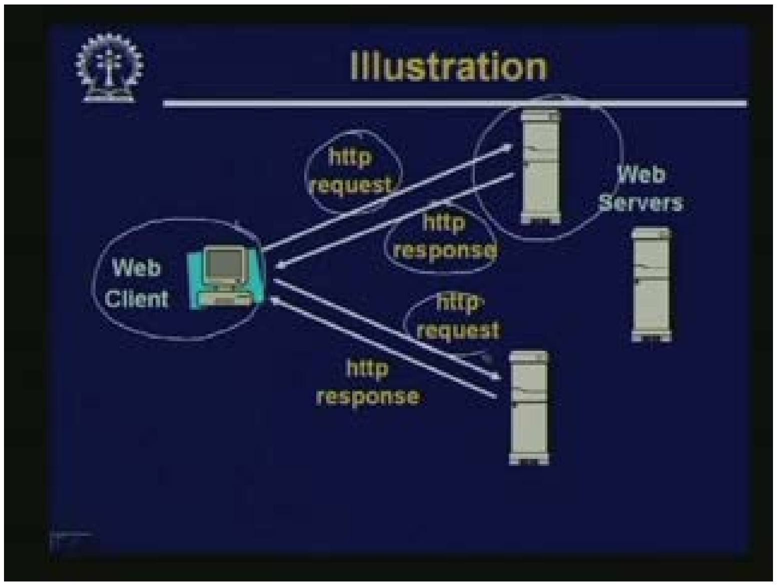


HTTP Protocol

- Web clients (browsers) and web servers communicate via HTTP protocol.
- Basic steps:
  - Client opens socket connection to the HTTP server.
    - Typically over port 80.
  - Client sends HTTP requests to server.
  - Server sends back response.
  - Server closes connection.
    - HTTP is a stateless protocol.

So as I mentioned browsers and web server communicate using http the basic steps look like this. The client first opens a socket connection to the http server typically the http server runs over port number 80 but some web servers also run over some other port number like 8080 or something else also. So you must know the port number on which the server is running. So that you can establish a connection to it typically by default the port number is 80. So after connection establish the client sends http request to the server. Server will send back response and the server will immediately close down the connection. This is the state less mode. For every connection you are opening and closing the connection, there is another request then the whole process will we will have to repeat it again. Again start a connection, again transact and then close.

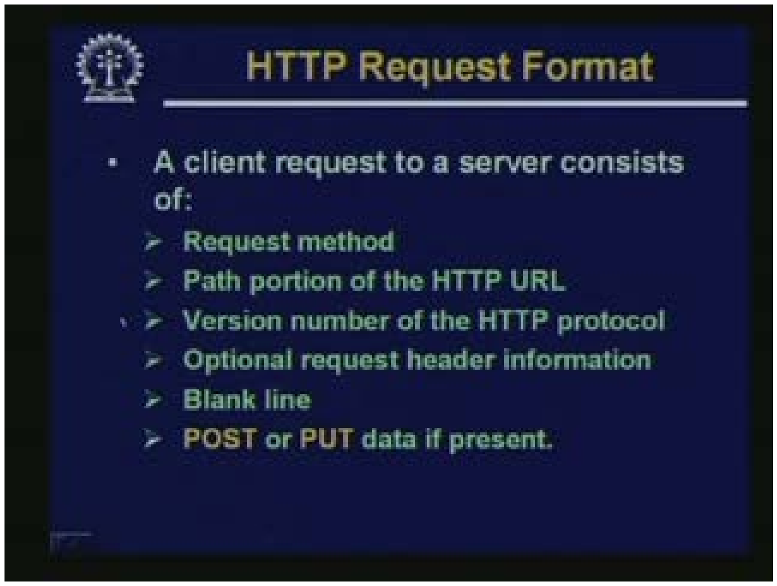
(Refer Slide Time: 08:00)



Illustration

So just look at this diagram, suppose you are sitting here. This is a web client here you are using a browser here when you are typing the address of the site say www.yahoo.com, say this is the yahoo.com server. So an http request has been sent to the yahoo.com server and the server sends back a response which is actually an html page which gets displayed on your screen. Now after the html page comes back on your page and if you click on one of those so called hyper link switch which links to other documents. Then what happens? Now another http request goes to fix the other document the other document is possibly residing in some other web server and that some other web server will again be sending you back response and this will continue. Sending a request getting back a response clicking a link going to some other server, this continues. This is called web browsing and this browsing can take you to one server to another across the world. You really do not know that where you have started and where you have ultimately stops in this process.

(Refer Slide Time: 09:18)

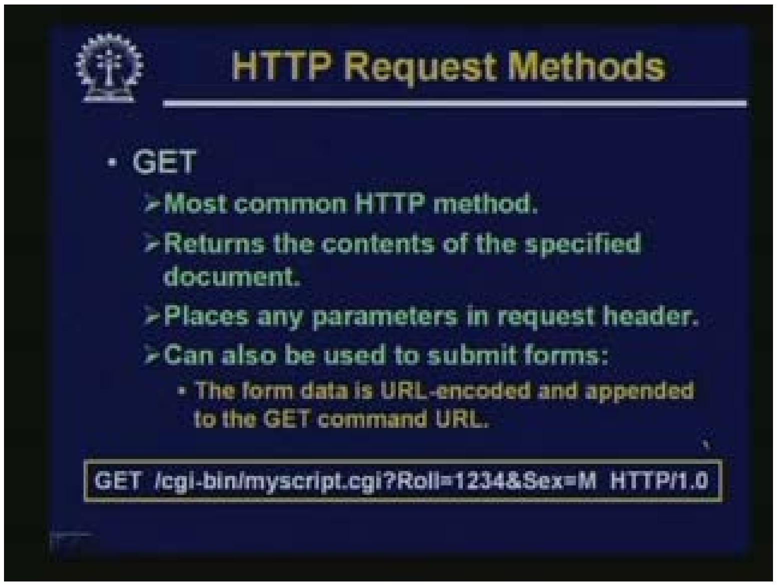


HTTP Request Format

- A client request to a server consists of:
  - Request method
  - Path portion of the HTTP URL
  - Version number of the HTTP protocol
  - Optional request header information
  - Blank line
  - POST or PUT data if present.

Now talking about the http request format a client request to the server consists of the following. It consists of a request method there are several request methods we shall be talking about get head POST port etcetera. The path position of the http URL is URL is a way of specifying the path we shall be talking about URL later in the lecture. Version number of the http protocol and for some of the request methods you may need some optional header information and in case you need to supply some additional data. The data has to be supplied after a blank line in between. This blank line acts as a delimiter between the header portion and the additional data that you are supplying. Some of the request methods like POST or PUT, they need the additional data. But for those method which do not need the data that part may not be there.

(Refer Slide Time: 10:25)



HTTP Request Methods

- GET
  - Most common HTTP method.
  - Returns the contents of the specified document.
  - Places any parameters in request header.
  - Can also be used to submit forms:
    - The form data is URL-encoded and appended to the GET command URL.

```
GET /cgi-bin/myscript.cgi?Roll=1234&Sex=M HTTP/1.0
```

So let us look at some of the important request methods one by one. GET is the most widely used method this is the method which is used to fetch a web page from a server. It returns the contents of the specified document. GET can place any parameter that you need in the request header itself. It does not need any additional lines of header information like an example is shown here. This, the get command here we are putting some additional information out here followed by the http version number. You see the first part of it cgi bin myscript.cgi specifies a path to a particular resource myscript.cgi. But after that following this question mark there is some additional data roll is equal to 1264 amber cent sex equal to m. This actually comes from so called forms. Well we shall be studying forms later when you study html in detail.

See form is something like you have seen many work pages like the search engines where some blank form like place come where you can type in something press enter or search something else comes back. So whatever you are typing in those boxes empty boxes and pressing enter you are actually filling up a form and submitting it when the data goes back to the web server it goes in this form. For example there was one space for roll no one place of sex I have typed in 1234 and m and this what which goes. This is the data you get filling you physically fill up in the form and the way this roll number is equal to amber sent this is called URL. It is URL encoding this we shall study detail when you talk about cgi scripting and html forms later in course. Of this lecture but this example shows how you can give this kind of information along with the get command.

(Refer Slide Time: 12:45)

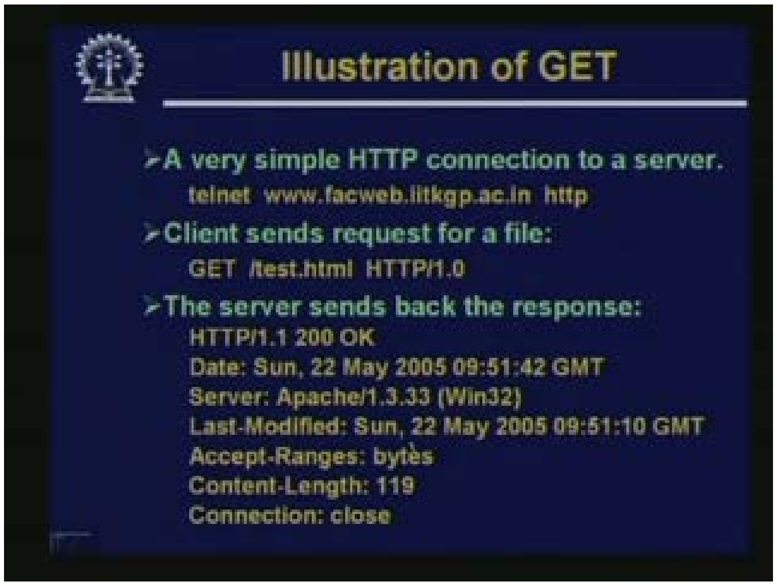


Illustration of GET

- A very simple HTTP connection to a server.

```
telnet  www.facweb.iitkgp.ac.in  http
```

- Client sends request for a file:

```
GET  /test.html  HTTP/1.0
```

- The server sends back the response:

```
HTTP/1.1 200 OK
Date: Sun, 22 May 2005 09:51:42 GMT
Server: Apache/1.3.33 (Win32)
Last-Modified: Sun, 22 May 2005 09:51:10 GMT
Accept-Ranges: bytes
Content-Length: 119
Connection: close
```

Now a very simple illustration of GET server trying to retrieve a simple documents no sending of form data. A simple document see to start with it you can just open a command window again and you can start this transaction. You can give a command telnet you can give the name of the server to which you want to connect and http you can either give http or you can give the default port number 80 whatever. This http will mean that the default port number of the http protocol well after this connection is done. You will get back a screen with nothing displayed which means that the server is expecting you to type something. So you type a command like this get is the request method slash test dot html is the document you are requesting and this is the version of the http.

The server will be sending back a response like this. First you see there are several lines of header http. The http version name of the server it is running “1.1” version date server type last modified this document when it was last modified. Accept ranges of course this is a optional we will talk about it later content length what is the size of the document connection close. Means this is the default approach that means after the transaction is over you tell me the d http connection. This is the so called stateless approach that I was talking about. So instead of close you can also specify connection keep alive that is supported in the http version “1.1” or you can leave the connection open by default.

(Refer Slide Time: 14:44)

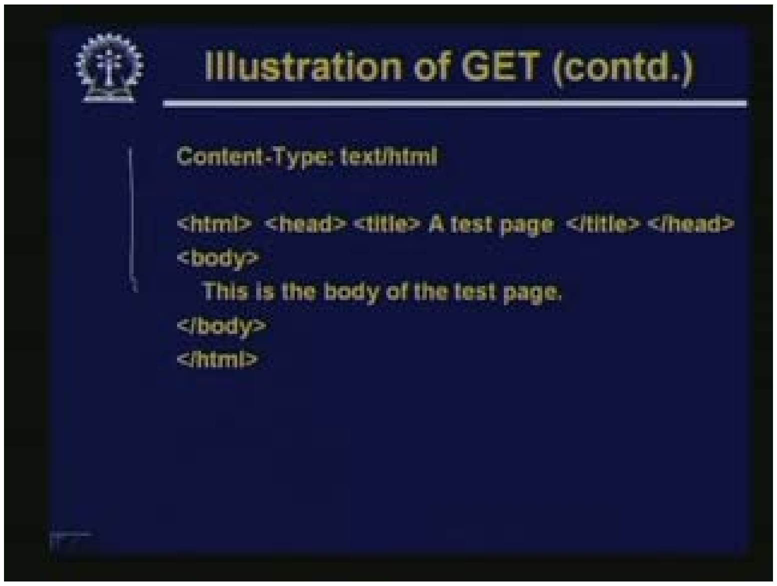


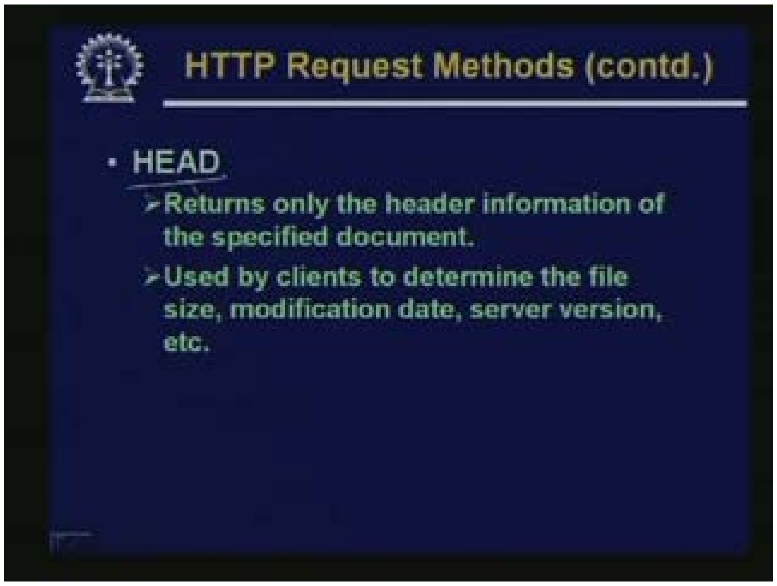
Illustration of GET (contd.)

```
Content-Type: text/html

<html>  <head> <title> A test page  </title> </head>
<body>
   This is the body of the test page.
</body>
</html>
```

And after these lines come this content type. These are text of type html here there is a blank line you see there is a blank line in between this blank line indicates or it acts a delimiter that now the actual data follows. So the actual content of the file follows after this. This is a free simple html file which is created to give you this demonstration. So the content of the html file will come after that. So you see that this command get command can be used to fetch an html file from a server if you know of course the name of the file.

(Refer Slide Time: 15:32)



HTTP Request Methods (contd.)

- HEAD
  - Returns only the header information of the specified document.
  - Used by clients to determine the file size, modification date, server version, etc.

There is another request method called head which returns only the header information. Well you may not be interested to get the whole file; but you need some information about the file like the file size when was it last modified server version etcetera.

(Refer Slide Time: 15:52)

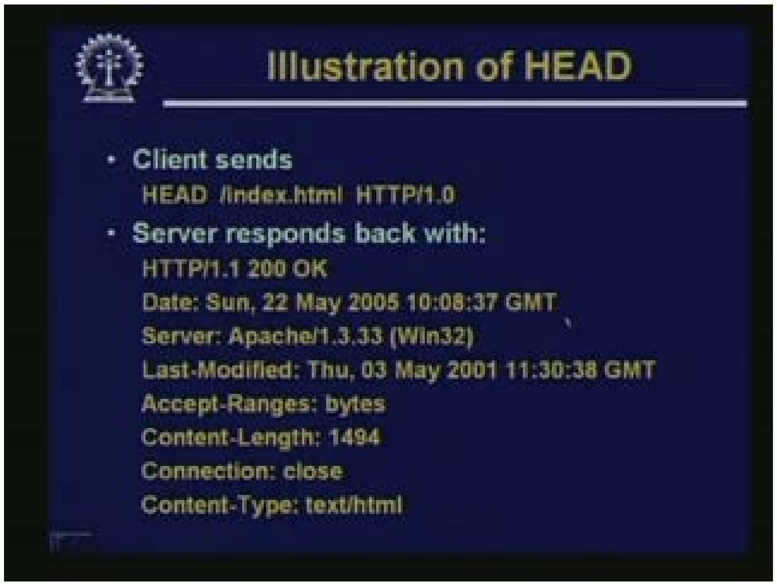


Illustration of HEAD

- Client sends

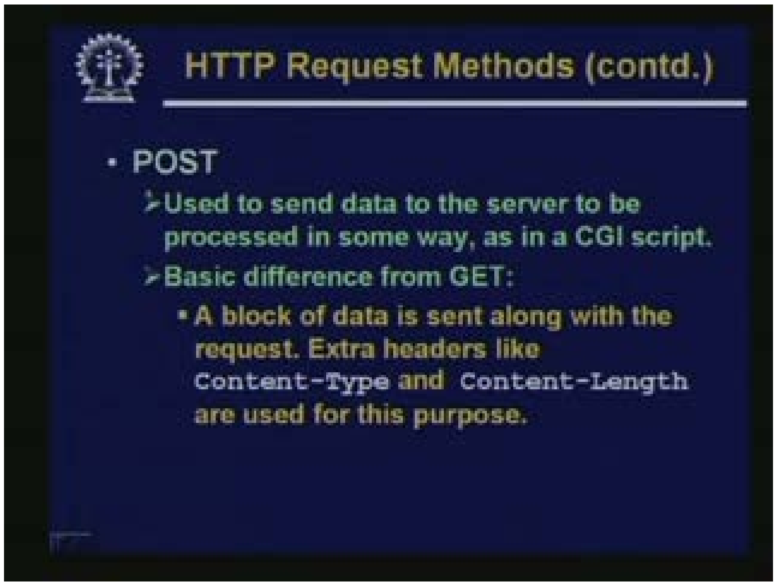
```
HEAD /index.html HTTP/1.0
```

- Server responds back with:

```
HTTP/1.1 200 OK
Date: Sun, 22 May 2005 10:08:37 GMT
Server: Apache/1.3.33 (Win32)
Last-Modified: Thu, 03 May 2001 11:30:38 GMT
Accept-Ranges: bytes
Content-Length: 1494
Connection: close
Content-Type: text/html
```

So a typical session with head will look like this after doing the telnet, you send a command like this. Head name of a file followed by http version you get back only the header portion. What is missing is actually the body of the contents of the file that you are not requesting here. You are requesting only the header portion. So only the header portion is coming back to you and you can have a look at the header position to see what it contains. If example this, file size is 1494 bytes.

(Refer Slide Time: 16:27)

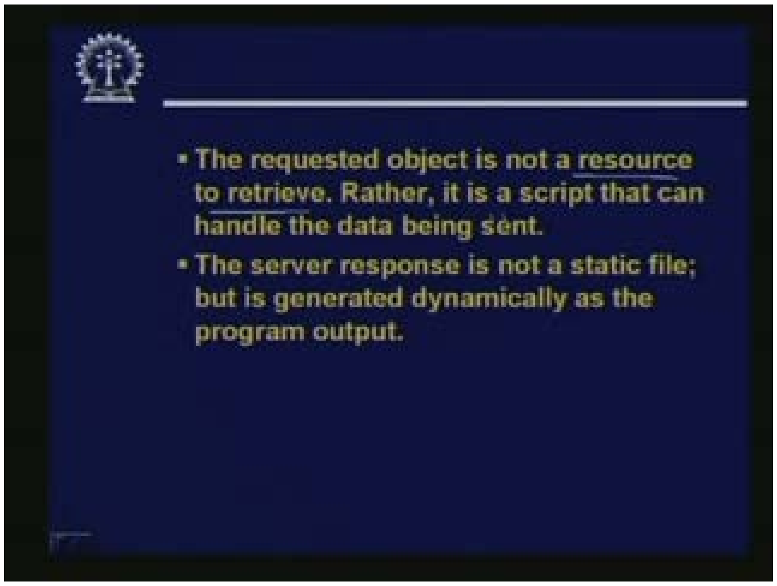


HTTP Request Methods (contd.)

- POST
  - Used to send data to the server to be processed in some way, as in a CGI script.
  - Basic difference from GET:
    - A block of data is sent along with the request. Extra headers like `Content-Type` and `Content-Length` are used for this purpose.

POST is another request method. POST is again another method which is used in conjunction with the submission of the form. So we will be talking about POST now. But you can actually put it in a proper cont to a context when we discuss suggest script later during our class. Now this POST is used to send some additional data to the server to be processed typically by a cgi script; cgi script is a program cgi stands for common gateway interface it is a program which running on the server side. Whenever you fill up a form typically the data in the form goes to the cgi script; the cgi script reads the data and does some processing and generates some output which comes back to the client and the browser. It gets displaced there; this is how cgi scripts work. Now this POST request method allows you send data to the cgi script, but the why this data sent is different from the gate approach that we have discussed. But this uses two addition header type; content type and content length.

(Refer Slide Time: 17:49)



- The requested object is not a resource to retrieve. Rather, it is a script that can handle the data being sent.
- The server response is not a static file; but is generated dynamically as the program output.

The requested object is not a resource to retrieve. Rather it is a script an executable script. Similarly server response which comes back is not a static file. But rather it is the output of the cgi program which I have seen. So the server is not sending you back the contents of a file which already there. Rather it is sending you back the output of that cgi program. For instance which is receiving the data executing it and sending back the output to you?

(Refer Slide Time: 18:24)

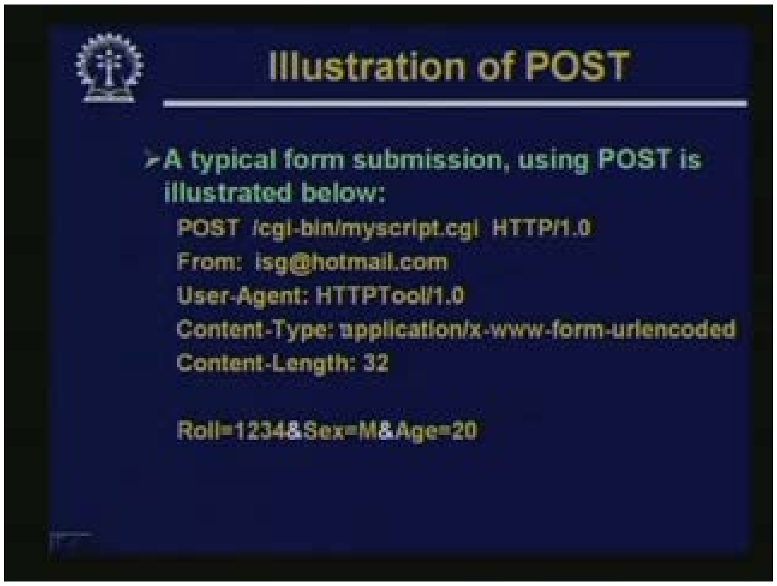


Illustration of POST

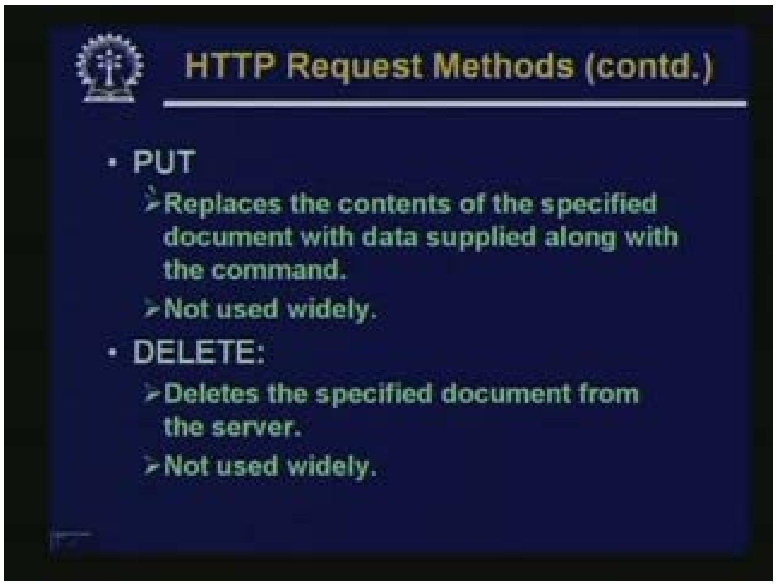
➢ A typical form submission, using POST is illustrated below:

```
POST /cgi-bin/myscript.cgi HTTP/1.0
From: isg@hotmail.com
User-Agent: HTTPTool/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 32

Roll=1234&Sex=M&Age=20
```

Now the POST command can be illustrated with the simple example. Well again after telnet, suppose you give a command like this, POST followed by a path name. This cgi slash bin myscript.cgi. This does not refer to a document or a resource which you want to download; rather this is the name of the program. The cgi script program for instance which you want to execute whenever this POST command is given. And this is the additional information you will also have supply after POST. See, in GET or head you give a one line command and a lot of response come back. But in a POST you will have to give a multiple line command. This POST is the first lines followed by from you have to identify yourself from where you are sending user agent the name or the version of the user agent that is running on your machine, content type or type of content it is. Well application something x-www form urlencoded this is an URL encoded see this is the actual content. The way this has been encoded this is called urlencoding and the name is x www form urlencoded and content length the total size is 32. There is a blank line in between followed by the actual body of this additional information. So in GET this information is put on the same line after a question mark; in POST it is put as part of the command after a blank line. Now, in POST you can put as much material as you can. So if we have a lot of data to send, POST is the preferred approach because in GET the total size of the string is limited by usually 256. The maximum size of the string that the machine can support. So other request method PUT.

(Refer Slide Time: 20:32)

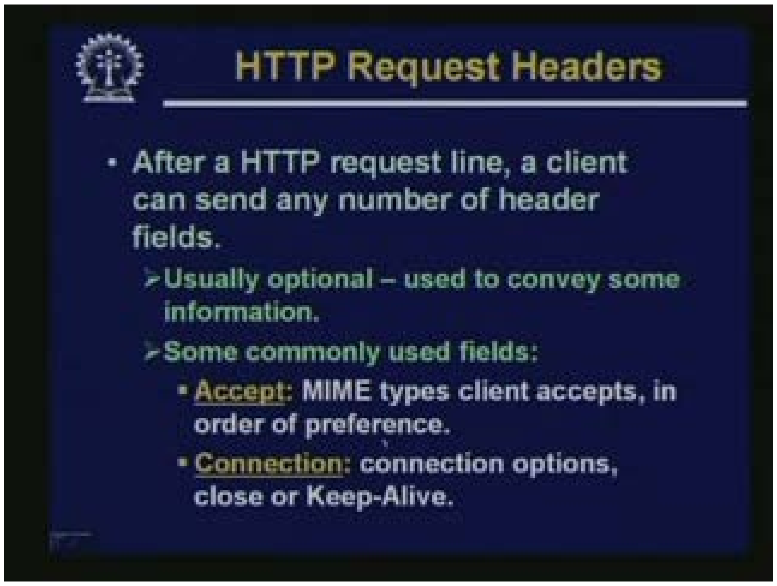


HTTP Request Methods (contd.)

- PUT
  - Replaces the contents of the specified document with data supplied along with the command.
  - Not used widely.
- DELETE:
  - Deletes the specified document from the server.
  - Not used widely.

PUT is used to upload the contents of a webpage. It replaces the contents of the specified document with the data supplied along with the command. So here the data that was supplying, this you can supply exactly like the POST command. So after some initial header give a blank line, then give your entire web page which you want to upload. So the entire thing will be the data you are trying to send to the server; the server will handle it and it will be uploading it. But of course as you can understand. Not all web servers allow you uploading facility. Only a few web server will allow to do that and of course only after proper authentication. Delete is similar in a sense that this also makes updation this can delete a specified document this also is not used very widely.

(Refer Slide Time: 21:24)

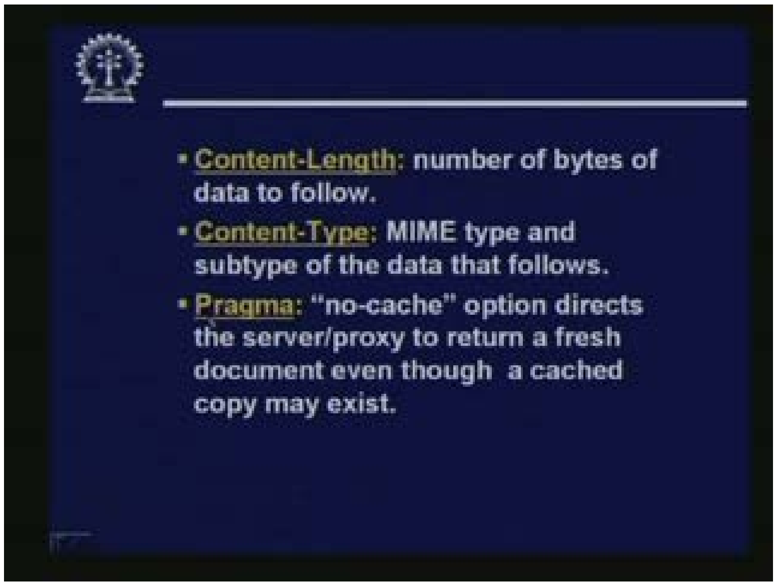


HTTP Request Headers

- After a HTTP request line, a client can send any number of header fields.
  - Usually optional – used to convey some information.
  - Some commonly used fields:
    - Accept: MIME types client accepts, in order of preference.
    - Connection: connection options, close or Keep-Alive.

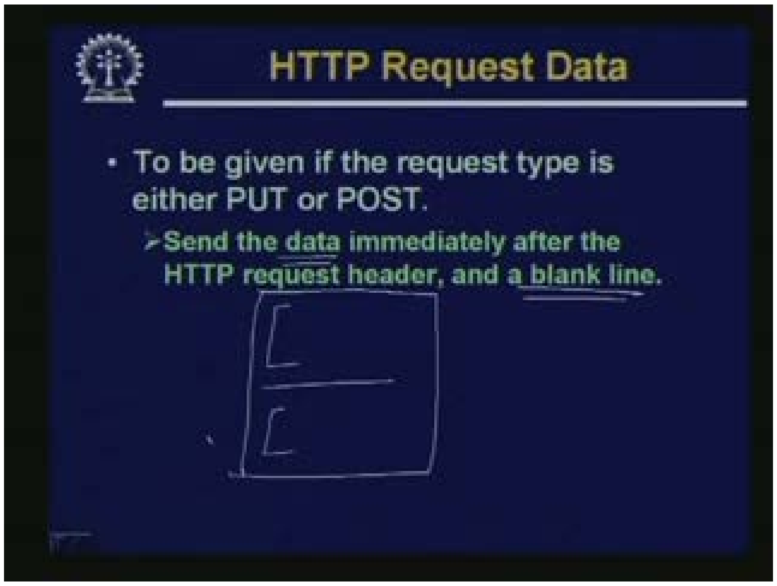
Only in some servers where you have proper authentication you can do that. And after sending the request, well. There some request header you see. That means when you are sending the request some request header information is required. Now after the initial line GET, POST, PUT, etcetera. So a client can send any number header lines. Some of these where illustrated with the example for POST. So after the POST command there were several additional lines of header followed by the data that were supplied. So this is usually optional for some request types like get you do not need it for some others you need it. Some of the common headers types are accept. This accept followed by a string indicates at which MIME types the client will accept. Connection. Connection is the connection type; if it is close you are asking it to close connection after every transaction. If it keeps alive, then you are saying that well it will not be stateless. You let the connection to be persistent I can send more than one file one transaction over that open connection. So you can specify this as part of the header.

(Refer Slide Time: 22:50)


- Content-Length: number of bytes of data to follow.
- Content-Type: MIME type and subtype of the data that follows.
- Pragma: "no-cache" option directs the server/proxy to return a fresh document even though a cached copy may exist.

Content length of course you have seen number of bytes of data. Content type what MIME type of data is there. There is a pragma header; pragma colon followed by no cache. This indicates that if you are redirecting your command to a proxy server. You are saying that you please do not send me my requested information from your cache you try to send me a fresh copy from the original server.

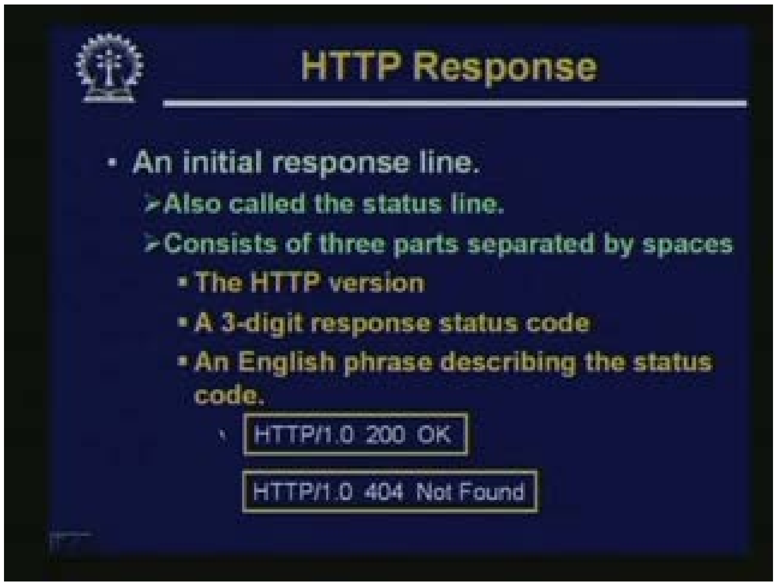
(Refer Slide Time: 23:36)


HTTP Request Data

- To be given if the request type is either PUT or POST.
  - Send the data immediately after the HTTP request header, and a blank line.

So this pragma no cache tells you to return a fresh document well although a cached copy may exists, you may want sometime that you want the latest updated copy from the original server. And for command switch request we require request data like the POST and PUT. So you will have to put a blank line followed by the actual data. So there will be the initial header followed by a blank line then the data part. This will be the structure of your overall http request.

(Refer Slide Time: 24:04)



HTTP Response

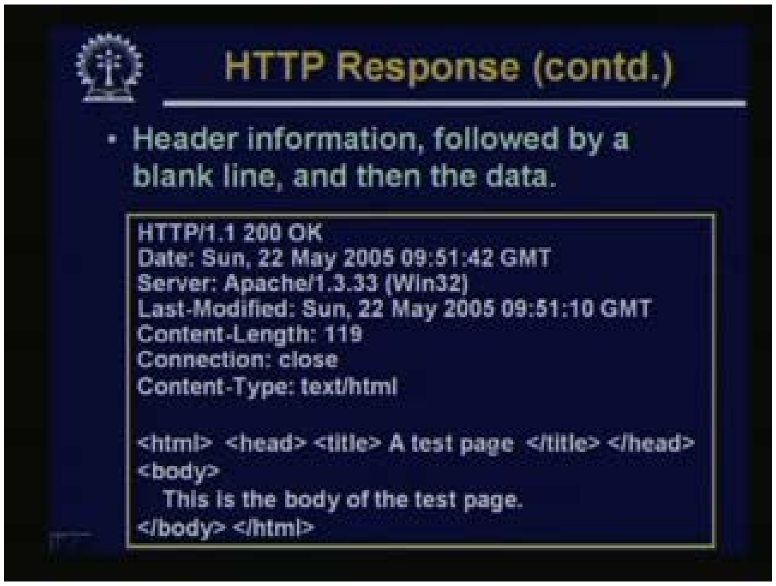
- An initial response line.
  - Also called the status line.
  - Consists of three parts separated by spaces
    - The HTTP version
    - A 3-digit response status code
    - An English phrase describing the status code.

```
HTTP/1.0 200 OK
```

```
HTTP/1.0 404 Not Found
```

Now after the request is sent, now it is time to get back the responses. So the responses look similar the requests for most of them other than get. There are some standard request headers followed by optional data. The responses also look very similar. There will be an initial response line which is also called the status line; response status line. It consists of three parts the http version, a three digit response code and an English phrase indicating what type it is. Two examples I shown it will either be http version name 200 or version name 404 Not Found. This 200 404 are the error codes and Not Found are the error types.

(Refer Slide Time: 24:59)



HTTP Response (contd.)

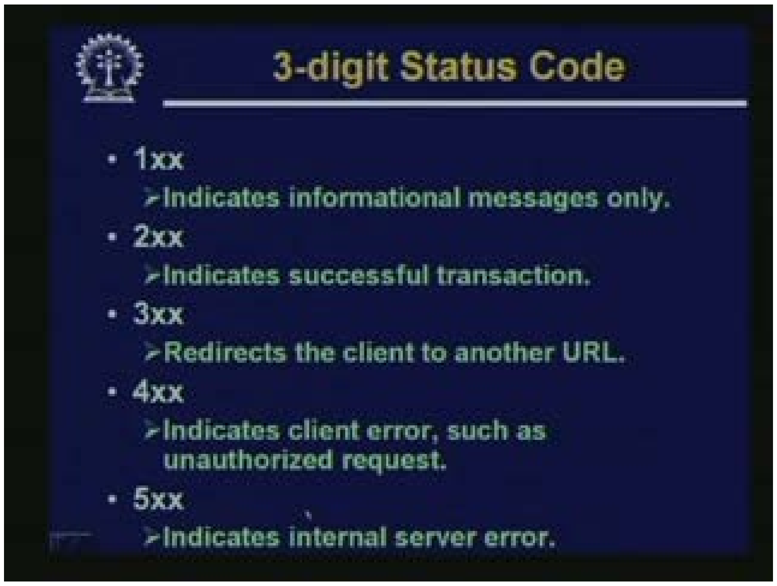
- Header information, followed by a blank line, and then the data.

```
HTTP/1.1 200 OK
Date: Sun, 22 May 2005 09:51:42 GMT
Server: Apache/1.3.33 (Win32)
Last-Modified: Sun, 22 May 2005 09:51:10 GMT
Content-Length: 119
Connection: close
Content-Type: text/html

<html>  <head> <title> A test page  </title> </head>
<body>
   This is the body of the test page.
</body> </html>
```

And after this initial status line will come, the actual content of the response. There will be some header at the beginning followed by the actual data and again a usual there will be blank line in between. So there will be a blank line separating them. So the header looks very much like the http request line. There will be some header types followed by the value. The content length, connection close, content type and at the end you will get the data that you are requesting.

(Refer Slide Time: 25:42)

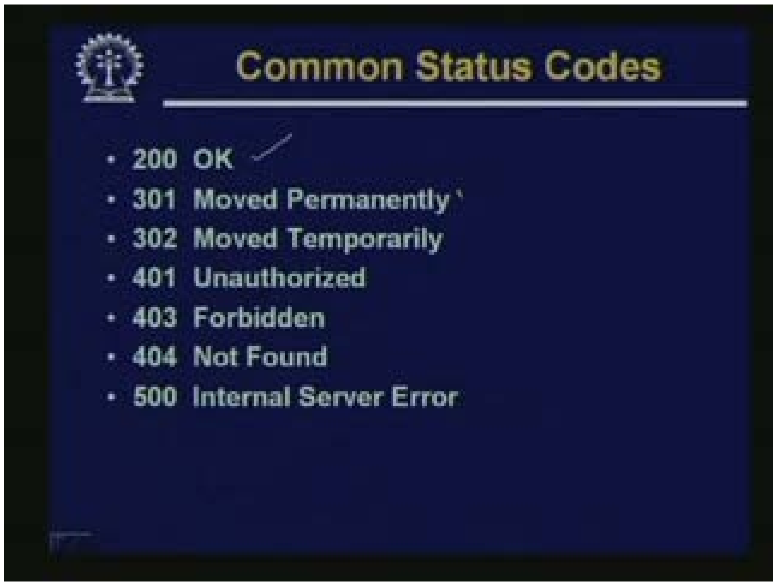


3-digit Status Code

- 1xx
  - Indicates informational messages only.
- 2xx
  - Indicates successful transaction.
- 3xx
  - Redirects the client to another URL.
- 4xx
  - Indicates client error, such as unauthorized request.
- 5xx
  - Indicates internal server error.

And the 3 digit status code indicates anything starting with one, indicates that this is only information. This is not an error message. You need not do anything special with respect to this. This is just for your information some informational message. Two, anything starting with 2 indicates successful transaction. Anything starting with 3 indicates that you have been redirected to another URL; some servers supports redirection. Starting with 4 means some error condition; 5 means there is some internal server error. So these kinds of errors are reported with an error code starting with these initial digits.

(Refer Slide Time: 26:27)

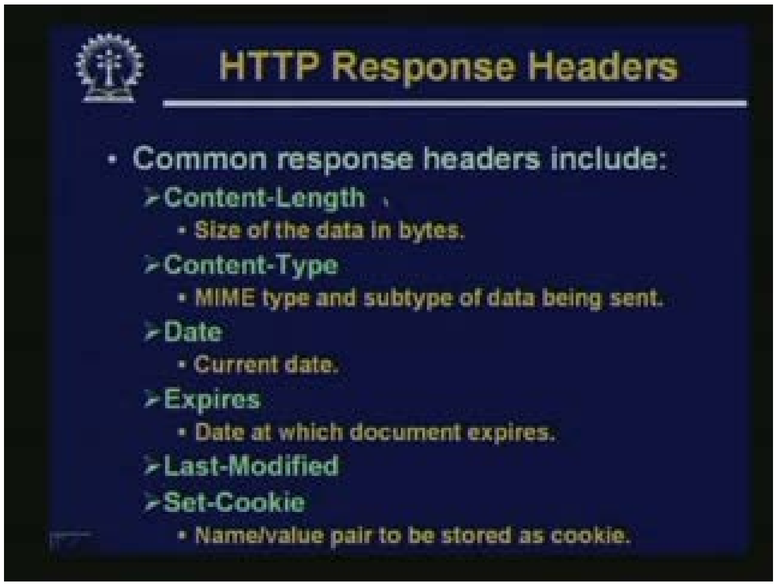


Common Status Codes

- 200 OK
- 301 Moved Permanently
- 302 Moved Temporarily
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error

Some of the common status codes are like this 200 means OK, 301 some documentary requesting has been moved permanently. 302 means temporarily; 40 means there is some authorization problem may be you have not supplied the password correctly. 403, some documents some access permissions are not there; 404 Not Found, 500 some server error.

(Refer Slide Time: 26:55)

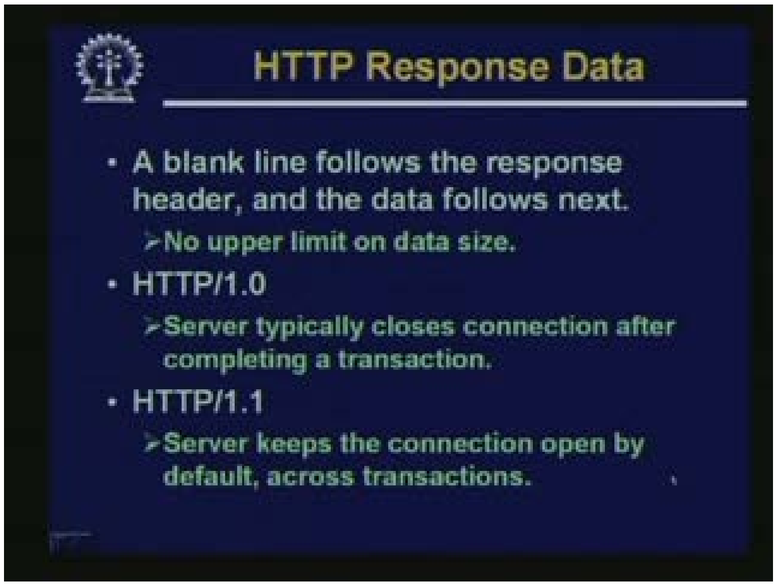


HTTP Response Headers

- Common response headers include:
  - Content-Length
    - Size of the data in bytes.
  - Content-Type
    - MIME type and subtype of data being sent.
  - Date
    - Current date.
  - Expires
    - Date at which document expires.
  - Last-Modified
  - Set-Cookie
    - Name/value pair to be stored as cookie.

Now in the response header there are several things which are included like content length you already know it specify the size of the data in bytes. Content type, again MIME type and subtype what kind of data is coming. Date means the current date. Well each document that resides on the server can also have something like an expiry date associated with it. So whenever you are fetching a document it will also contain what is the expiry date? Some document may contain an expiry date; some may not contain. If it contains, then we will have a look at the expiry date and find out whether this is an absolute document or not. There is a last modified the name is obvious what it means and set-cookie. Set-cookie is another information which is sometimes sent back. See set-cookie followed by some name value pair. See whenever you want to have cookie in your browser, cookie has some values name and value pair which is sent from the other side which can be used by a browser to do some kind of a local processing. Like when you are maintaining a session you use cookies you maintain some information about the session and session id. You continuously check whether your present session variable matches with the session id stored in the cookie. If not you give an error message that your session is expired something like that. So this cookie can also be sent back.

(Refer Slide Time: :)



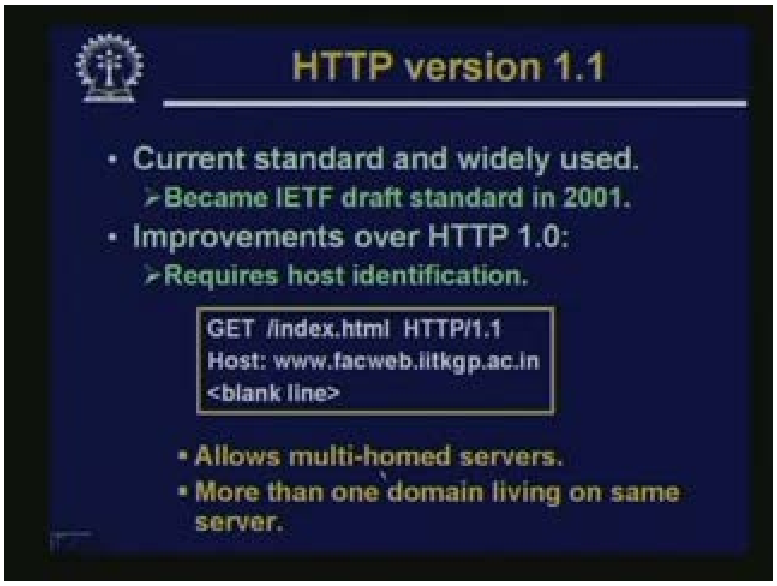
HTTP Response Data

- A blank line follows the response header, and the data follows next.
  - No upper limit on data size.
- HTTP/1.0
  - Server typically closes connection after completing a transaction.
- HTTP/1.1
  - Server keeps the connection open by default, across transactions.

And after the initial header, again there will be a blank line which will be followed by the actual data. So the way the requests are send and the responses are sent back are very similar. They use the mind format for the heading a blank line then followed by the data. There is no upper limit to the data size. It can be very big also, but two versions of http which you see around you today. This http “1.0” is the older version. But most of the modern web servers use the later version “1.1”. Now in “1.0” the server typically closes connection after completing a transaction that is the default. Well in “1.1” the server keeps the connection, open it is persistent.

So it depends on the scenario that you are working in whether you want to keep it persistent or not. See for servers which are very heavily used keeping the connection persistent will add the load on the server. Because every machine has a maximum limit to the number of connections that it can handle simultaneously. So if it is a persistent and there are many connections which are coming and remaining because it is active for long time, it may so happen that the maximum limit may reach and after that request which are coming they will not be honored they will be denied. So for most servers which are pretty popularly used they prefer to use stateless or non-persistent connection type.

(Refer Slide Time: 30:21)



HTTP version 1.1

- Current standard and widely used.
  - Became IETF draft standard in 2001.
- Improvements over HTTP 1.0:
  - Requires host identification.

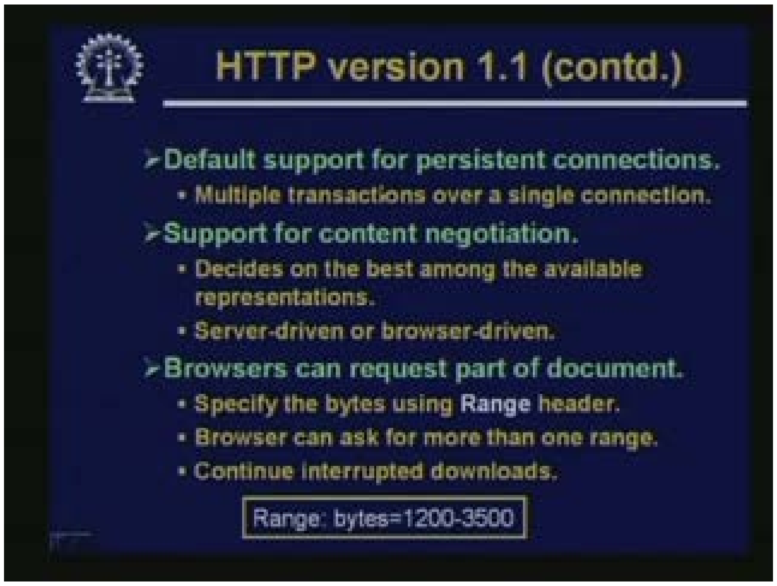
```
GET /index.html HTTP/1.1
Host: www.facweb.iitkgp.ac.in
<blank line>
```

- Allows multi-homed servers.
- More than one domain living on same server.

Now http version "1.1", let us try to see in what way this is different from the http "1.0". See http version "1.0" has become an IETF graph standard as early as in 2001. So details about this version are available on the net. Now the improvements that have taken place over http version "1.0" are this version requires mandatory host identification. See what is meant is that, in the conventional get command in version "1.0" you needed only a single line but in version "1.1", you also need an additional line after that host colon followed by the domain name of the server. What does this mean? Suppose I want to get a document from a server. I give a GET command next line I give following the host type. The header type I follow it with the name of the server. But you can argue any way I am establishing a connection to the server.

Either by using an explicit telnet or by typing the address on the browser. So why do I have to specify the name of server again? See this is required for some of the server types which we see around us today. These are called multi homed servers. Multi homed server means there is a single machine but more than one domain reside on them. Like what I mean to say is that there is a single server computer. But on the same computer there is one domain name xyz.com. There is another domain name say for example abc.com both are residing. So whenever I am sending a request to that computer I must identify that which of the two servers I am actually trying to send my request to. So what for this purpose explicit server identification is required and of course there has to be a blank line after that and then enter.

(Refer Slide Time: 32:55)



HTTP version 1.1 (contd.)

- Default support for persistent connections.
  - Multiple transactions over a single connection.
- Support for content negotiation.
  - Decides on the best among the available representations.
  - Server-driven or browser-driven.
- Browsers can request part of document.
  - Specify the bytes using **Range** header.
  - Browser can ask for more than one range.
  - Continue interrupted downloads.

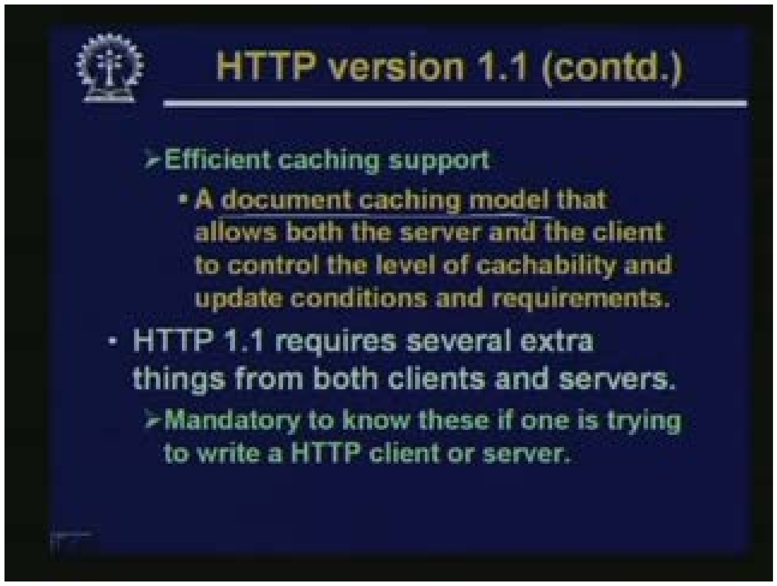
Range: bytes=1200-3500

Default support for persistent connection. This I have already mentioned. S that by default multi multiple transactions takes place over a single connection. But if you want you can make the connection type as close and you can make it a state list. Support for content negotiation. Let us see you can have several alternate representations. This may be respect to the browser, this is may be with respect to the server also. So you can negotiate and find out which of these are the best among the available ones. This may be driven by the server. This may be driven by the browser. Like you are looking for a document. The document may be presented to you in one several formats possibly the browser can give a preference. That well I prefer the document in html.

Well if it is not possible for you to give an html, give me in some other format like pdf format like that. So these are some preferences which have to be negotiated and this can be negotiated based on some initiative made by browser or also the servers; both are possible. Now one interesting thing which http version “1.1” supports which was not there in “1.0” is that browsers can request part of the document. See earlier what happened, say for example when you are accessing net you are clicking on a link and some file has been downloaded suppose it was a big file. Now in between if due some error if the connection got broken then again if you have to resume the connection. You have to start from the beginning.

Because ultimately request for the document has to go as http request that I want that document. But in this new feature what has been incorporated is that you can specify that I want this document. But you start sending me this document from this particular offset may a part of this document I already have with me. I want to start down loading from this particular point. This helps in saving a lot of time for downloading. This is can be done by using the range header type by specifying bytes like an example range colon bytes equal to 1200 dash 350. This means that I want to fetch only bike number 1200 up to 3500. Browsers can ask for more than one range. There can be several such range lines. This as I mentioned this allows us to continue with interrupted downloads.

(Refer Slide Time: 35:55)

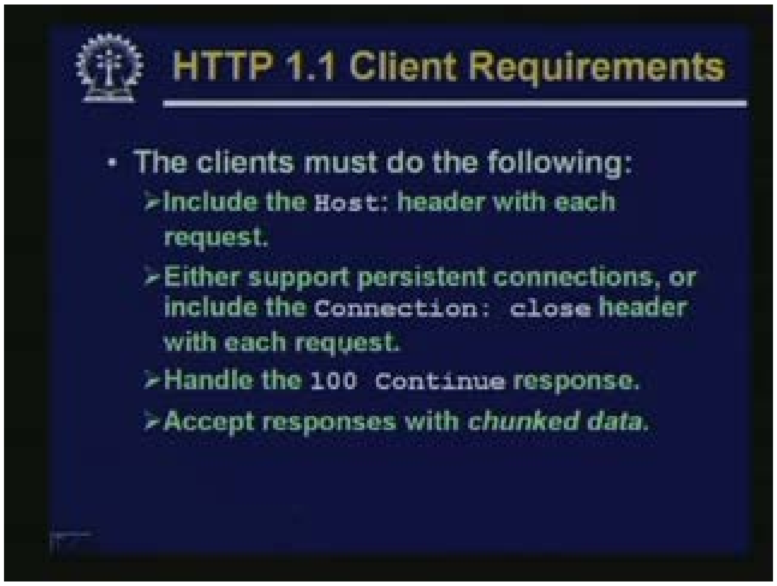


HTTP version 1.1 (contd.)

- Efficient caching support
  - A document caching model that allows both the server and the client to control the level of cachability and update conditions and requirements.
- HTTP 1.1 requires several extra things from both clients and servers.
  - Mandatory to know these if one is trying to write a HTTP client or server.

The http version one allows efficient caching support. This is another feature which has been added. So this supports a document caching model that allows both the server and client to control the two what at extent you need to maintain the cache. See when you say I am maintaining cache. There are a number of things. What is the size of the cache? Up to how many days I will maintain information in the cache? What is my cache architecture? Lot of things can be negotiated upon. So using this caching support through some specific headers, the client and the server can negotiate and fix up exactly what kind of caching you desire. See earlier the caching architecture was fixed statically. But here it can be negotiated upon and you can fix upon an architecture. But in order to have all this facilities working http “1.1”. Obviously require several extra things from both client and server. Just one example I have just given that explicit host identification is required after the GET command the client must also send that extra host command. So if you are writing a new client today you must keep this in mind; you must incorporate this in your architecture.

(Refer Slide Time: 37:28)

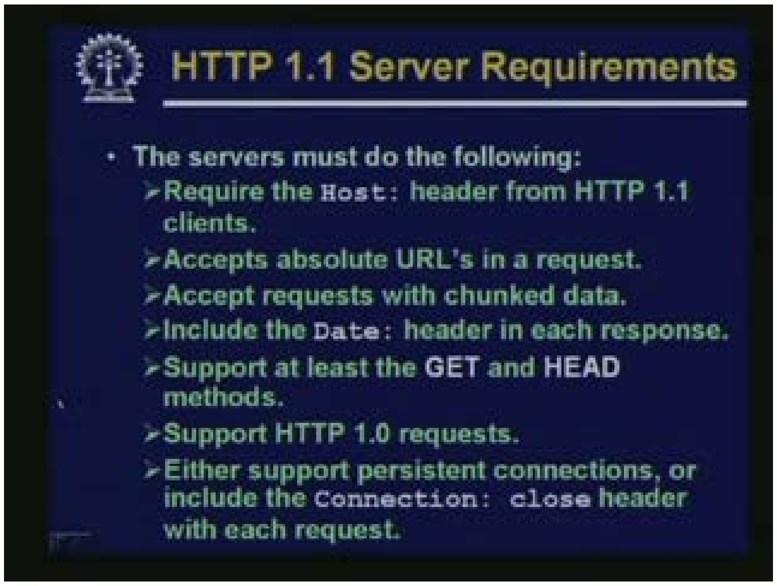


HTTP 1.1 Client Requirements

- The clients must do the following:
  - Include the `Host:` header with each request.
  - Either support persistent connections, or include the `Connection: close` header with each request.
  - Handle the `100 Continue` response.
  - Accept responses with *chunked data*.

So just to have a quick look at what are different features you need to have. Suppose you are going ahead with the design of the http "1.1" client. So what are the new requirements you must have? First as I said the explicit host identification is required you must include the host header with each request. Well persistent connection is supported by default. But if you want to have the other stateless transaction then you must have this connection close option specified in the header explicitly. There are some other features also I am not going to detail. But you can have a look at the http documentation and try to find out what these are. There are 100 continued responses and there is something called chunked data which is also sometimes useful when you are sending a number of different things together. So these are the kind of supports you must have from the client if you want to have a http "1.1" compatible client which can faithfully talk to a http server and communicate.

(Refer Slide Time: 38:45)

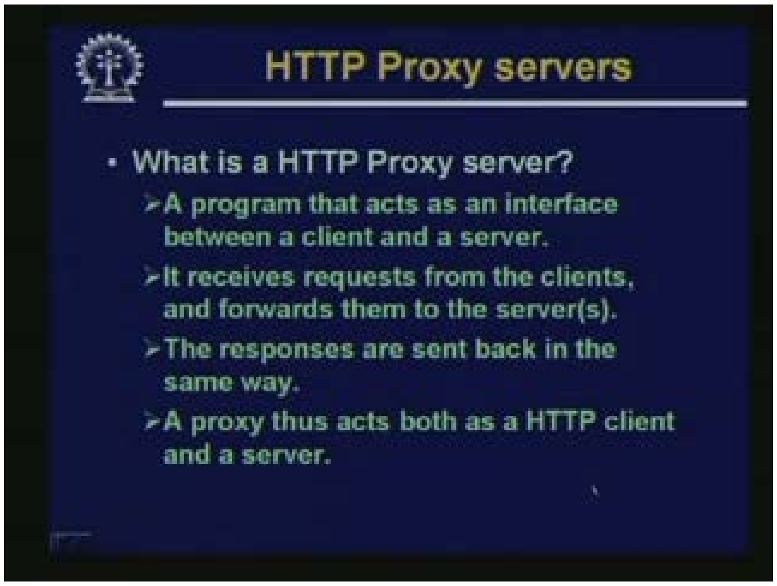


HTTP 1.1 Server Requirements

- The servers must do the following:
  - Require the `Host:` header from HTTP 1.1 clients.
  - Accepts absolute URL's in a request.
  - Accept requests with chunked data.
  - Include the `Date:` header in each response.
  - Support at least the GET and HEAD methods.
  - Support HTTP 1.0 requests.
  - Either support persistent connections, or include the `Connection: close` header with each request.

Similarly from the point of view of the server, there are a few thing the server must also do like it will require a host header from the client. It should accept absolute URLs in a request because most of the earlier servers they only accepted realative URLs. See absolute URL meaning you start from the root http slash slash host name slash path the complete path. That is absolute URL then accept request with chunked data. This again you can have a look what is chunked data include the date header in each response. This is the additional thing support at least get and head mandatory. But others you can keep depending on the environment in which you want to use this and in order to have backward compatibility you should also support “1.0” http requests and either support persistent connections or include the connection close header request explicitly in the request. So if due to some reason persistent connection cannot be supported by the server then connection colon close, this must explicitly be present in the headers of every response you generate. So that the clients will know well this particular server cannot support persistent. So let me go for non-persistent.

(Refer Slide Time: 40:19)



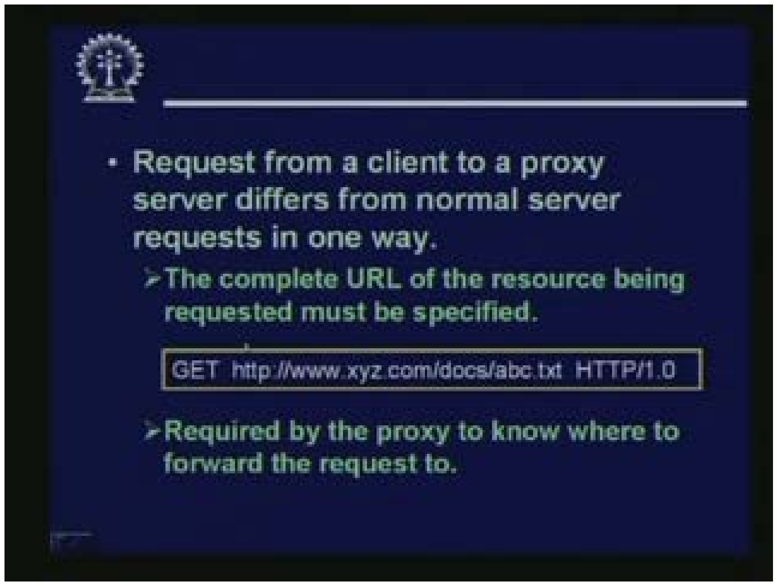
HTTP Proxy servers

- What is a HTTP Proxy server?
  - A program that acts as an interface between a client and a server.
  - It receives requests from the clients, and forwards them to the server(s).
  - The responses are sent back in the same way.
  - A proxy thus acts both as a HTTP client and a server.

So talking about http proxy server we will be talking about proxy servers in more detail later. Proxy server is basically an interface between the client and a server. Say here you have a proxy server you the client out here you have the server out here. So the client sends the request to the proxy server. Proxy server sends the request to the server. The server when it sends back the request the request goes to the proxy. The proxy sends the request back to the client. So proxy works more like a middle man; proxy sits in between. But what does actually proxies do? Well the proxy has a number of other features that we shall talk about later. It can provide some kind of authentication. It can provide some kind of filtering, some kind of simple fire wall configuration. It can implement there a few other things. The proxy server will receive request from the client on one side and it will forward them to the server on the other side.

Similarly when the response comes back responses they come back from the server and they are forwarded to the client. So you can say the proxy at the same time is acting both as http client as well as http server. So whenever it is receiving request from the client. From one side it is receiving request from the client it is acting as the server. On the other side it is forwarding that same request to the web server at that time it is acting as a client. So I just mentioned proxy server here because proxy server is an example where you need to have both the functionalities of http client and server in the same software package software tool or software program. So the details of proxy server we shall see later but the point I wanted to emphasize is this that, this is one example of a program where if you want to code you must have both the functionalities of the server and client in it.

(Refer Slide Time: 42:40)



- Request from a client to a proxy server differs from normal server requests in one way.
  - The complete URL of the resource being requested must be specified.

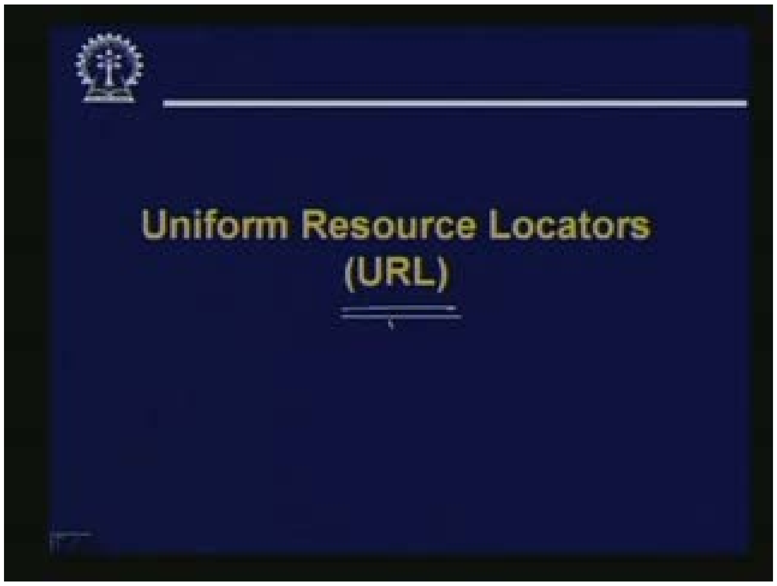
```
GET http://www.xyz.com/docs/abc.txt HTTP/1.0
```

  - Required by the proxy to know where to forward the request to.

So another thing about proxy server is that the request from a client to proxy server is a little different from a normal server. Why? Because in a normal web server what you do? You establish a connection, then you specify the path name of a document. So the server knows that well a connection will already established with me, I have this path. So I can locate the particular file on this path and I can send it back to the requesting client. But in a proxy server the situation is different. Here every request must contain the complete path along with the name of the machine which means, the complete URL. Well I will come back to this URL shortly. Complete URL means complete path starting from the name of the machine along with the document. Say for example www xyz com slash doc, slash abc dot text.

Because the proxy server will have to forward this request to some other server. If it does not know which server the request is going to how will you do that. If it was just a relative URL relative path name then it would not know where to forward the request to. So that is why a request coming to a proxy server must be coming in a complete way. So that you will see in most of the browsers there is a facility for specifying proxy server. You can specify that request from your browser will go through a proxy server is specify the name or the ip address on. And the port number in which the proxy server is working the why the browser will behave is that whenever it is trying to send something to the proxy server. It will send the complete path name to the proxy server so that it can handle.

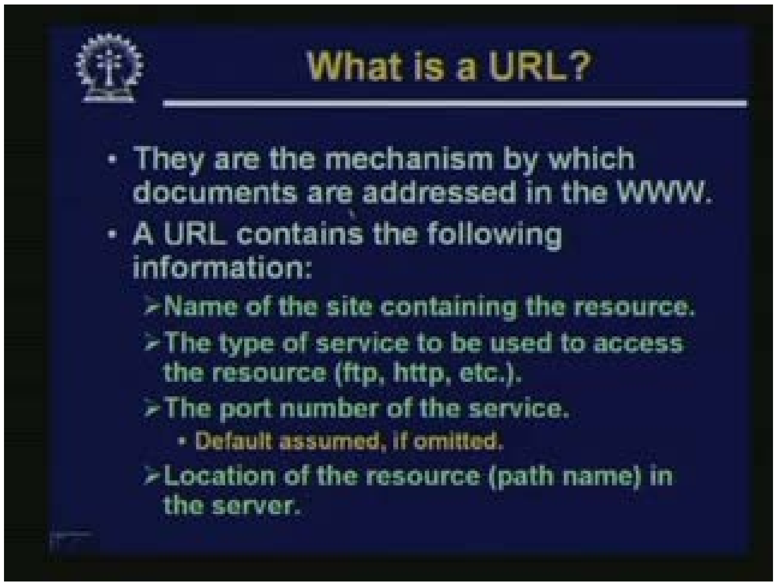
(Refer Slide Time: 44:43)



Uniform Resource Locators (URL)

Now let us finally talk about the Uniform Resource Locator or URL. See URL are nothing but mechanisms to locate some resource on the internet. In the simple case I want to locate that particular file in the wed server that is the example of URL. URL is the short form for Uniform Resource Locator.

(Refer Slide Time: 45:08)

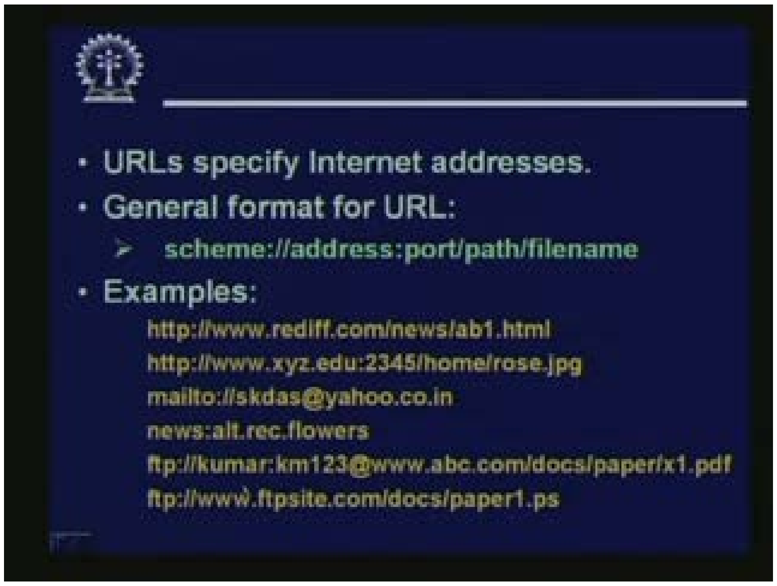


What is a URL?

- They are the mechanism by which documents are addressed in the WWW.
- A URL contains the following information:
  - Name of the site containing the resource.
  - The type of service to be used to access the resource (ftp, http, etc.).
  - The port number of the service.
    - Default assumed, if omitted.
  - Location of the resource (path name) in the server.

So as I mentioned, this is this URL is a mechanism or it is a format whatever you call by which documents or resources can be addressed in the internet in the World Wide Web. A URL contains several information. First of course the name or the address of the site or the server where the actual resource you are trying to address is located. Secondly what is the type of service you want to use to access the resource? Do you want to use Http. So you want to use http or something else. Port number of the service well in most cases some default port number is assumed. But you can explicitly specify a port number also and once you have look at it or specified everything. You may also specify a path name on the server which specifies where the resource is located on that particular server. So all this things taken together will actually identify where the server will actual where the particular resource you are trying to address is actually located on the server.

(Refer Slide Time: 46:30)



- URLs specify Internet addresses.
- General format for URL:
  - scheme://address:port/path/filename
- Examples:

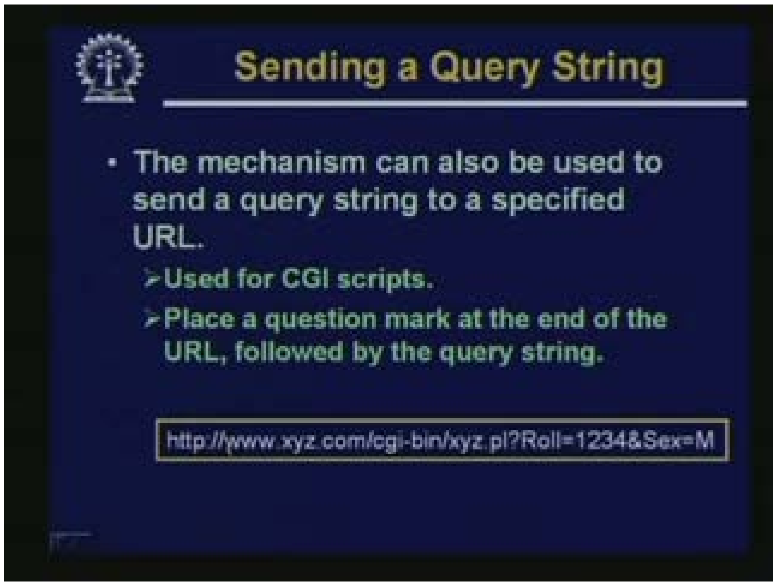
  http://www.rediff.com/news/ab1.html
  http://www.xyz.edu:2345/home/rose.jpg
  mailto://skdas@yahoo.co.in
  news:alt.rec.flowers
  ftp://kumar:km123@www.abc.com/docs/paper/x1.pdf
  ftp://www.ftpsite.com/docs/paper1.ps

Now let us see what are the different URL types, I will list out the example that we are going to detail. These URLs are nothing but a form of specifying internet addresses. So the whatever you have specified, it starts with a scheme this is the syntax scheme which access method you are using followed by a colon double slash where there is one example, where you do not need the double slash followed by the address of the machine. This colon port is optional if you want to specify an explicit port number give colon port followed by a path name some examples follow.

The first example is a URL which says it is an http. This is the name of the server and this is the path of the resource the resource is ab1.html. The second one is also http, but it specify an explicit port number here the http server is running on port number 2345. The resource is rose.jpg in this path. The third is an example of an email mail to at the end you have the email address. Well if you have this mail to URL as a part of a web page. If you click on it automatically a mail window will open where you can type in a mail. This is the reason why mail to URL is required.

NEWS is one example where double slash is not required. This refers to NEWS groups FTP is for file transfer in the first case after the slash, you give the user name colon; you give the password, then this at the rate sign. Then the machine where you want to do a FTP then this the path name where the path name or resource is located. The last example refers to anonymous ftp. But user name and password are not given so if the user name and password are missing then by default it is taken to the anonymous FTP and this is the address of a machine and this is the address of the document or resource under the anonymous directory.

(Refer Slide Time: 48:53)



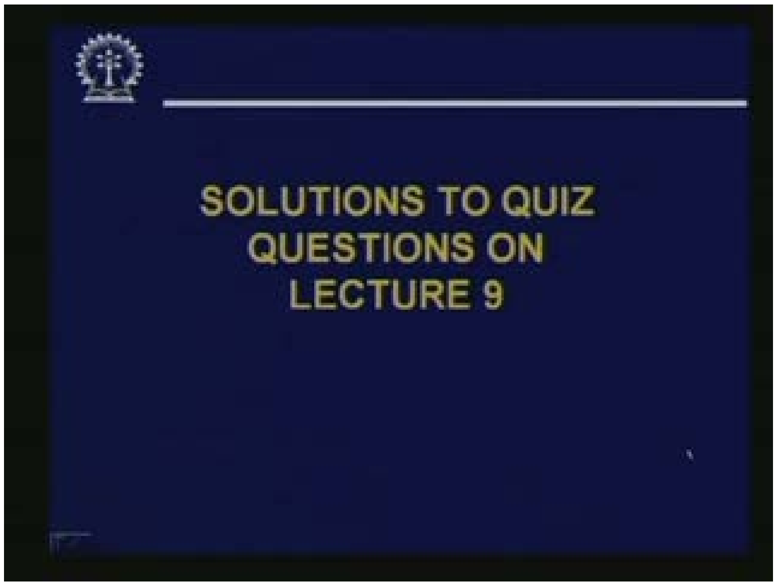
Sending a Query String

- The mechanism can also be used to send a query string to a specified URL.
  - Used for CGI scripts.
  - Place a question mark at the end of the URL, followed by the query string.

```
http://www.xyz.com/cgi-bin/xyz.pl?Roll=1234&Sex=M
```

And this URL can also be used to send a send query string like the example that we had given earlier for GET. Well GET is a way to send it as a as a http string. But even when you type a URL on the browser there also you can explicitly specify that. Like an example follows, it says http. This is the name of the machine; this is the path name of the resource. Here this path name refers to a cgi program again a question mark followed by some additional information. Now this whole thing is a URL. This you can type on a browser and press enter. What the browser will do it will generate a get http request from this and it will send it to the requested server. So actually what will go to the server will be the same GET request as I mentioned earlier. But this how you can also specify in an URL as part of http. So with this we come to the end of first part of our discussion on World Wide Web. Now let us quickly have a look at the answers to the questions that where POST in our last class.

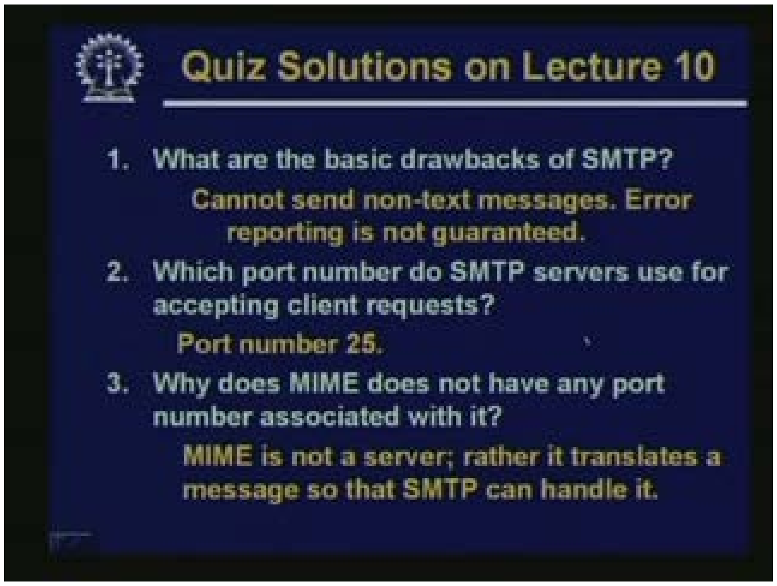
(Refer Slide Time: 50:16)



SOLUTIONS TO QUIZ QUESTIONS ON LECTURE 9

So listen to the quiz questions on lecture 9.

(Refer Slide Time: 50:23)



Quiz Solutions on Lecture 10

1. What are the basic drawbacks of SMTP?
   Cannot send non-text messages. Error reporting is not guaranteed.
2. Which port number do SMTP servers use for accepting client requests?
   Port number 25.
3. Why does MIME does not have any port number associated with it?
   MIME is not a server; rather it translates a message so that SMTP can handle it.

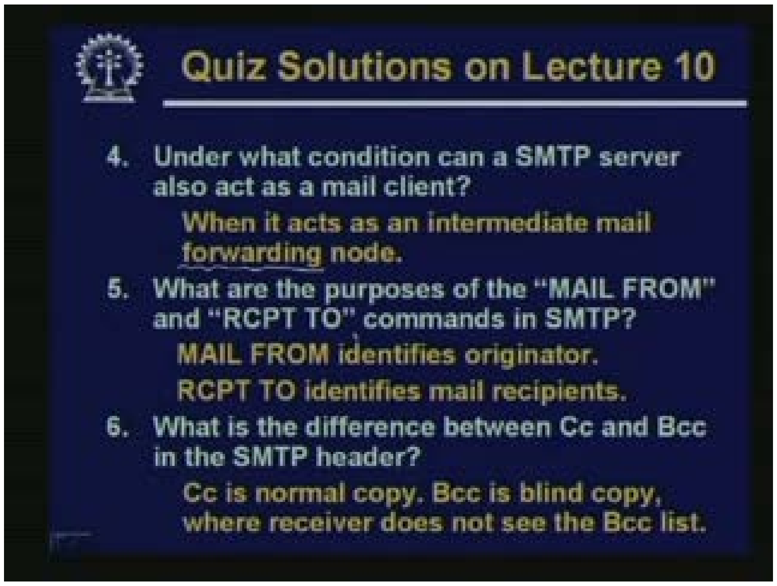
What are the basic drawbacks of SMTP?

Well in SMTP the main drawback is that, this SMTP cannot send non text messages. For this purpose you need MIME and error reporting is not guaranteed. These are the basic drawbacks.

Number two. Which port number does SMTP use for accepting client requests?
This is port number 25.

Why does MIME does not have port number associated with it?
Well as I said MIME is not a server what happens is that suppose this MIME, this is your SMTP. So MIME actually translates a document in to an alternate form and gives it to SMTP. It is the SMTP server which actually sending the mail and it is SMTP which is acting which is residing as a server program. This MIME is not a server it is a simple translator.

(Refer Slide Time: 51:28)



Quiz Solutions on Lecture 10

4. Under what condition can a SMTP server also act as a mail client?
   When it acts as an intermediate mail forwarding node.
5. What are the purposes of the "MAIL FROM" and "RCPT TO" commands in SMTP?
   MAIL FROM identifies originator.
   RCPT TO identifies mail recipients.
6. What is the difference between Cc and Bcc in the SMTP header?
   Cc is normal copy. Bcc is blind copy, where receiver does not see the Bcc list.

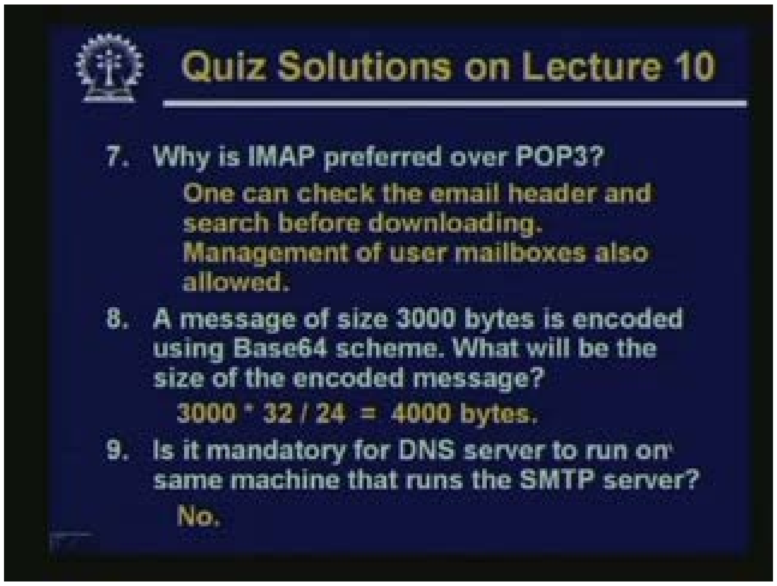
So under what condition can SMTP server also act as a mail client?
Just I mentioned when an intermediate SMTP node is doing mail forwarding receiving a mail from one server and forwarding it to another then it is simultaneously acting as mail receiver and mail sender both client and server.

What are the purpose of mail from and receipt to commands in SMTP?
Mail from will identify who is sending the mail receipt to will identify to whom you are sending the mails.

What is the difference between cc and bcc in the SMTP header?
See cc and bcc both are used to send copies of the mail to some other people. For example your primary destination may be somebody you are sending copies of the mail to someone else. If you use cc suppose the mail was destined to me when I receive the mail I can also see that well this mail was also copied to these fellows. But if you specify

those copies using bcc or blind carbon copies, cc means carbon copy. Bcc means blind carbon copy then the final recipient will not know that to whom the copies were sent to. That is why it is blind.

(Refer Slide Time: 52:52)



Quiz Solutions on Lecture 10

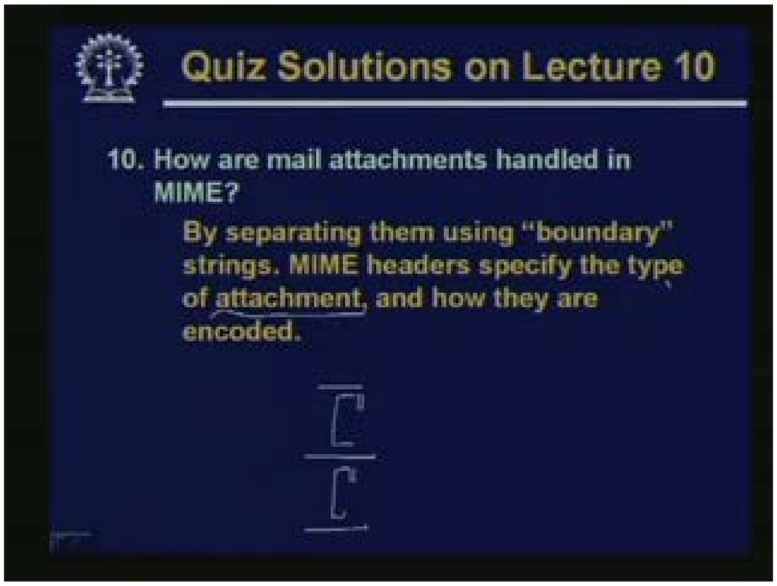
7. Why is IMAP preferred over POP3?
   One can check the email header and search before downloading. Management of user mailboxes also allowed.
8. A message of size 3000 bytes is encoded using Base64 scheme. What will be the size of the encoded message?
   3000 * 32 / 24 = 4000 bytes.
9. Is it mandatory for DNS server to run on same machine that runs the SMTP server?
   No.

Why is IMAP preferred over POP3?
Because in IMAP there are some additional features like you can check the email headers you can search the mail contents you can manage the mail box these are the main extra features.

A message has 3000 bytes you are encoding it using base 64. What will be the size of the encoded message?
Now in base 64 if you recall every 24 bit of the message gets encoded in 32 bits. So the total size will be 4000 bytes.

Is it mandatory for DNS server to run on same machine as SMTP?
No. DNS can run on any other machine whenever you configure your computer you have to specify the address of the DNS server. So whenever there is a name involved the DNS server will be automatically queried and the ip address is obtained from there.

(Refer Slide Time: 53:48)



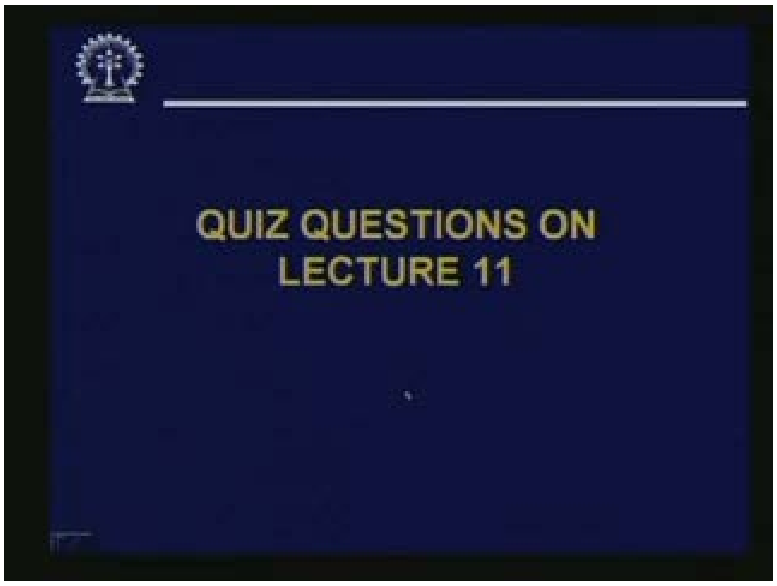
Quiz Solutions on Lecture 10

10. How are mail attachments handled in MIME?

    By separating them using "boundary" strings. MIME headers specify the type of attachment, and how they are encoded.

How are mail attachments maintained in MIME?
Mail attachments are separated by using the boundary strings. MIME headers will be associated with each such each attachment. For example mail has two attachments. There will be two such attachments with appropriate boundary headers. And in each such attachment there will be additional header section specified what is the type of attachment where there is an image this is the audio file, video file what it is and how they are encoded is the encoding system base 64 or something else.

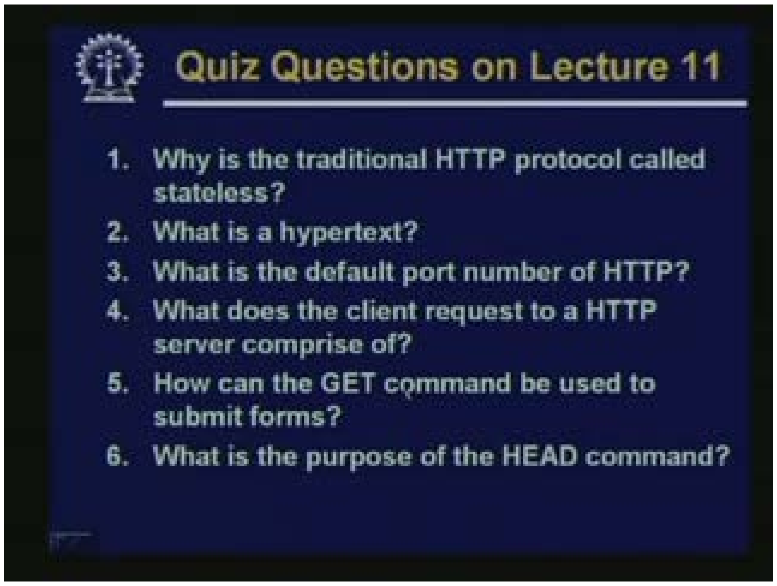
(Refer Slide Time: 54:28)



QUIZ QUESTIONS ON LECTURE 11

Now some questions from today's lecture.

(Refer Slide Time: 54:31)

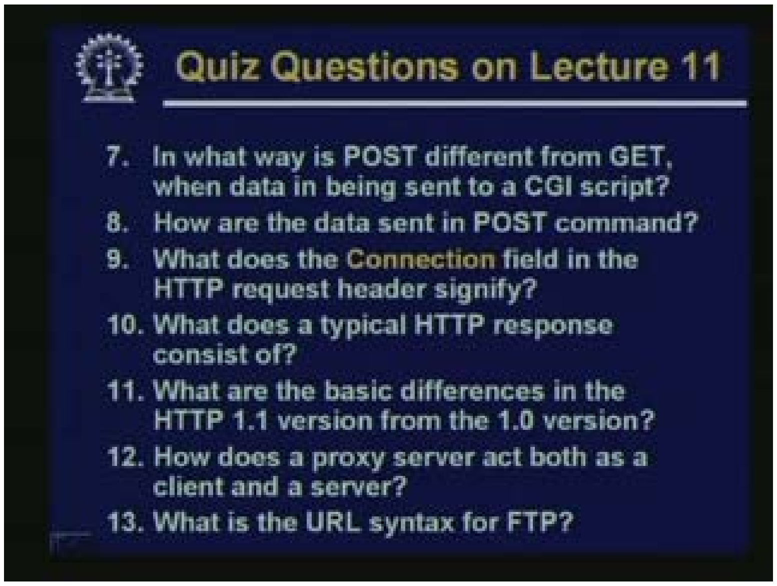


Quiz Questions on Lecture 11

1. Why is the traditional HTTP protocol called stateless?
2. What is a hypertext?
3. What is the default port number of HTTP?
4. What does the client request to a HTTP server comprise of?
5. How can the GET command be used to submit forms?
6. What is the purpose of the HEAD command?

Why is the traditional http protocol called stateless?
What is a hypertext?
What is the default port number of http?
What does the client request to http server comprise of?
How can the get command be used to submit forms?
What is the purpose of the head command?

(Refer Slide Time: 54:59)



Quiz Questions on Lecture 11

7. In what way is POST different from GET, when data in being sent to a CGI script?
8. How are the data sent in POST command?
9. What does the Connection field in the HTTP request header signify?
10. What does a typical HTTP response consist of?
11. What are the basic differences in the HTTP 1.1 version from the 1.0 version?
12. How does a proxy server act both as a client and a server?
13. What is the URL syntax for FTP?

In what way is POST different from get when data is being sent to a cgi script?

How are the data sent in POST command?
What are the connection field in the http request header signify?
What does a typical http response consists of?
What are the basic differences in the http “1.1” version from the “1.0” version?
How does a proxy server act both as a client and a server?
What is the URL syntax for ftp?

So with this we stop today’s lecture. In our next lecture we shall be talking about some other aspect of World Wide Web subsequent to which we will be talking about how we can design web pages using various technologies like HTML, ASP and so on. Thank you.