WKW SCHOOL

2020

CI6206 Internet Programming

Request & Response
HTTP (GET/POST)

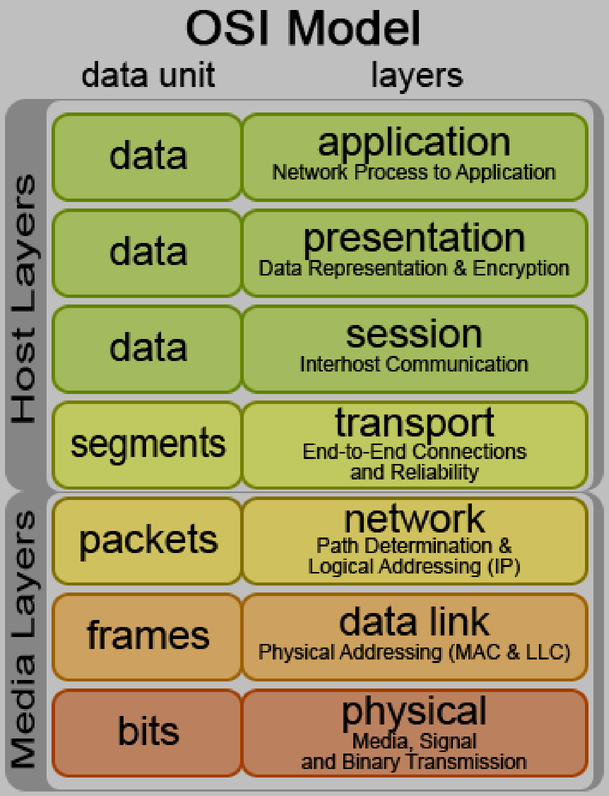
Wong Twee Wee

Ver1.6

HTTP

- Browsers interacts with web servers using the HTTP.
- The HTTP is based on a request-response model.
  - The client (your browser) requests data from a web application that resides on a physical machine.
  - The web application in turn responds to the request with the data your browser requested.

OSI 7-LAYER MODEL


OSI Model
data unit
layers
Host Layers
Media Layers
data
application
Network Process to Application
data
presentation
Data Representation & Encryption
data
session
Interhost Communication
segments
transport
End-to-End Connections and Reliability
packets
network
Path Determination & Logical Addressing (IP)
frames
data link
Physical Addressing (MAC & LLC)
bits
physical
Media, Signal and Binary Transmission

Source : wikimedia

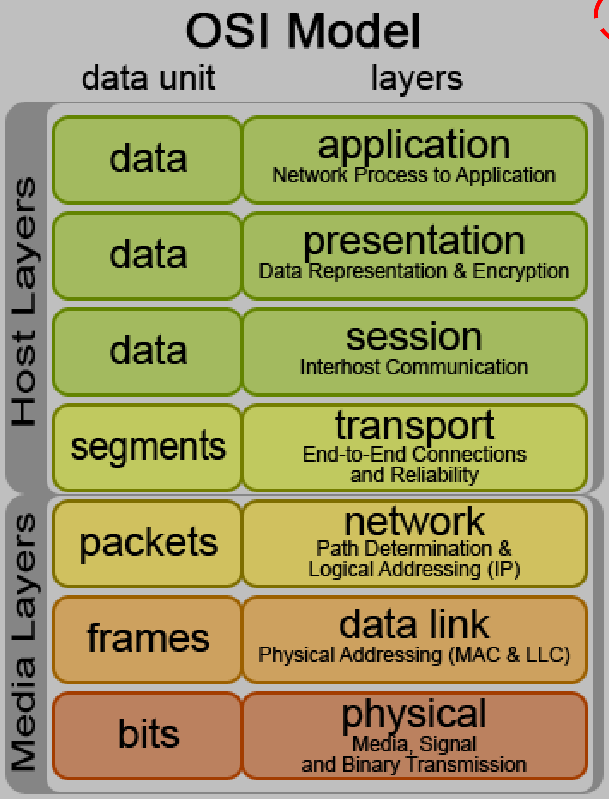
Open Systems Interconnection model

- developed by the ISO (International Organization for Standardization) in 1984 provides an abstract model of networking
- divides the tasks involved in moving information between networked computers into 7 task groups each task group is assigned a layer

Each layer is reasonably self-contained

- can be implemented independently changes/updates to a layer need not effect other layers

PROTOCOL LAYERS

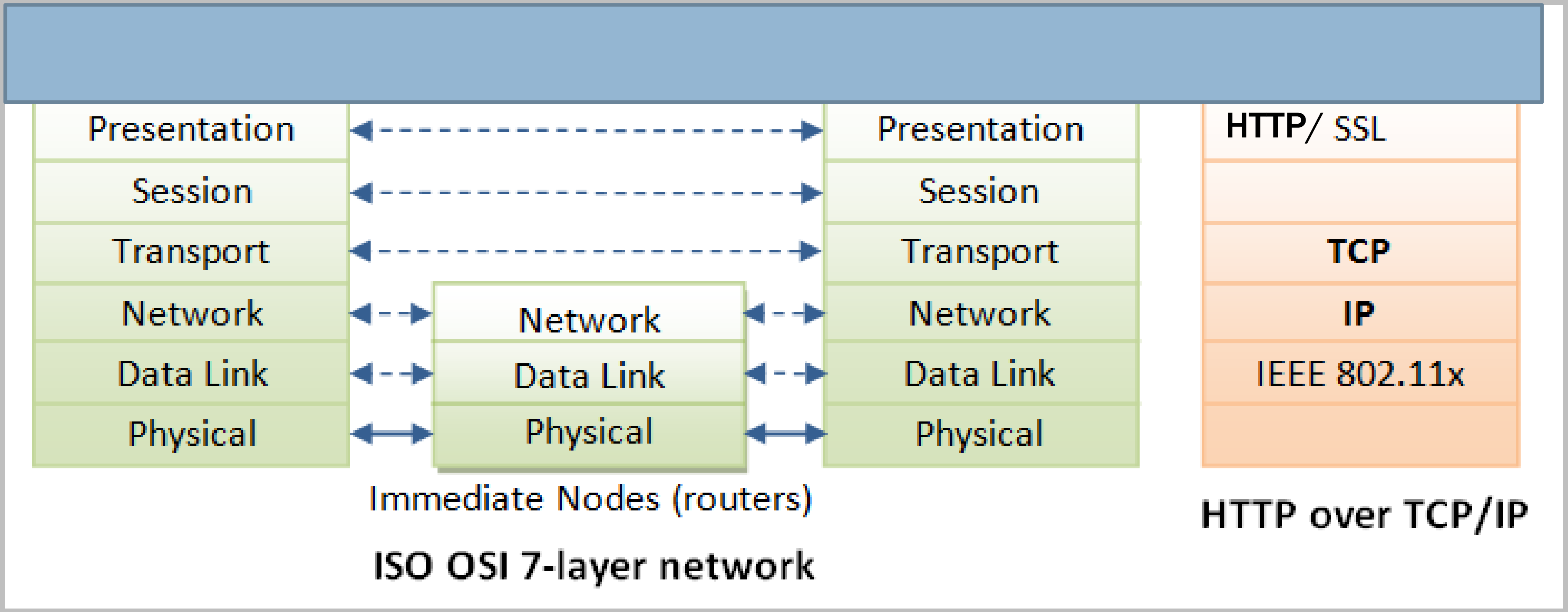


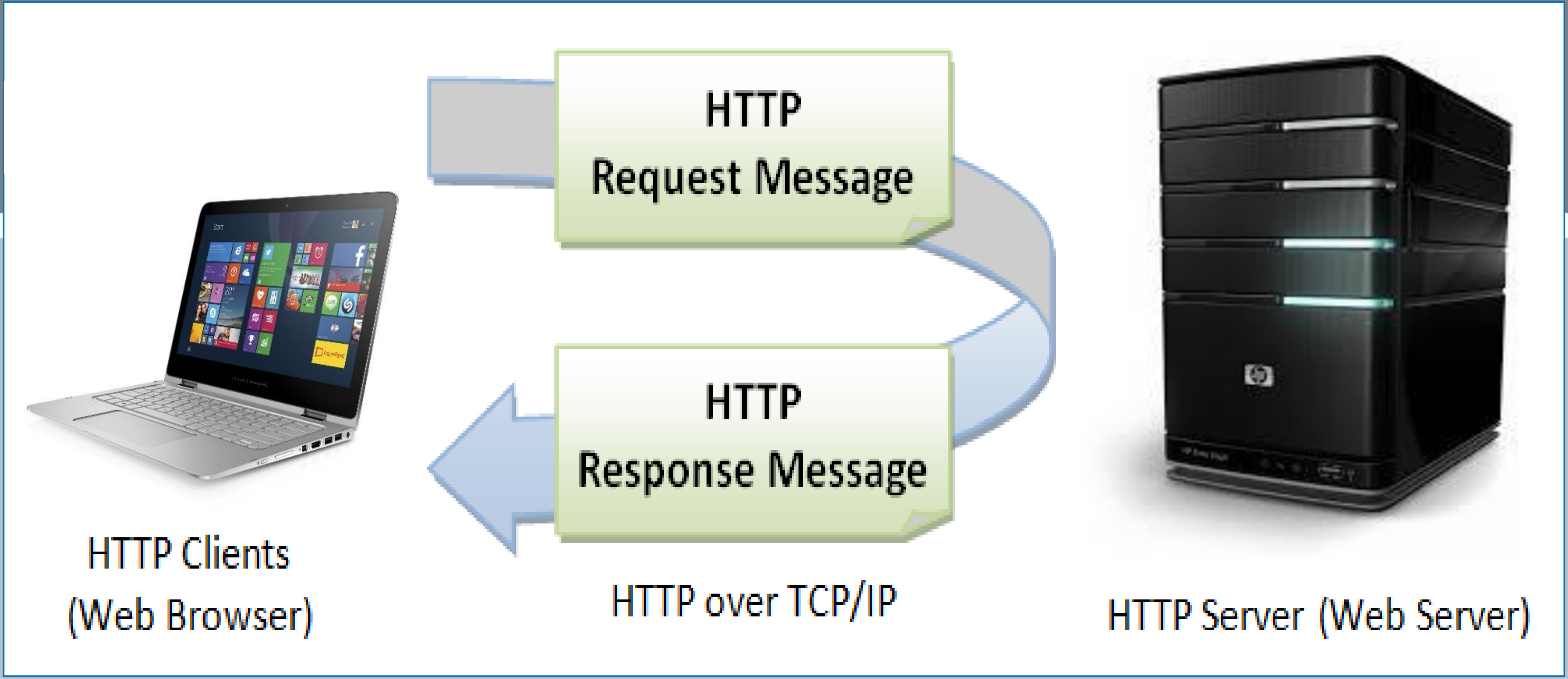
OSI Model
data unit
layers
Host Layers
Media Layers
data
application
Network Process to Application
data
presentation
Data Representation & Encryption
data
session
Interhost Communication
segments
transport
End-to-End Connections and Reliability
packets
network
Path Determination & Logical Addressing (IP)
frames
data link
Physical Addressing (MAC & LLC)
bits
physical
Media, Signal and Binary Transmission

Source : wikimedia

- **Application layer**
  - **describes how applications will communicate**
    e.g., HTTP, FTP, Telnet, SMTP
- Presentation layer
  - describes the form of data being transferred & ensures that it will be readable by receiver
    e.g., floating point formats, data compression, encryption
- Session layer
  - describes the organization of large data sequences & manages communication session
    e.g., coordinates requests/responses ("traffic flow")
- Transport layer
  - describes the quality and nature of data delivery
    e.g., how retransmissions are used to ensure delivery
- Network layer
  - describes how a series of exchanges over various data links can deliver data across a network
    e.g., addressing and routing
- Data Link layer
  - describes the logical organization of data bits transmitted on a particular medium
    e.g., frame sequencing, error notification
- Physical layer
  - describes the physical & electrical properties of the communications media
    e.g., voltage levels, data rates, max distances

HTTP OVER TCP/IP

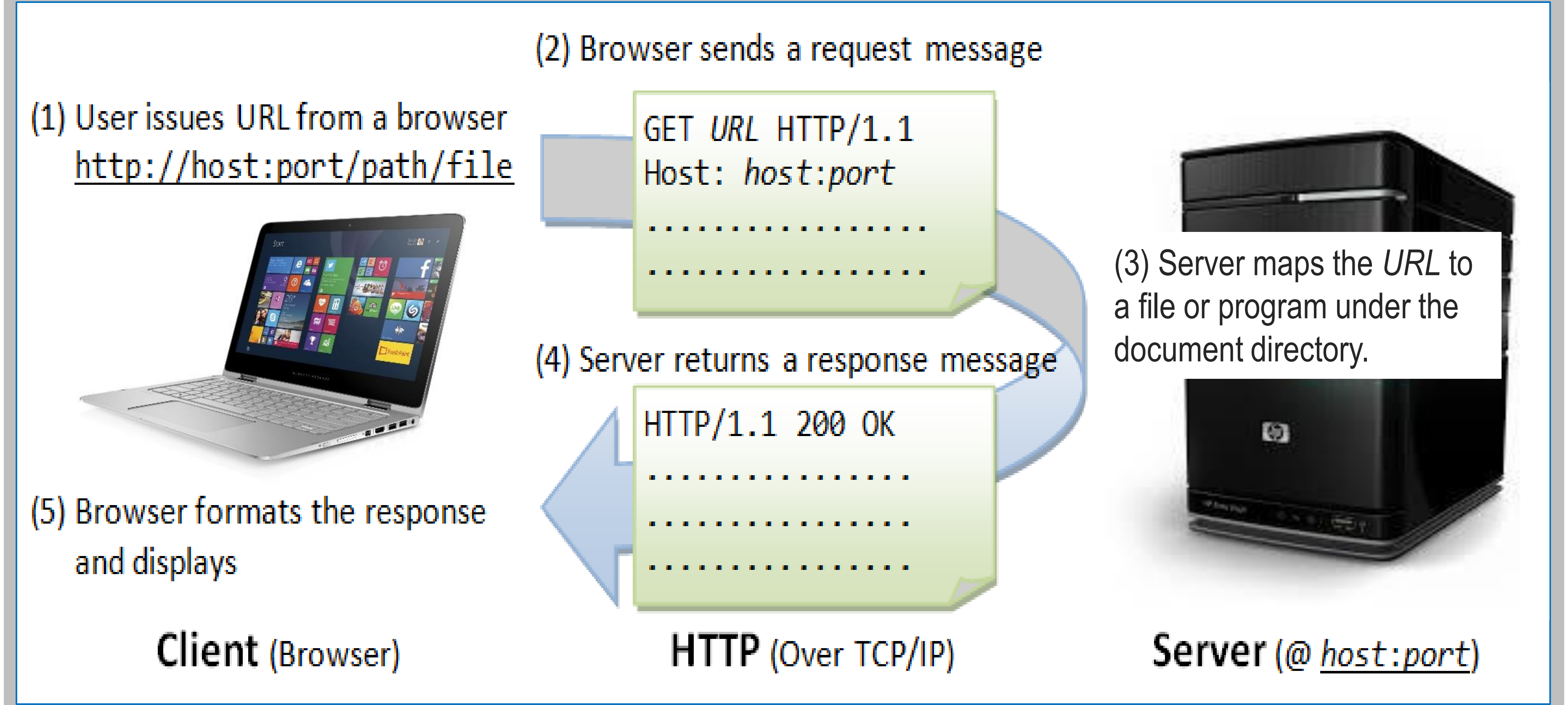




Client Pull or Server Push ?

Quoting from the RFC2616: "The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers."

HTTP://WWW.NTU.EDU.SG:80/INDEX.HTML


(1) User issues URL from a browser
http://host:port/path/file
(2) Browser sends a request message
GET *URL* HTTP/1.1
Host: *host*:*port*
....
....
(3) Server maps the *URL* to a file or program under the document directory.
(4) Server returns a response message
HTTP/1.1 200 OK
....
....
....
(5) Browser formats the response and displays
Client (Browser)
HTTP (Over TCP/IP)
Server (@ *host*:*port*)

URI – UNIVERSAL RESOURCE LOCATOR

- A URL (Uniform Resource Locator) is used to uniquely identify a resource over the web. URL has the following syntax:

  protocol://*hostname*:*port*/*path-and-file-name*
- http://www.aliexpress.com:80/category/100003070/men-clothing.html
- https://www.aliexpress.com:443/category/100003070/men-clothing.html

There are 4 parts in a URL:

- ***Protocol***: The application-level protocol used by the client and server, e.g., HTTP, FTP, and telnet.
- ***Hostname***: The DNS domain name (e.g., www.test101.com) or IP address (e.g., 192.128.1.2) of the server.
- ***Port***: The TCP port number that the server is listening for incoming requests from the clients.
- ***Path-and-file-name***: The name and location of the requested resource, under the server document base directory.

HTTP REQUEST MESSAGES

Request line

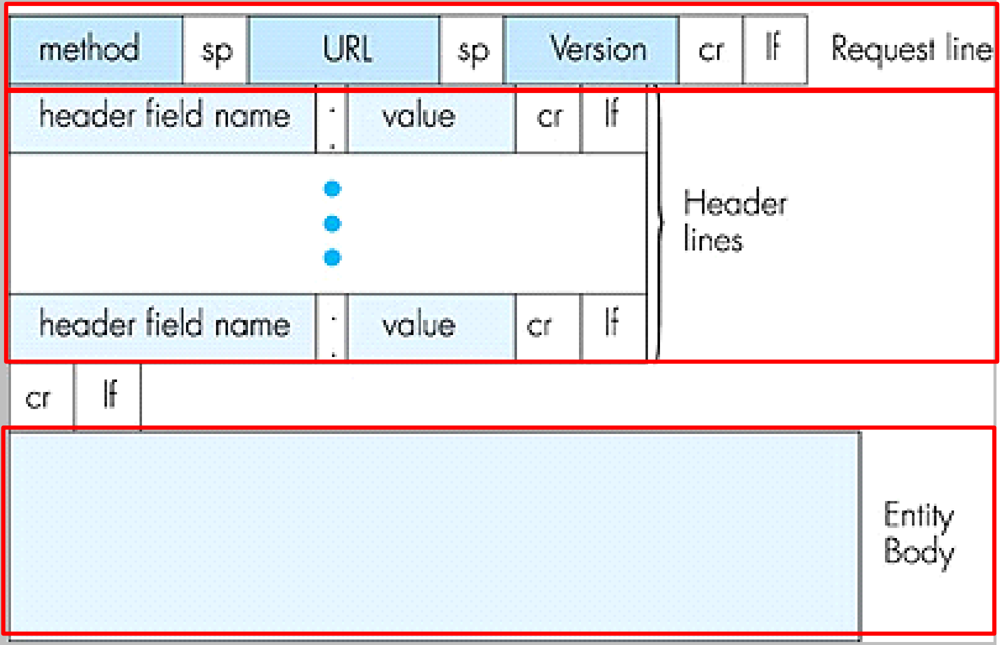
- Method – GET, POST, HEAD, etc.
- URI – absolute path or absolute URL
- HTTP version

Request header

- Parameters/options sent to server
- Information about client

Entity body

- Data sent to the server



HTTP REQUEST METHODS

- GET method
  - Data sent using name/value pair appended to URL.
    - http://ntu.edu.sg/test/MyServlet?name1=value1&name2=value2
    - http://ntu.edu.sg/test/ShoppingCart.php?name1=value1&name2=value2
    - Data can be seen in address bar.
  - GET requests can be cached
  - GET requests remain in the browser history
  - GET requests can be bookmarked
  - GET requests should never be used when dealing with sensitive data
  - GET requests have length restrictions - limit the length of the URL to less than 1KB (2048 chars)
  - GET requests should be used only to retrieve data

HTTP REQUEST METHODS

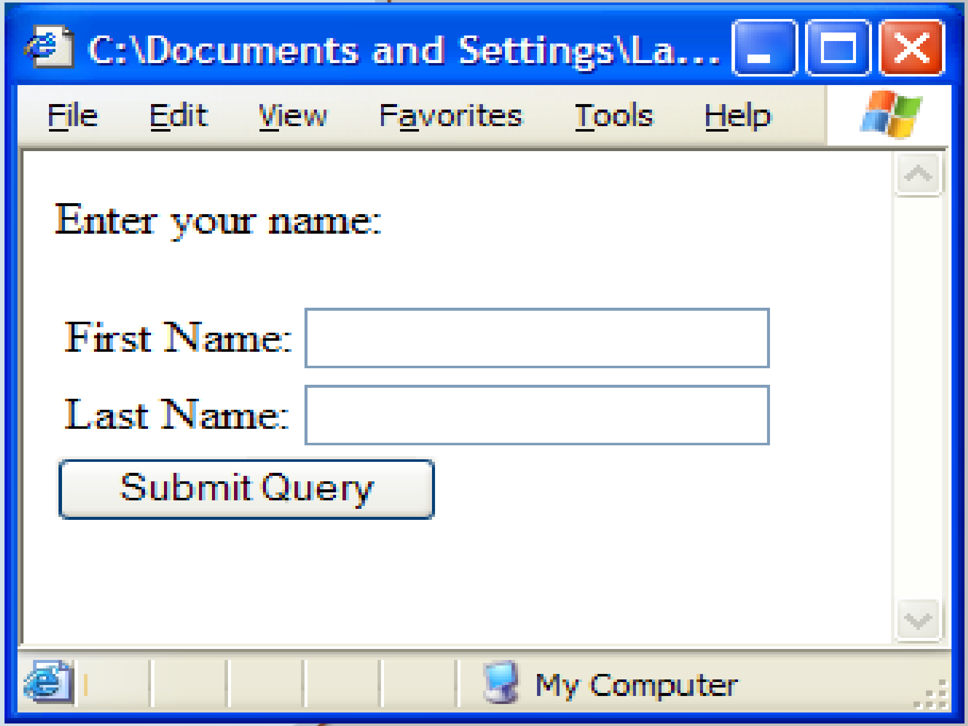
- POST method
  - Data sent as part of the HTTP request entity body.
  - Data hidden from users.
  - POST requests are never cached
  - POST requests do not remain in the browser history
  - POST requests cannot be bookmarked
  - POST requests have no restrictions on data length

  - Examples
    - Providing a block of data (e.g. form submission)
    - Posting a message to a forum, mailing list, etc.

```
POST /test/demo_form.php HTTP/1.1
Host: ntu.edu.sg
name1=value1&name2=value2
```

WEB FORMS (HTML)

```
<html>Enter your name:
<form method="GET" action="servlet/MyServlet">
<table>
<tr>
   <td>First Name:</td>
   <td><input type="text"
name="FirstName"></td>
</tr>
<tr>
   <td>Last Name:</td>
   <td><input type="text"
name="LastName"></td>
</tr>
</table><input type="submit">
</form>
</html>
```


C:\Documents and Settings\La...
File Edit View Favorites Tools Help
Enter your name:
First Name:
Last Name:
Submit Query
My Computer

To learn basics of HTML: https://www.w3schools.com/html/default.asp

GENERATING RESPONSES

- In Java, Servlets is the backend code to return any HTTP response they want.
- Servlet scenarios:
  - Redirecting to another web site/web page.
  - Restricting access to approved users.
  - Specifying content-type other than text/html.
  - Return images instead of HTML.
  - Getting data from Database
  - etc ...

HTTP RESPONSE MESSAGES

Status line

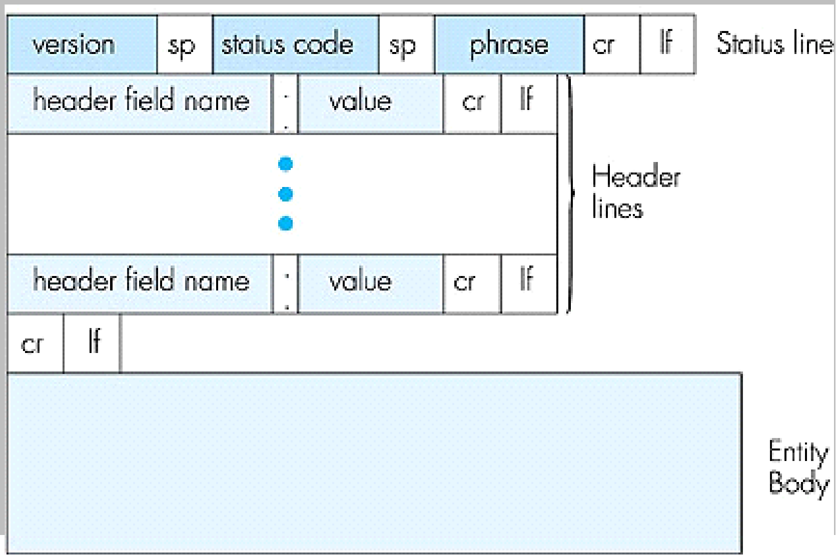
- Protocol version being used
- Status code
- Status message

Five categories
1. Informational
2. Success
3. Redirection
4. Client error
5. Server error

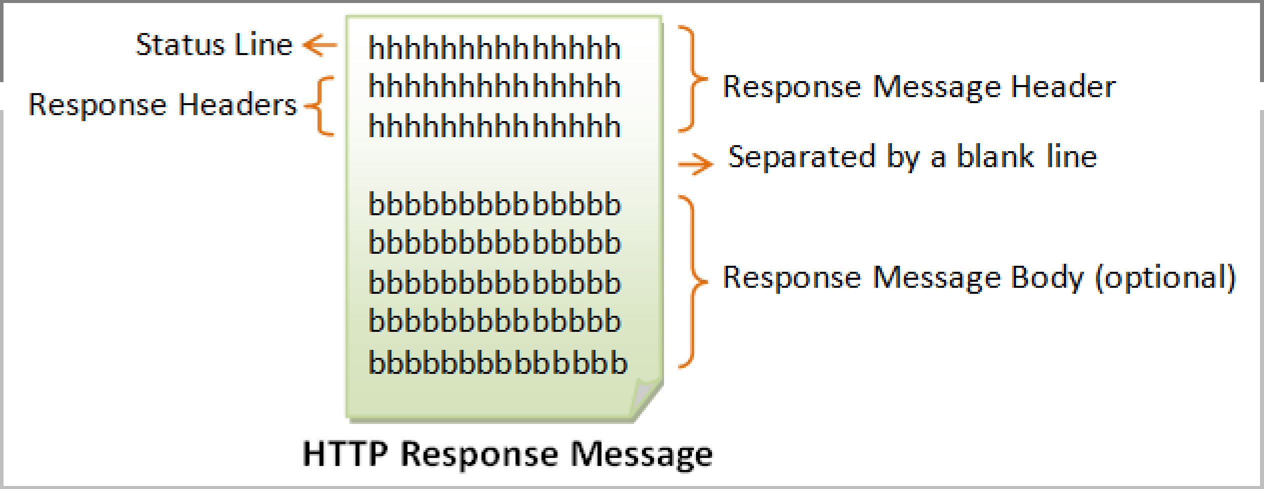
Header lines

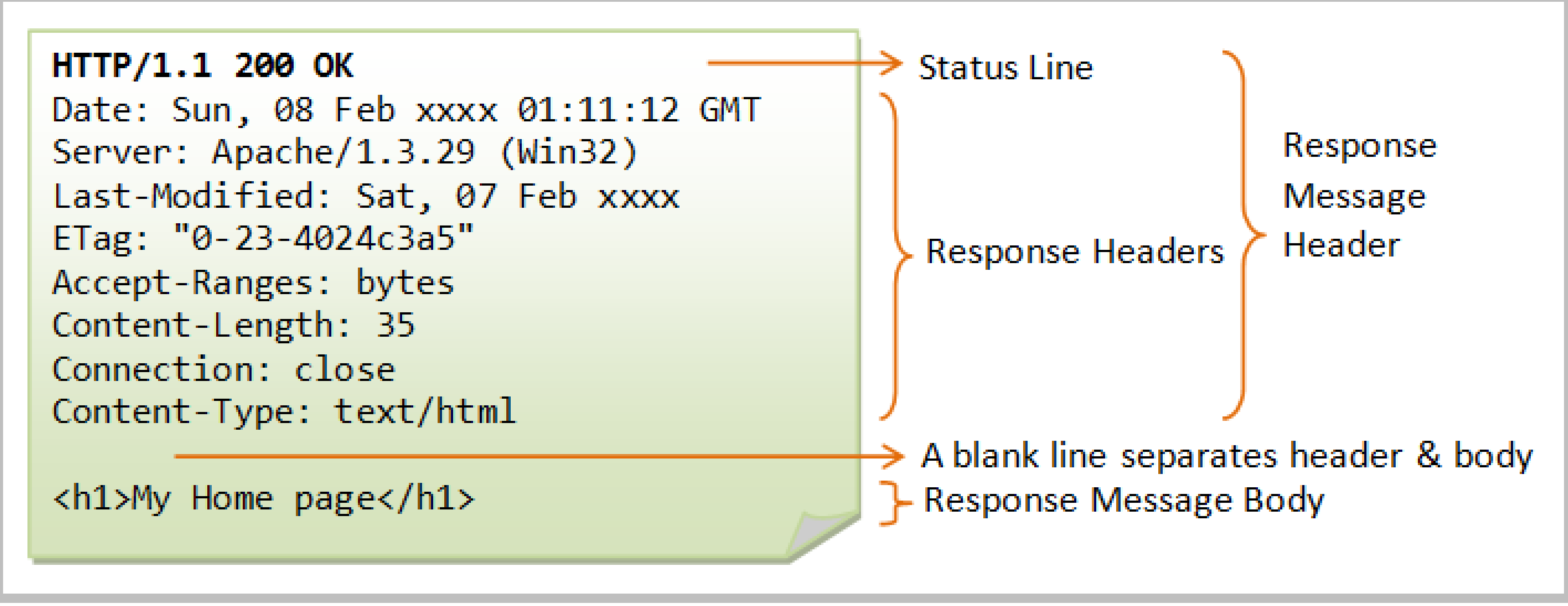
- Information about document and other parameters

- Entity body
  - Contains the document or object



EXAMPLE – RESPONSE MESSAGE





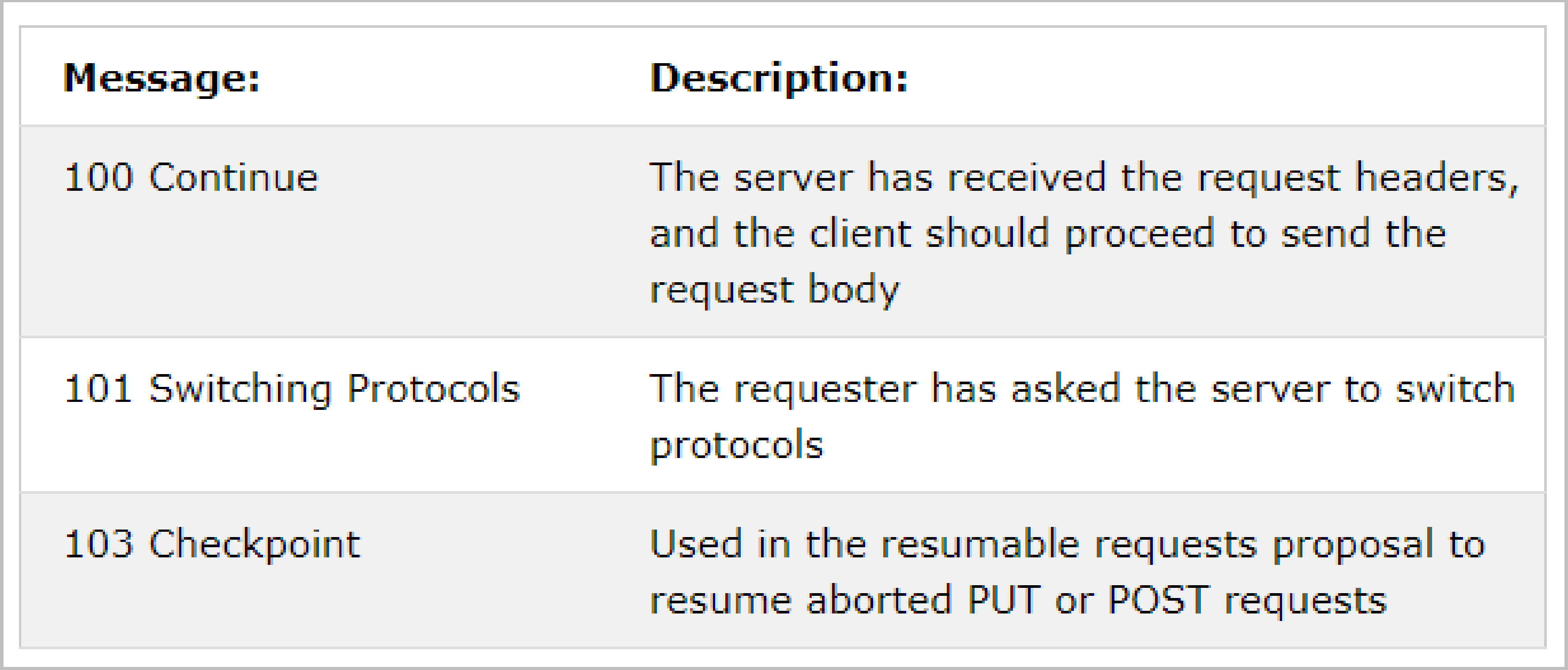
```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

STATUS CODES

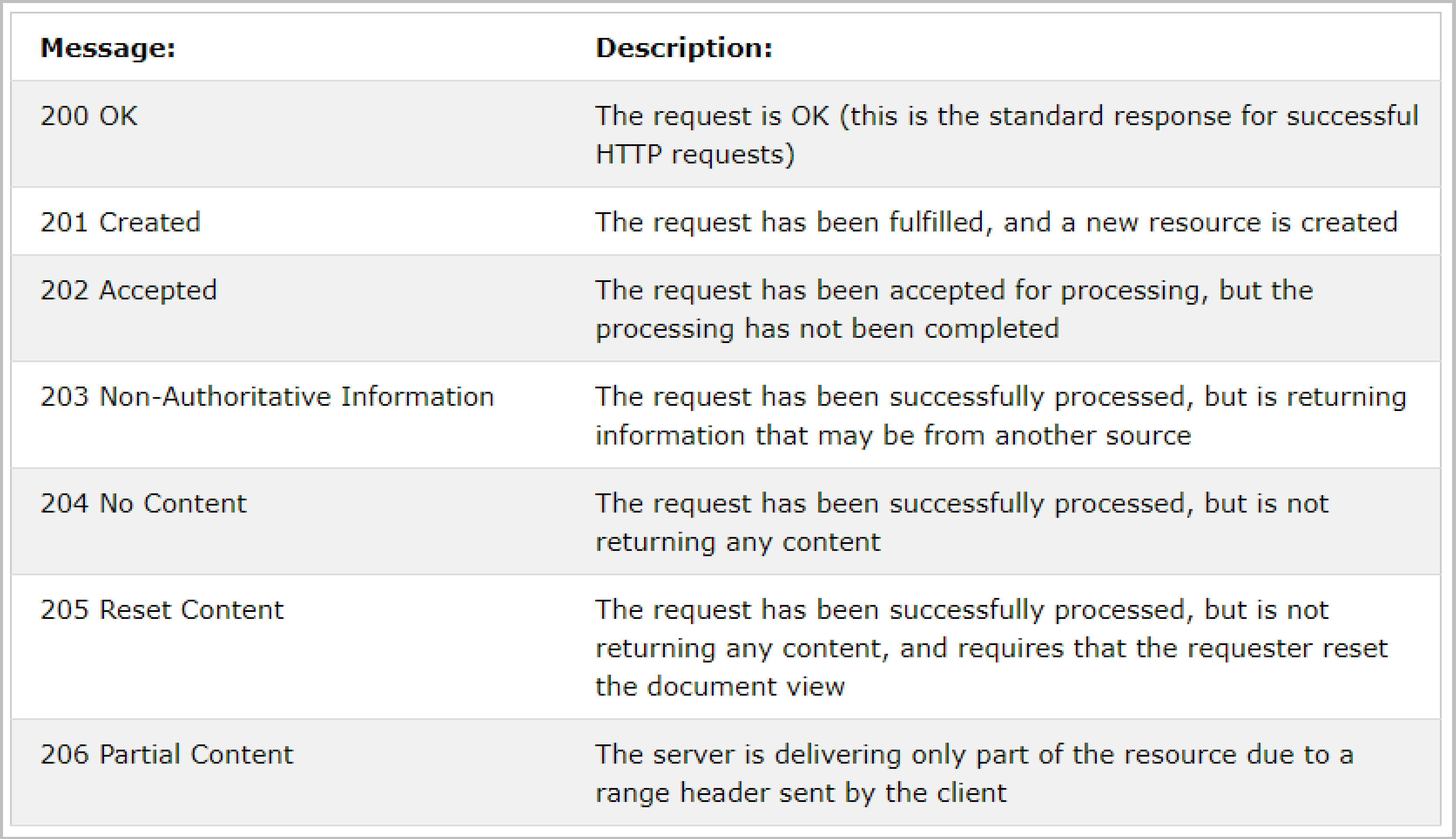
- 200 OK
- 201 created
- 202 accepted
- 204 no content
- 301 moved perm.
- 302 moved temp
- 304 not modified
- 400 bad request
- 401 unauthorized
- 403 forbidden
- 404 not found
- 500 int. server error
- 501 not impl.
- 502 bad gateway
- 503 svc not avail

1XX - INFORMATION



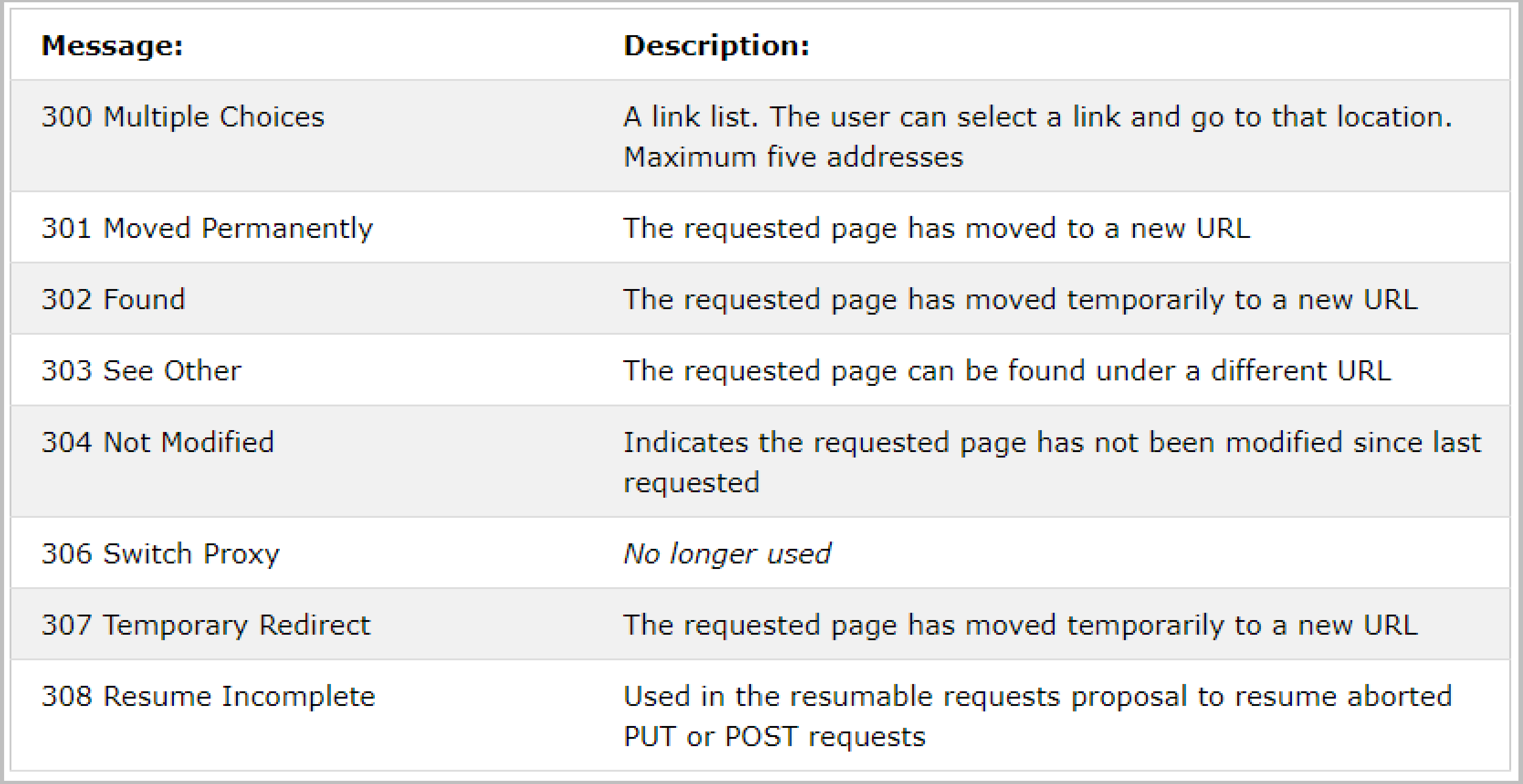
Message:	Description:
100 Continue	The server has received the request headers, and the client should proceed to send the request body
101 Switching Protocols	The requester has asked the server to switch protocols
103 Checkpoint	Used in the resumable requests proposal to resume aborted PUT or POST requests

2XX - SUCCESSFUL



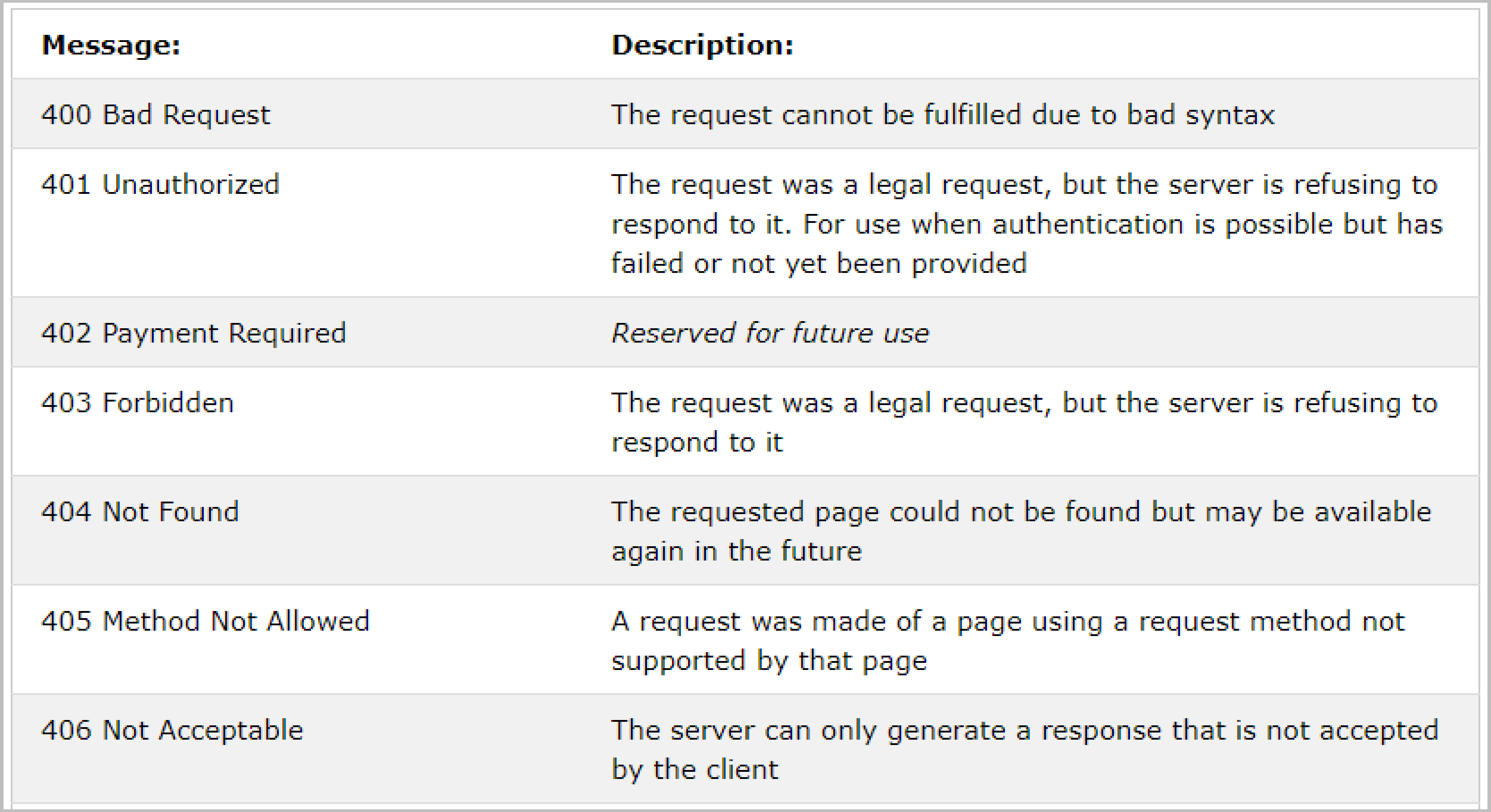
Message:	Description:
200 OK	The request is OK (this is the standard response for successful HTTP requests)
201 Created	The request has been fulfilled, and a new resource is created
202 Accepted	The request has been accepted for processing, but the processing has not been completed
203 Non-Authoritative Information	The request has been successfully processed, but is returning information that may be from another source
204 No Content	The request has been successfully processed, but is not returning any content
205 Reset Content	The request has been successfully processed, but is not returning any content, and 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

3XX - REDIRECTION



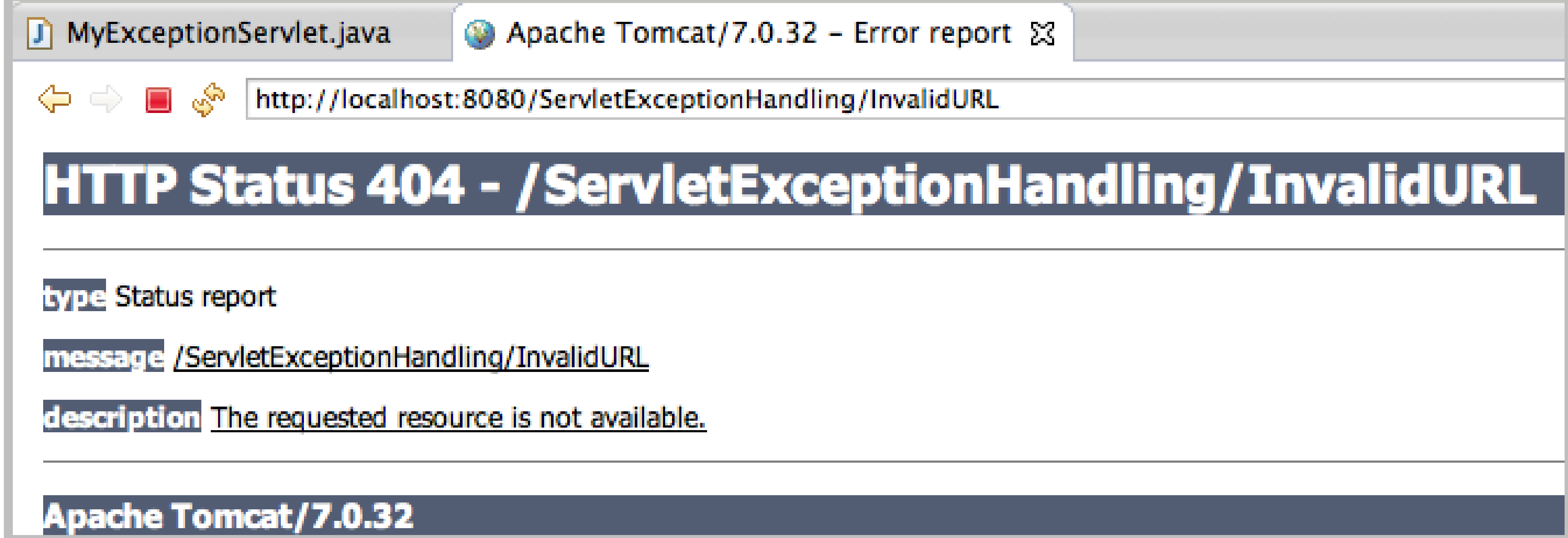
Message:	Description:
300 Multiple Choices	A link list. The user can select a link and go to that location. Maximum five addresses
301 Moved Permanently	The requested page has moved to a new URL
302 Found	The requested page has moved temporarily to a new URL
303 See Other	The requested page can be found under a different URL
304 Not Modified	Indicates the requested page has not been modified since last requested
306 Switch Proxy	*No longer used*
307 Temporary Redirect	The requested page has moved temporarily to a new URL
308 Resume Incomplete	Used in the resumable requests proposal to resume aborted PUT or POST requests

4XX – CLIENT ERROR



Message:	**Description:**
400 Bad Request	The request cannot be fulfilled due to bad syntax
401 Unauthorized	The request was a legal request, but the server is refusing to respond to it. For use when authentication is possible but has failed or not yet been provided
402 Payment Required	*Reserved for future use*
403 Forbidden	The request was a legal request, but the server is refusing to respond to it
404 Not Found	The requested page could not be found but may be available again in the future
405 Method Not Allowed	A request was made of a page using a request method not supported by that page
406 Not Acceptable	The server can only generate a response that is not accepted by the client

404 – CLIENT ERROR



```
http://localhost:8080/ServletExceptionHandling/InvalidURL
```

HTTP Status 404 - /ServletExceptionHandling/InvalidURL

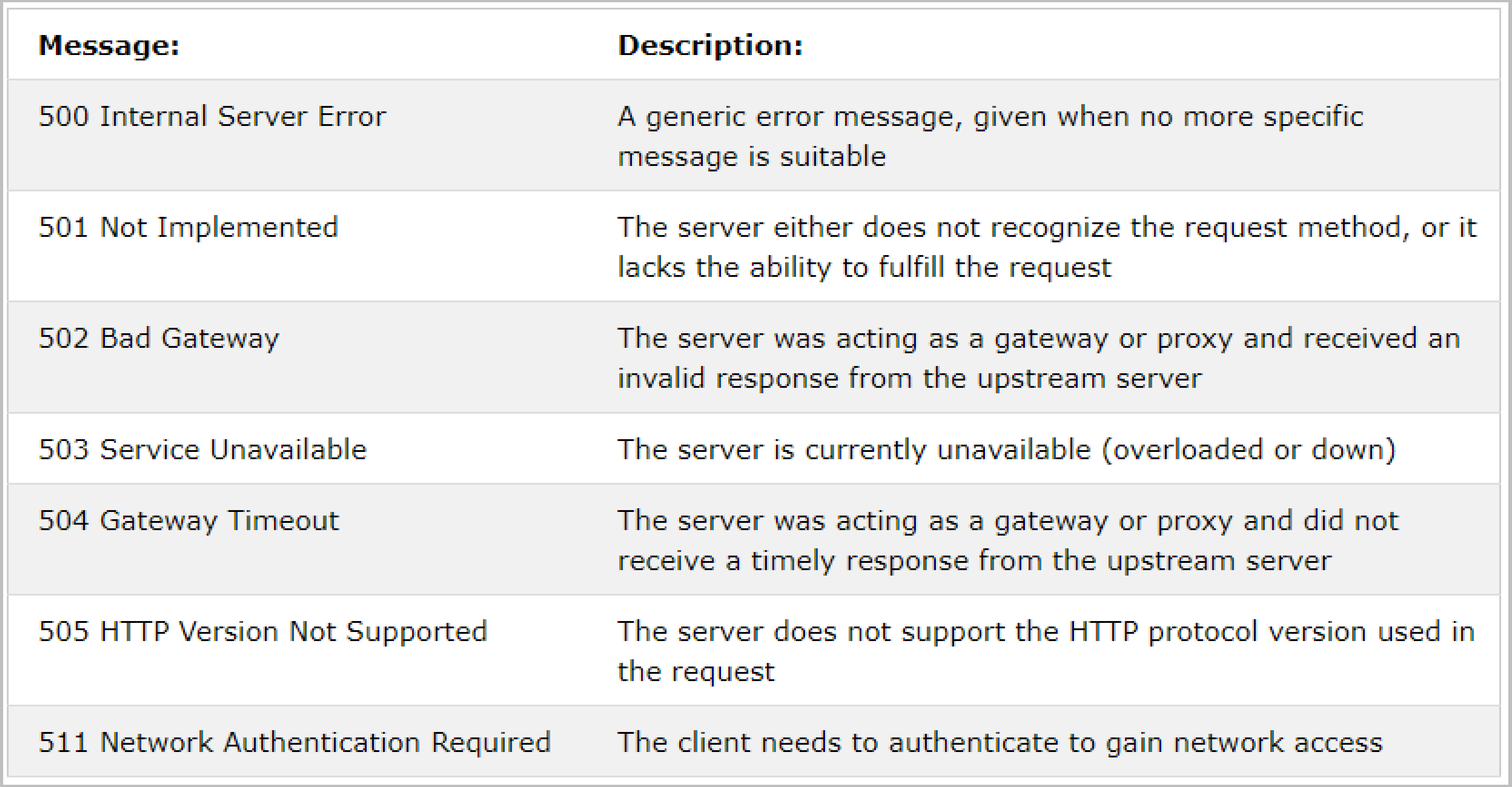
type Status report

message /ServletExceptionHandling/InvalidURL

description The requested resource is not available.

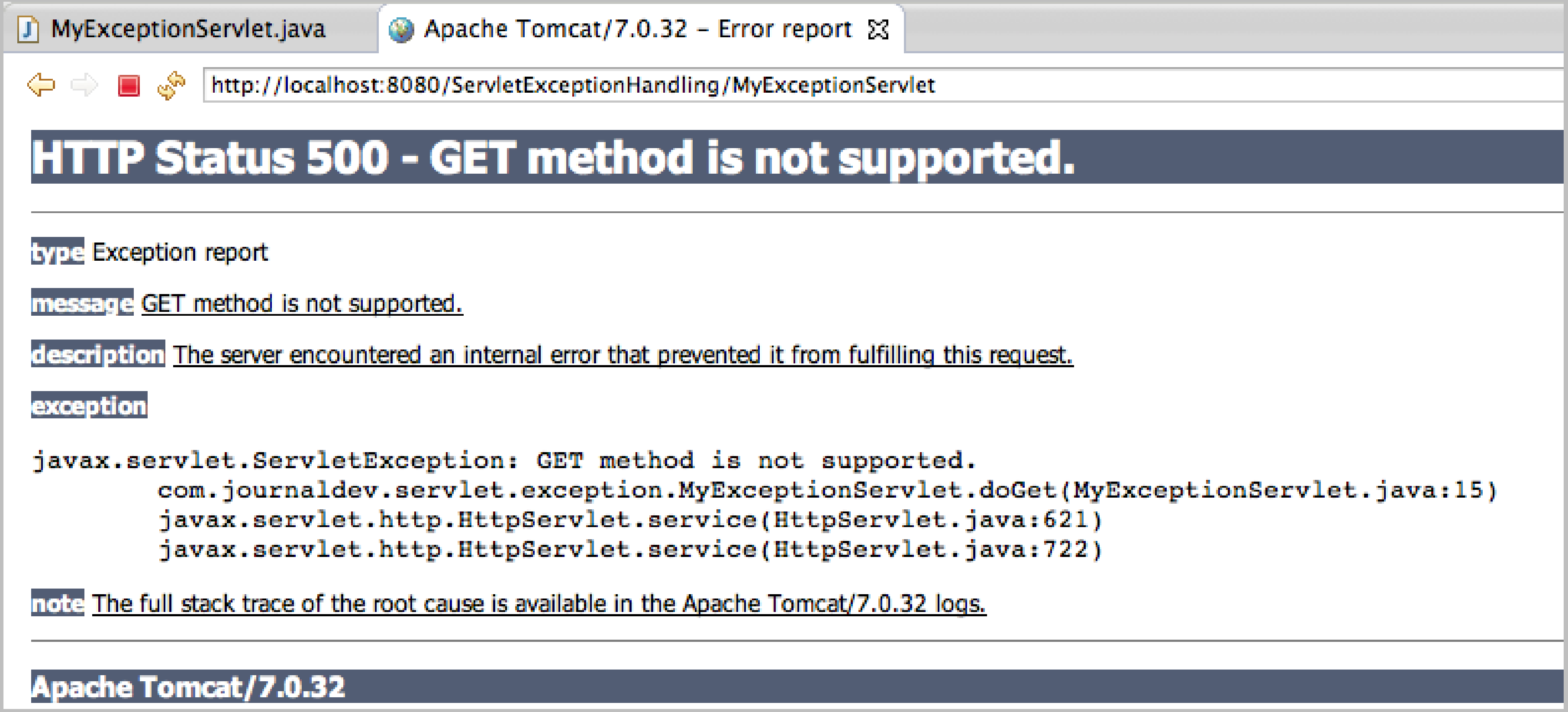
Apache Tomcat/7.0.32

5XX – SERVER ERROR



Message:	Description:
500 Internal Server Error	A generic error message, given when no more specific message is suitable
501 Not Implemented	The server either does not recognize 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 (overloaded or down)
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
511 Network Authentication Required	The client needs to authenticate to gain network access

500 – SERVER ERROR


MyExceptionServlet.java
Apache Tomcat/7.0.32 - Error report
http://localhost:8080/ServletExceptionHandling/MyExceptionServlet

HTTP Status 500 - GET method is not supported.

type Exception report

message GET method is not supported.

description The server encountered an internal error that prevented it from fulfilling this request.

exception

```
javax.servlet.ServletException: GET method is not supported.
	com.journaldev.servlet.exception.MyExceptionServlet.doGet(MyExceptionServlet.java:15)
	javax.servlet.http.HttpServlet.service(HttpServlet.java:621)
	javax.servlet.http.HttpServlet.service(HttpServlet.java:722)
```

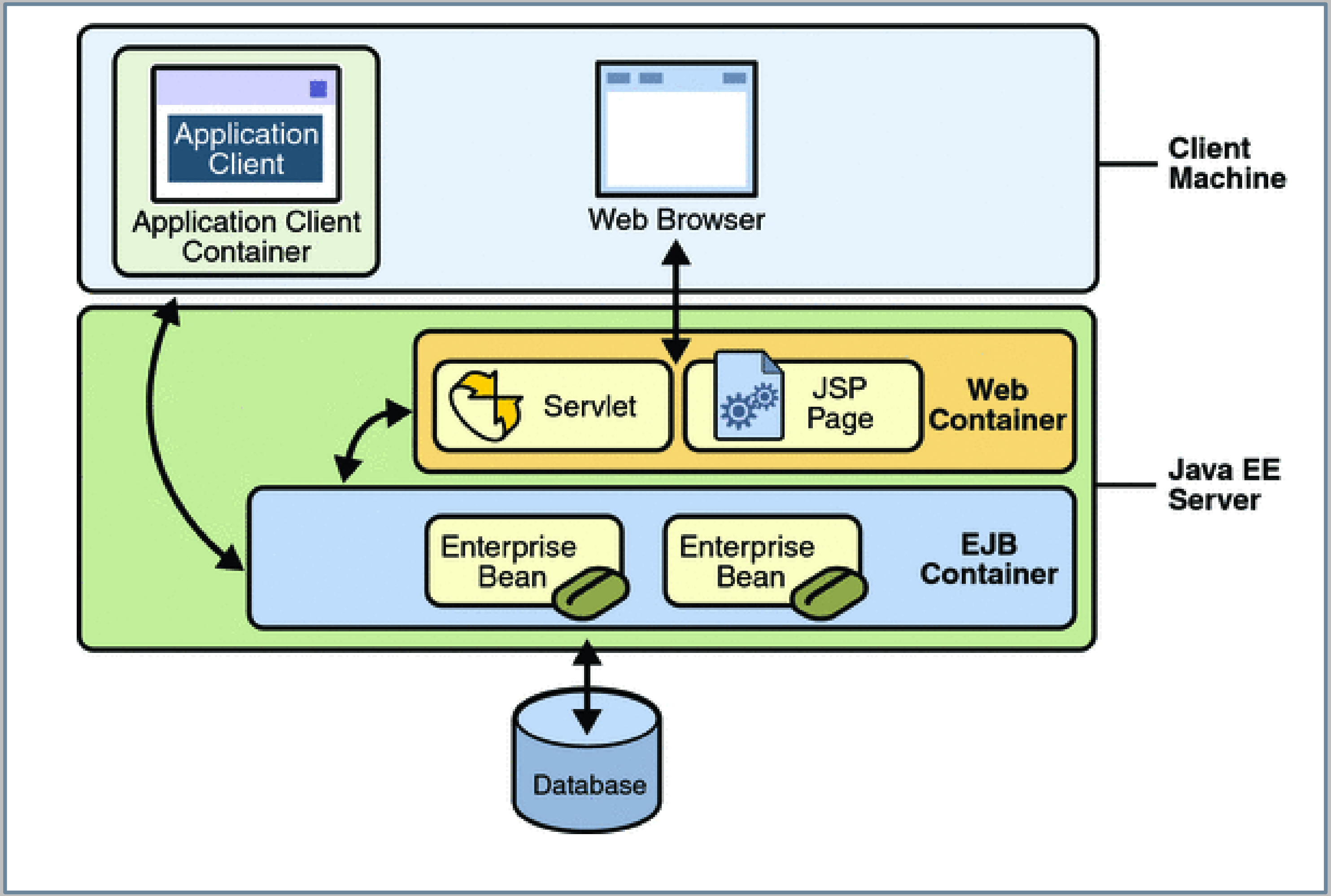
note The full stack trace of the root cause is available in the Apache Tomcat/7.0.32 logs.

Apache Tomcat/7.0.32

WEB APPS (HTTP://WW.XX.YY/WEBAPP)

- A web application is basically a web site that:
  - "Knows who you are"--it doesn't just give you static pages, it interacts with you
  - Can permanently change data (such as in a database)
- A web application can consist of multiple pieces
  - Static web pages (possibly containing forms)
  - Servlets
  - JSP
  - Database connectivity (MySQL)
- A Web Server organizes all these parts into a single directory structure for each web application (E.g Apache_tomcat)

WEB COMPONENTS & CONTAINER

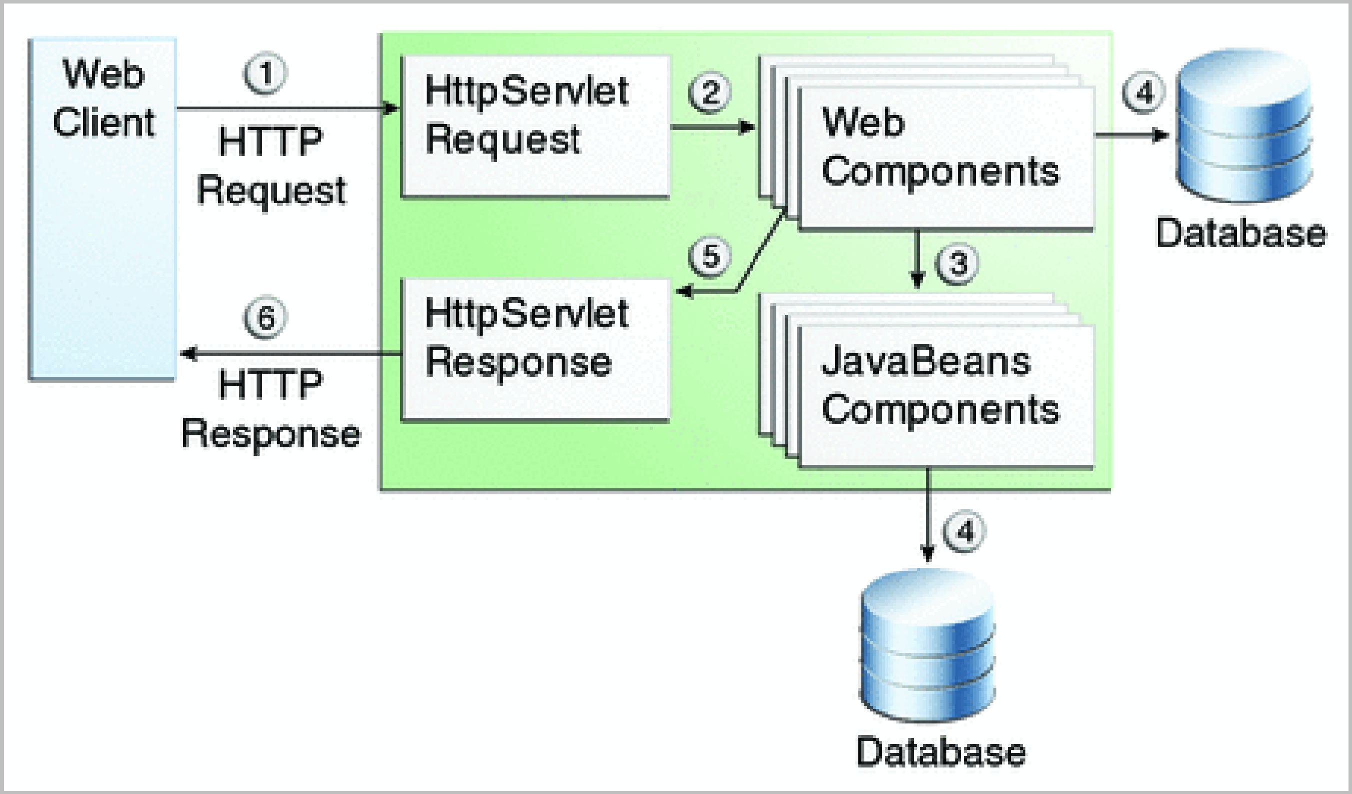


WEB COMPONENTS & CONTAINER

- Web Components are in the form of either Servlet of JSP
- Web components run in a Web Container
  - Tomcat, IBM WebSphere, Sun Glassfish
- Web container provides system services to web components
  - Request dispatching, security, life cycle management

WEB COMPONENTS & CONTAINER

- Web component handles client request, and generate dynamic response with data retrieved from data source.



STRUCTURE OF WEB APPLICATIONS

- Consists of resources that make up a complete application on a Web server
  - Servlets, JSP pages, Java classes
  - Static documents (HTML, images, sounds, etc.)
  - Descriptive meta information (deployment descriptor)
- Represented by a hierarchy rooted at context path
  - http://host.com/context-path/…

Context-path

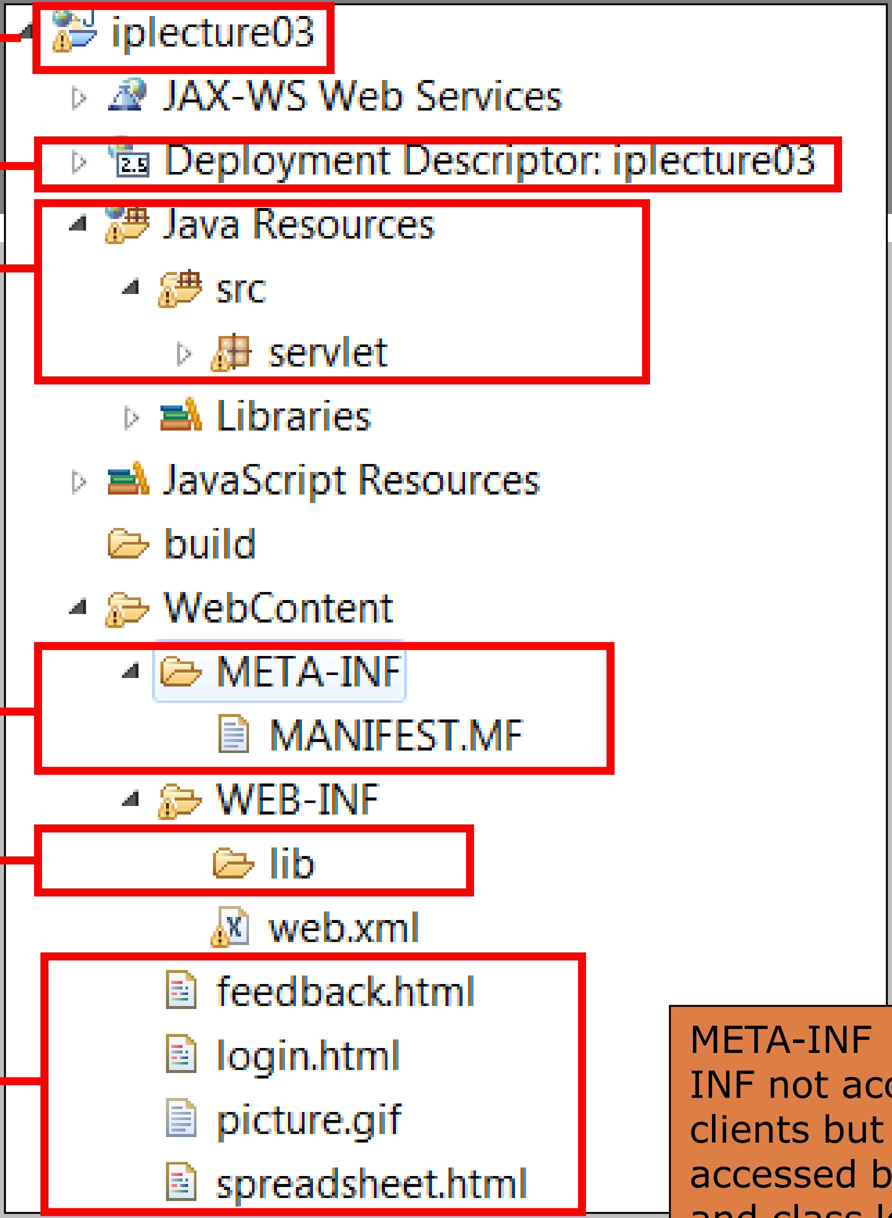
Deployment descriptor

Servlets and Java utility classes can be in subdirectories

Meta information for WAR files

Jar files

JSPs and regular Web content (HTML, CSS, images, etc).



META-INF and WEB-INF not accessible to clients but can be accessed by servlets and class loaders

2020

CI6206 Internet Programming

HTML FORMS

Wong Twee Wee

Ver1.1

WHAT ARE FORMS?

- <form> is just another kind of XHTML/HTML tag
- Forms are used to create (rather primitive) GUIs on Web pages
  - Usually the purpose is to ask the user for information
  - The information is then sent back to the server
- A form is an area that can contain form elements
  - The syntax is: <form ***parameters***> ***...form elements...*** </form>
  - Form elements include: buttons, checkboxes, text fields, radio buttons, drop-down menus, etc
    - Other kinds of tags can be mixed in with the form elements
  - A form usually contains a Submit button to send the information in he form elements to the server
  - The form's ***parameters*** tell JavaScript how to send the information to the server (there are two different ways it could be sent)
  - Forms can be used for other things, such as a GUI for simple programs

THE <FORM> TAG

- <form> tag allows the client to submit the request or/and data to the server
- The submission then can be handled by doGet() method in HttpServlet

CLIENT (HTML/JSP) ← (Get/Post) → SERVER (JAVA)

THE <FORM> TAG

- The <form ***arguments***> ... </form> tag encloses form elements (and probably other elements as well)
- The arguments to form tell what to do with the user input
  - action="***url***" ***or servlet name*** (required)
    - Specifies where to send the data when the Submit button is clicked
  - method="get" (default)
    - Form data is sent as a URL with ?form_data info appended to the end
    - Can be used *only* if data is all ASCII and not more than 100 characters
  - method="post"
    - Form data is sent in the body of the URL request
    - Cannot be bookmarked by most browsers
  - target="***target***"
    - Tells where to open the page sent as a result of the request
    - ***target***= _blank means open in a new window
    - ***target***= _top means use the same window

HTML <FORM> TAG WITH GET REQUEST (CLIENT)

- <form> tag allows the client to submit the request or/and data to the server
- The submission then can be handled by doGet() method in HttpServlet
- Example of <form> tag with GET:

```
<form
action="http://IP_address:8080/servlet/file_name"
method=GET>
        <Input> … </Input>
</form>
```

HTML <FORM> TAG WITH GET REQUEST (SERVER SIDE)

- Example of doGet() method of HttpServlet:

```
public void doGet(HttpServletRequest req,
HttpServletRequest resp) throws ServletException,
IOException
        {
          String message = req.getParameter(“Order”);
          PrintWriter out=resp.setContentType(“text/html”);
          out.println(“<strong>display ”+message+“</strong>”);


        }
```

THE <INPUT> TAG

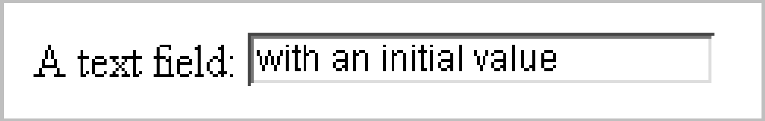
- Form elements use the input tag, with a type="..." argument to tell which kind of element it is
  - type can be text, checkbox, radio, password, hidden, submit, reset, button, file, or image
- Other common input tag arguments include:
  - name: the name of the element
  - id: a unique identifier for the element
  - value: the “value” of the element; used in different ways for different values of type
  - readonly: the value cannot be changed
  - disabled: the user can’t do anything with this element
  - Other arguments are defined for the input tag but have meaning only for certain values of type

TEXT INPUT

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_text

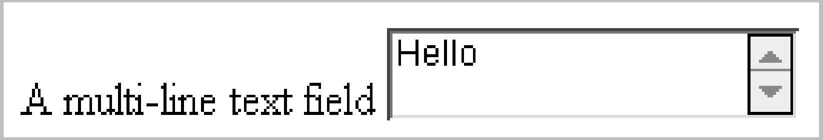
A text field:

```
<input type="text" name="textfield" value="with an initial value" />
```


A text field: with an initial value

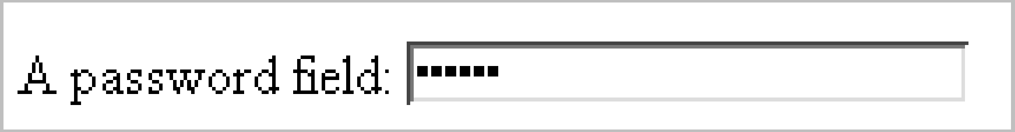
A multi-line text field

```
<textarea name="textarea" cols="24“ rows="2">Hello</textarea>
```


A multi-line text field Hello

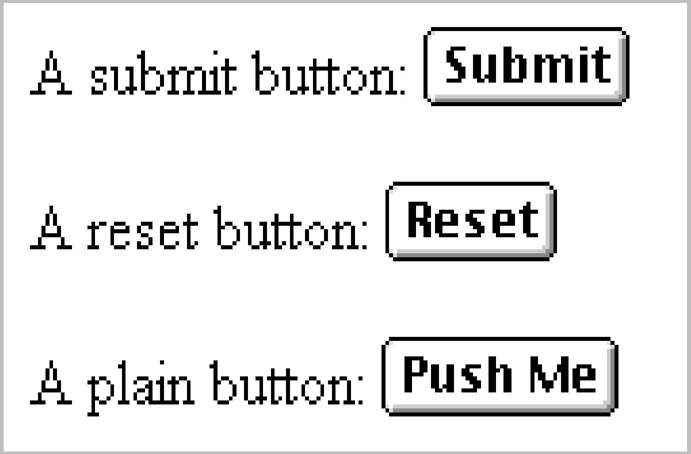
A password field:

```
<input type="password" name="textfield3" value="secret" />
```


A password field: ••••••

BUTTONS

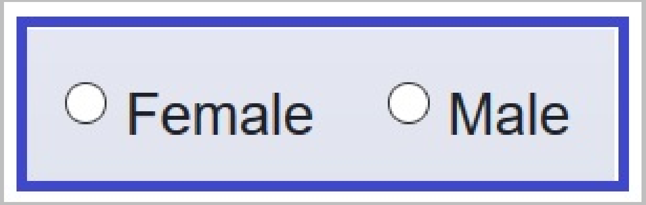
- A submit button:
```
<input type="submit" name="Submit" value="Submit" />
```
- A reset button:
```
<input type="reset" name="Submit2" value="Reset" />
```
- A plain button:
```
<input type="button" name="Submit3" value="Push Me" />
```



- submit: send data
- reset: restore all form elements to their initial state
- button: take some action as specified by JavaScript

RADIO BUTTONS

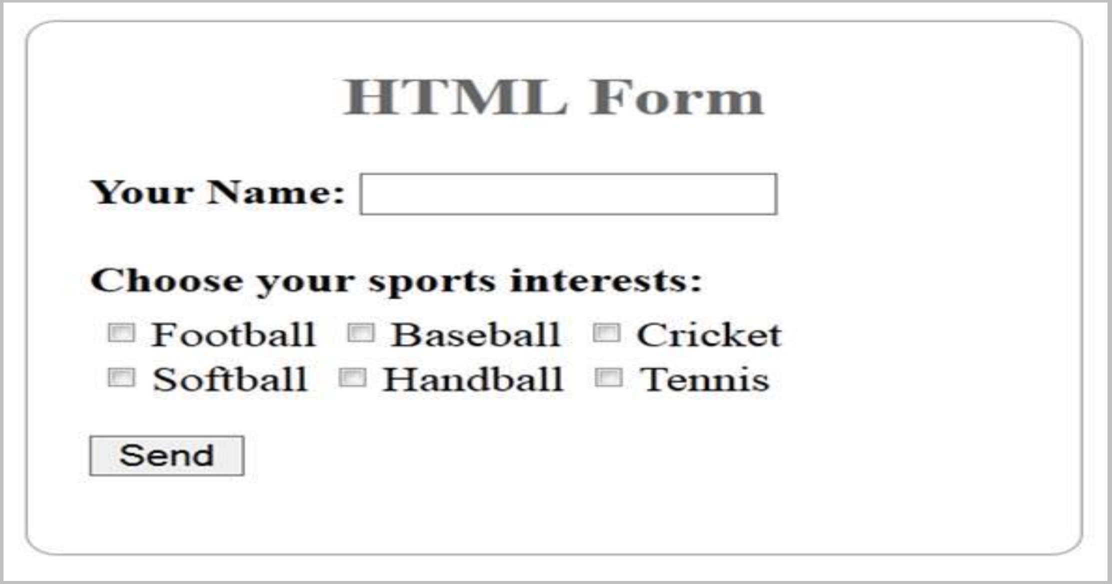
```
Radio buttons:<br>
<input type="radio" name="radiobutton" value="myValue1" male<br>
<input type="radio" name="radiobutton" value="myValue2” checked="checked“ />female
```



- If two or more radio buttons have the same name, the user can only select one of them at a time
  - This is how you make a radio button “group”
- If you ask for the value of that name, you will get the value specified for the selected radio button

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_radio

CHECKBOXES



HTML Form

Your Name:

Choose your sports interests:

☐ Football ☐ Baseball ☐ Cricket
☐ Softball ☐ Handball ☐ Tennis

name: used to reference this form element from JavaScript
value: value to be returned when element is checked

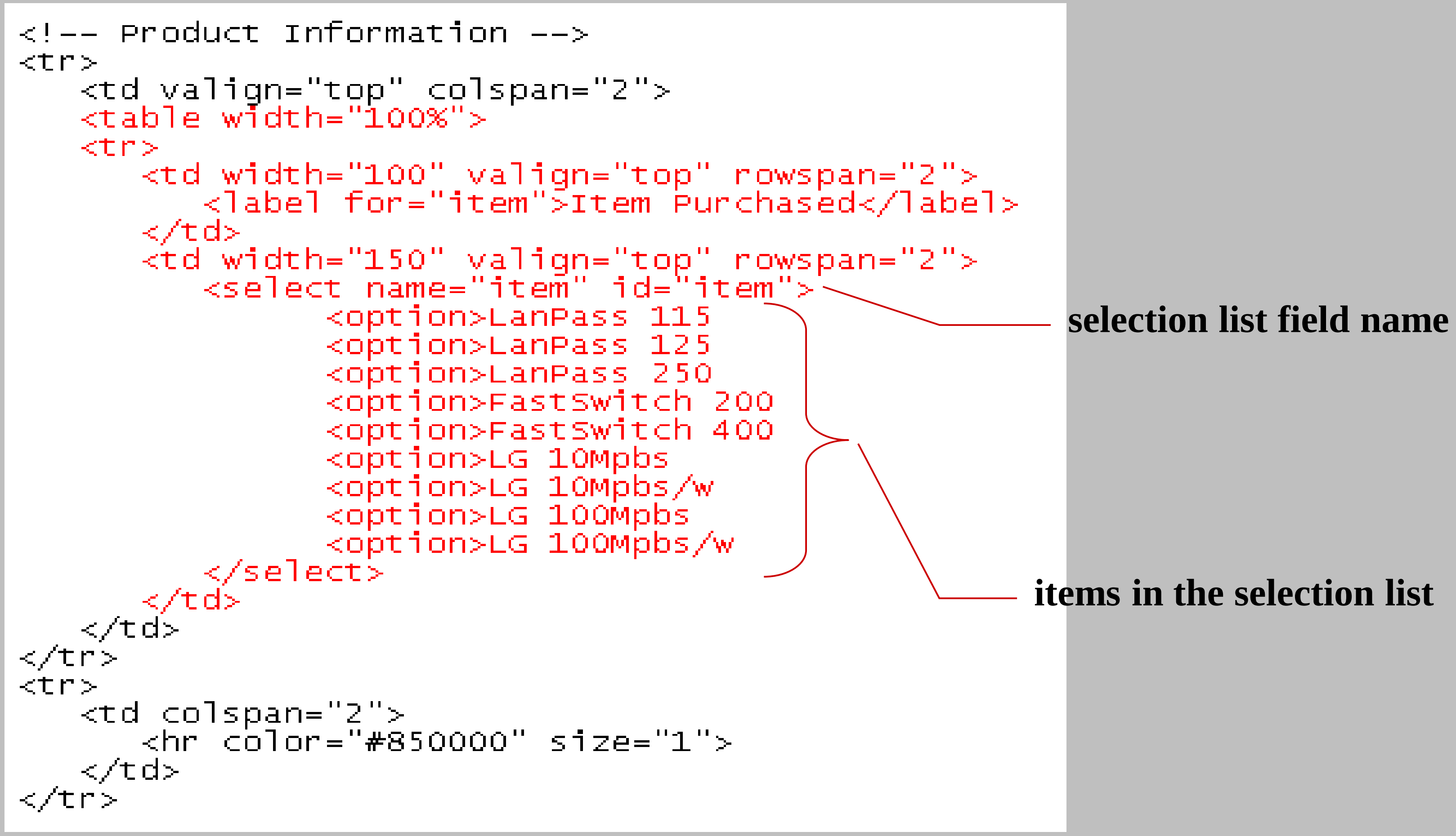
EXAMPLE

```
<input type="checkbox" name="checkbox"
 value="checkbox" checked="checked">
```

CREATING A SELECTION LIST

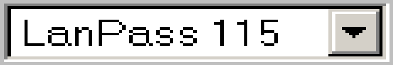
- A **selection list** is a list box from which a user selects a particular value or set of values.
- Selection lists are good to use when there is a fixed set of possible responses.
- Selection lists help prevent spelling mistakes and erroneous entries.
- A selection list is created using the `<select>` tag.
- The `<option>` tag is used to specify individual selection items.

CREATING A SELECTION LIST

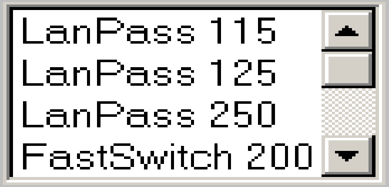


```
<!-- Product Information -->
<tr>
   <td valign="top" colspan="2">
   <table width="100%">
   <tr>
      <td width="100" valign="top" rowspan="2">
         <label for="item">Item Purchased</label>
      </td>
      <td width="150" valign="top" rowspan="2">
         <select name="item" id="item">
               <option>LanPass 115
               <option>LanPass 125
               <option>LanPass 250
               <option>FastSwitch 200
               <option>FastSwitch 400
               <option>LG 10Mpbs
               <option>LG 10Mpbs/w
               <option>LG 100Mpbs
               <option>LG 100Mpbs/w
         </select>
      </td>
   </td>
</tr>
<tr>
   <td colspan="2">
      <hr color="#850000" size="1">
   </td>
</tr>
```

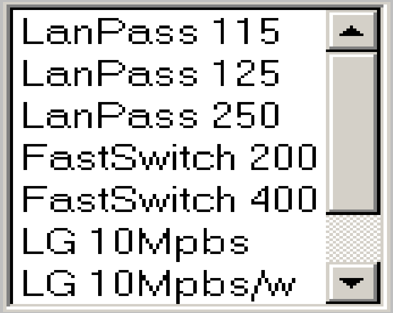
SELECTION LISTS WITH DIFFERENT SIZE VALUES



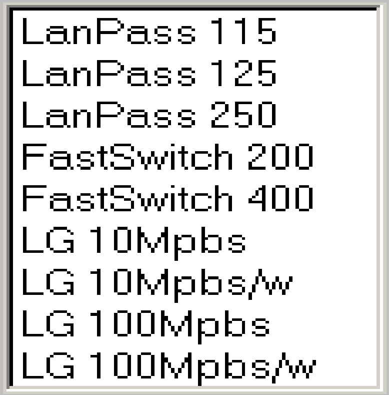
size = "1"



size = "4"



size = "7"



size = "9"

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_elem_select_pre
https://www.w3schools.com/html/tryit.asp?filename=tryhtml_elem_datalist

LAB 1