TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Servlets

EXPERIENTIAL
learning factory

Web Resources

Resources present inside a web application are called as Web Resources. There are 2 types of web resources

1. Static Web Resources

These resources "are present at web application" before making the request, Content/Response of these resources "does not change" for every request (Static Response). **In other words, resources which generates "static response" is called as Static Web Resources**. Few Examples:

1. Any Songs/Video/Movie files Download
2. Any Software's Download
3. Any Books(PDF, MS-Word, etc.,) Download

2. Dynamic Web Resources

These resources "does not present at web application" before making the request & they get generated at the time of request, Content/Response of these resources "May Change" for every request (Dynamic Response). **In other words, resources which generates "dynamic response" is called as Dynamic Web Resources**. Few Examples:

1. Any Net Banking Web Application Transaction Statement Download (PDF file)
2. Google Search Page (HTML Page), Gmail Inbox Page (HTML Page) & Facebook Home Page (HTML Page)
3. Gmail "Download All Attachments" (ZIP File)

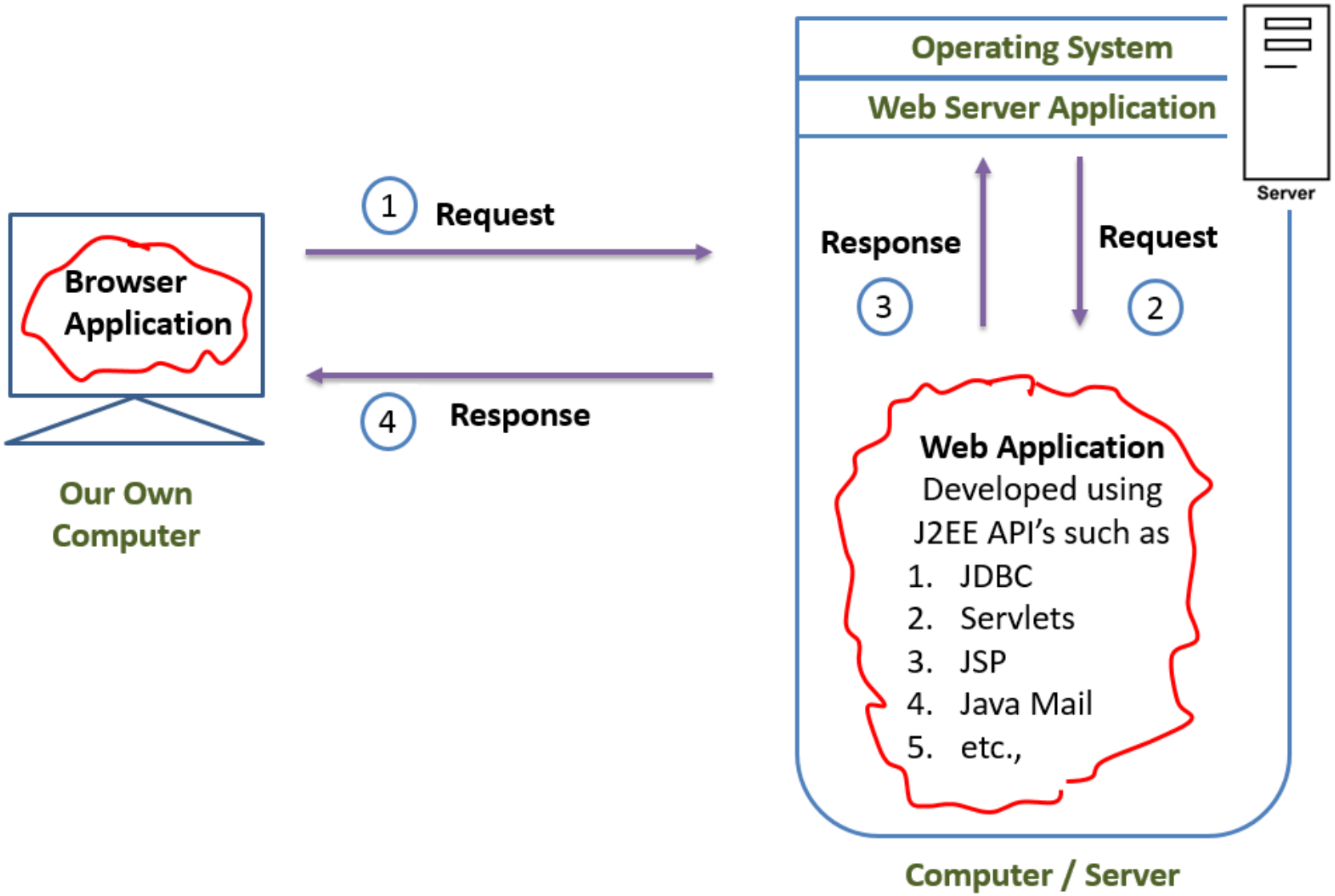
Web Applications

- Web Application is an application which is accessed over the network with the help of browser
- Web application is a collection of web resources
- If a web application consists of "ONLY static web resources" then it is called as "**Static Web Application**“
- If a web application consists of "one / more dynamic web resources" then it is called as "**Dynamic Web Application**”. Ex: Gmail, Facebook, Twiter, FlipKart, etc.,
- Servlets & JSP helps us to develop "Dynamic Web Applications"
- However HTML helps us to develop "Static Web Applications"

Web Server

- Like any other application (Adobe Reader, Media Player, etc.,), Webserver is also an application which runs on Operating System
- Webserver as the name implies “Serves requests to a Web Applications”
- In other words, it helps both web browser & web application to interact with each other
- Hence every web application (Static / Dynamic) is directly under the control of webserver
- Few Examples of Webservers :
    1. Apache Tomcat
    2. Redhat JBOSS
    3. IBM WebSphere
    4. Oracle WebLogic
    5. Oracle GlassFish & Many More …

Web Server Continued ...



Install Apache Tomcat Web Server

- Download the Apache Tomcat (ZIP version) & extract it into some directory
- Set the Following 2 Environment Variables

1. JAVA_HOME
2. CATALINA_HOME

Example,
JAVA_HOME = C:\Program Files\Java\jdk1.6.0_27
CATALINA_HOME = C:\Praveen\apache-tomcat-6.0.26

- Go to "<Tomcat_Location>\bin" folder & double click on "startup.bat" OR "startup" file
- After the above step, Server should start without throwing any exception in the Console
- If Server throws an exception, then it means that server started in "Exception Mode“
- In this case most likely there is a problem in setting up the Environment Variables
- To Shutdown the Web Server

✓ Close the Console Window OR
✓ Use "Ctrl + C" OR
✓ Go to "<Tomcat_Location>\bin" folder & double click on "shutdown.bat" OR "shutdown" file

Different ways to interact with Web Applications

Following are the different ways to interact with Web Applications

1. By Typing an URL in Browser
2. By Clicking on the Hyperlink
3. By Submitting the HTML Form

NOTE:-

- Step 1 happens ONLY ONCE
- However, Step 2 & 3 happens N number of times
- Clicking on "Bookmark" is equivalent to "Typing an URL in Browser"

Web URL Introduction

- Web URL, uniquely identifies a particular web resource inside a web application
- In other words, every web resource (Static/Dynamic) must have its unique address in the form of "Web URL"
- In case of Static Web Resources, URL Consist of Resource File Name

Servlets

- Servlets is an API of J2EE, it accepts request from Browser via Web Server, generates "Dynamic Response" (i.e response is generated at the time of Request) & gives it back to Browser via Web Server
- Hence Servlets & JSP acts like a "Dynamic web Resources“. Hence dynamic web applications, should consist of at least ONE Servlet/JSP
- The Dynamic Response, may be a "HTML response" or "Non-HTML Response“. For Example:

1. Any Net Banking Web Application, Transaction Statement Download(PDF file) is Non-HTML Response
2. Google Search Page (HTML Page) is a HTML Response
3. Gmail "Download All Attachments" (ZIP File) is a Non-HTML Response

- HTML helps to generate "Static Response“
- **In the world of Java, Servlets are the "One and Only API" that accepts web request and generate "Dynamic Response“**
- Since Servlets are like Dynamic Resources & hence Servlets must have its unique address in the form of "Web URL“
- i.e. Even though Servlets are Java Programs, we should not run them like normal Java Programs instead, we should access via Web Browser using corresponding Web URL

Servlet Exercise 1

- Creating a Servlet which Prints the Current Date & Time in Browser as a HTML Response

- Creating a Servlet which fetches particular employee data (hard coded value) & prints it in Browser

Auto Refresh a Web Page:-

1. Using HTML Header

```
<head>
  <meta http-equiv="refresh" content="1">
</head>
```

2. Using Java Code

```
resp.setHeader("Refresh", "1");
```

Where "resp" is a object reference of HttpServletResponse object. In both the cases "1" represents "Time in Sec."

Build & Deployment Process

Build Process

Process of Building JAR / WAR / EAR file is known as Build Process

Deployment Process

Process of deploying JAR / WAR / EAR files specific location is known as Deployment Process

WAR File

1. Collections of Classes & its corresponding packages
2. Other Necessary Resources
3. Dependent JAR files (one or more)

Web URL

- Web Application is a Collection of web resources (static OR dynamic)
- Web URL uniquely identifies ONE of the web resource (static OR dynamic) present inside a Web Application
- In other words, every web resource should have its unique address in the form of Web URL
- Max. number of characters allowed in Web URL is around 2000 characters (exact number depends on Browser. For ex, IE supports 2048 characters)

Web URL Structure:

Protocol://Domain:Port/Path?QueryString#FragmentID

Protocol

- When one application wants to communicate with other (in our case browser & server), there needs to be a common language which both application understands & that language should have set of rules and instructions. This common language is known as “Protocol” where protocol is a "Set of Rules“

- Web Browser & Web Server applications communicate using

1. Hyper Text Transfer Protocol (HTTP)
2. Hyper Text Transfer Protocol Secure (HTTPS)

- HTTPS encrypts any data which is shared between browser & server
- As the name implies most of the time HTTP Response contain HTML
- In URL

✓ it’s an optional information,
✓ "case in-sensitive" and
✓ default protocol is HTTP

Domain Name

- It uniquely identifies a computer in a network in which web application is present
- Domain consists of Computer Name (preferred) or IP address of the computer in which web application is present
- In URL

✓ **it's a Mandatory Information and**

✓ **it's case in-sensitive**

Port Number

- Port number in Web URL uniquely identifies web server application
- Default port number for **HTTP is 80 & HTTPS is 443**
- **In URL this is an optional information, When it's not used default port number is used depending on the protocol present in Web URL**
- **In Tomcat Webserver, default port number for HTTP is changed from 80 to 8080 and default port number for HTTPS is changed from 443 to 8443**

Web Resource Path

- We know that web application is a collection of web resources (Static / Dynamic) & also Web server can consist of one/more web applications
- Path is the full path of the web resource at web application side
- It consists of, Web Application Name / (File Name in case of Static Resource OR Configured URL in case of Dynamic Resource)
- "Web Application Name" uniquely identifies One web application inside webserver
- "File Name" uniquely identifies Static web resource inside that web application
- "Configured URL" uniquely identifies Dynamic web resource inside that web application
- In URL,

✓ it's an optional Information and
✓ it's case sensitive

Query String

- Query String is a name & value string pair which passes information ONLY to Dynamic Resources such as Servlets & JSPs

- In URL, It's an optional information and if present, it starts with question mark followed by one or more name=value pair which are separated by an ampersand(&)

- In Query String
✓ Name is Case-Sensitive but
✓ Value is Case-Insensitive

Examples:

www.google.com/search?q=Praveen
https://www.google.co.in/search?q=ABC&sitesearch=www.youtube.com

Fragment ID

- A Fragment ID or Fragment Identifier, as the name implies, it refers to a particular fragment / a section within a web page

- In URL,

✓ It's an optional information
✓ Case In-Sensitive &
✓ if present, it starts with a hash (#) character followed by an unique identifier

Example:

http://tomcat.apache.org/tomcat-6.0-doc/manager-howto.html#Introduction

How come “Web Path” is Optional?

1. Goto <Apache Tomcat Location>\webapps\ROOT

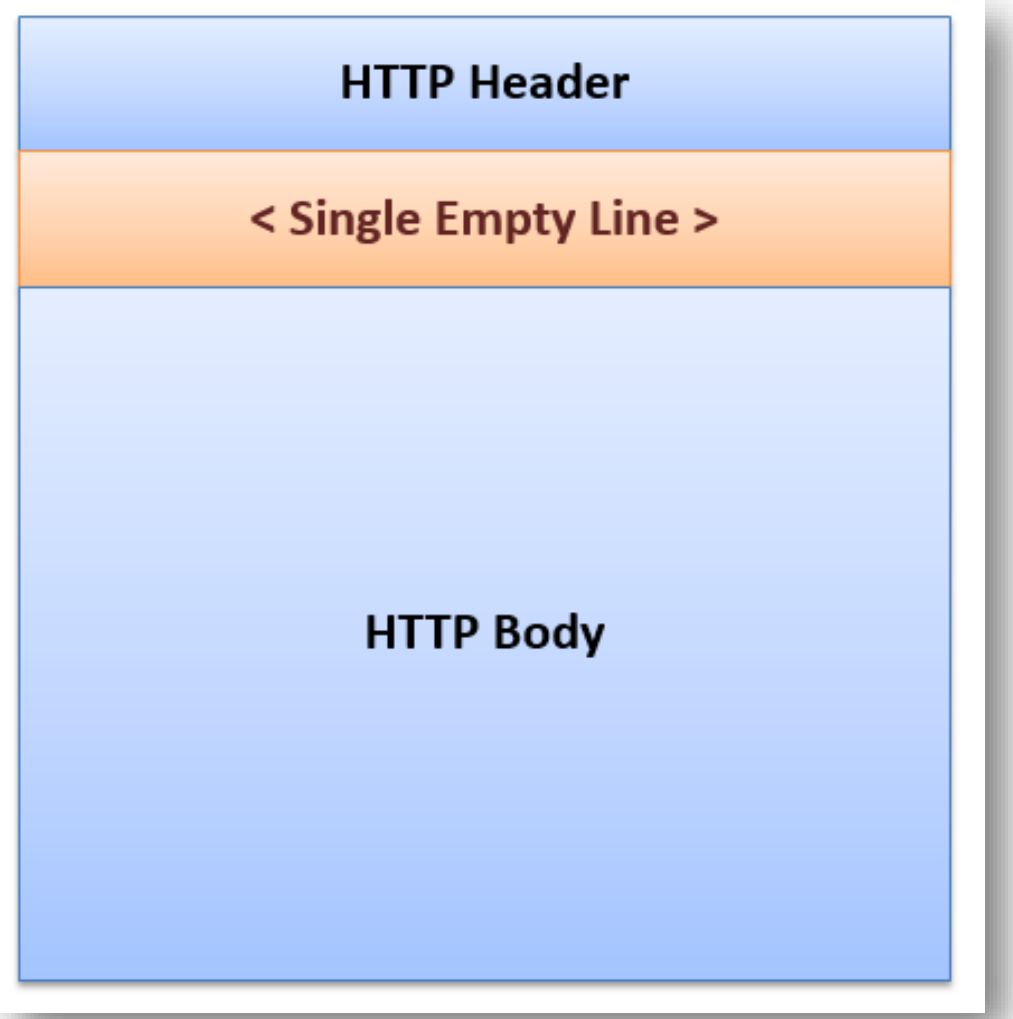
2. Open "index.jsp"

3. Remove all the Lines in the above file & write the below lines

4. <%
String url = "http://<Your-Computer-IP>:8080/studentsApp/Login.html";
response.sendRedirect(url);
%>

HTTP Structure

- Like HTML, HTTP protocol also has a structure and it consist of Header Part & Body Part
- Both Header Part & Body Part are separated by a "Single Empty Line"



TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

Installation of Fiddler Application/Tool

Raw HTTP Request & Response

Raw HTTP Request

```
POST http://localhost:8080/studentsApp/login HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 23
Cache-Control: max-age=0
Origin: http://localhost:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;
Referer: http://localhost:8080/studentsApp/Login.html
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.8

regno=123&pass=sadsadsa
```

Raw HTTP Response

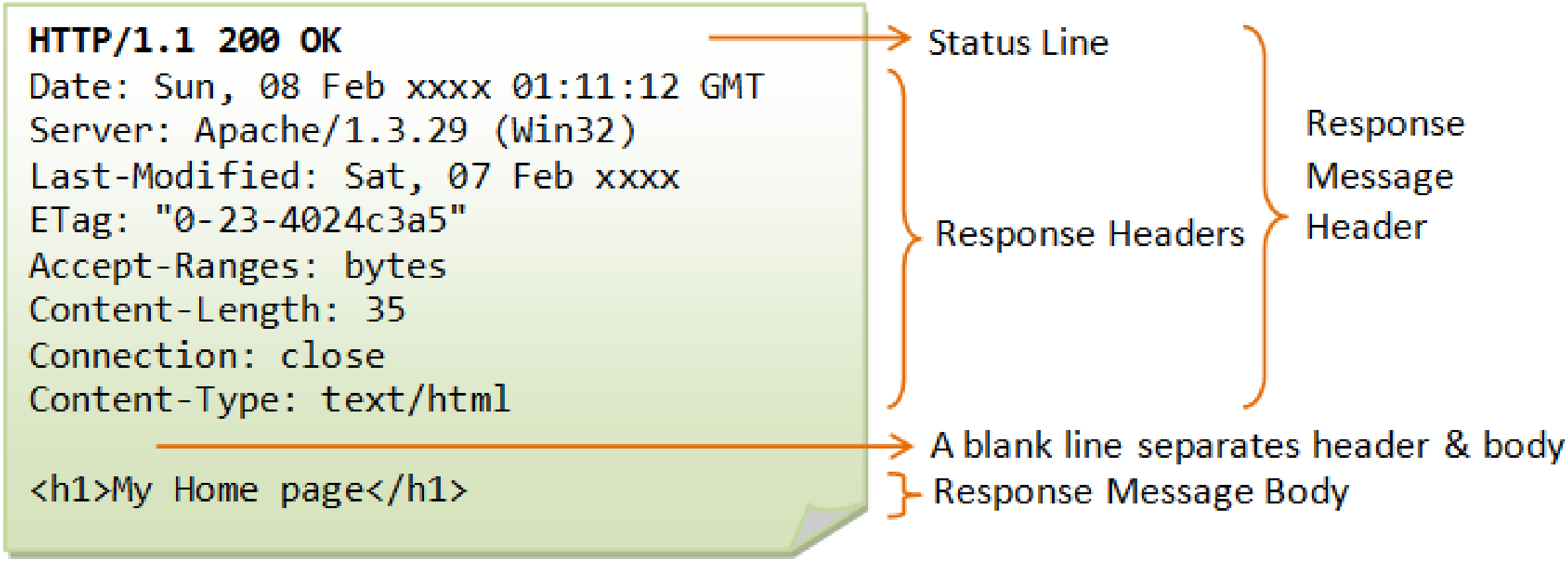
```
HTTP/1.1 200
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 61
Date: Tue, 19 Sep 2017 04:41:44 GMT

Inside doPost() Method ...
Reg. No. : 123 Password : sadsadsa
```

HTTP Request

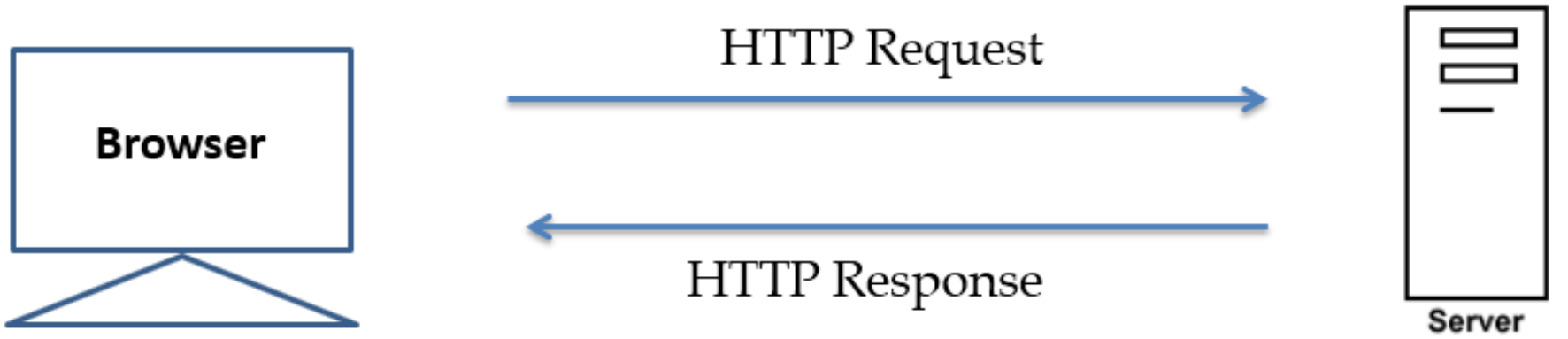

```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

- Request Line
- Request Headers
- Request Message Header
- A blank line separates header & body
- Request Message Body

HTTP Response



Key Elements of Request & Response



Key elements of HTTP Request are

1. **Web URL**
2. **Form Data (if any)**
3. **HTTP Method**
4. **Cookies (if any)**

Key elements of HTTP Response are

1. **Status code**
2. **Actual Content (if any)**
3. **Content Type (if any)**
4. **Cookies (if any)**

Status Code

- Status code represents the status of HTTP Request. For example,

a) 200 = Server Successfully handled the request
b) 404 = Requested Resource(static/dynamic) is Not Found at server side
c) 500 = Server encountered an unexpected condition which prevented it from fulfilling the request

- It’s a Mandatory information & it will be present in Header of HTTP Response

- Generally Webserver provides "Status Code" info in HTTP Response

Actual Content

- It’s an Optional Information & if present then its present in the Body of HTTP Response
- In case of static resource, content of the resource becomes the “Actual Content”
- In case of dynamic resource, content present in servlets / JSP becomes the “Actual Content”
- In case of Error Scenarios webserver generates error information & it becomes the “Actual Content”

Content Type OR MIME Type

- Content Type OR Multipurpose Internet Mail Extensions (MIME) Type, tells the browser that what "type of Actual Content" it's going to receive so that it can prepare itself to handle the response data

- For example,

1) Open an Adobe Reader to handle PDF content
2) Open Media Player to handle media content etc.,

- It's an Optional Information, if present, then its present in Header part of HTTP Response and case in-sensitive

- The default content type is "text/html"

- Few Examples :

➢ text/html
➢ application/pdf
➢ image/jpeg
➢ application/zip and Many More ....

Web URL

- Web application is a "Collection of Web Resources" & every web resource (static / dynamic) should have its unique address in the form of web URL

- Hence every request should consist of Web URL (Mandatory information) & it will be present in Header of HTTP Request

Form Data

- Data collected using HTML form is called as "Form Data"

- Size of the Form Data varies from "Form to Form". For Example,

1) "Gmail/Facebook Login", it's few KB
2) "Naukri Profile Creation", it's few MB
3) "YouTube Video Upload", it's few MB to GB

- Whenever we make request by Submitting Form, then ONLY HTTP Request will have Form Data

- Hence in HTTP Request it’s an Optional Information

- If Present, it may be present in either Header or Body of the HTTP Request which depends on HTTP Method present in the Request

Cookies

- Cookies are present in Both Request & Response

- It's

1) an Optional Information,
2) if present, then its present in Header part of HTTP Request/Response

Servlet Exercise 2

Creating a Servlet which fetches particular employee data from Data Base, which it gets from HTML Form or Search page & prints it in Browser

HTTP Methods

- It's a mandatory information present in the header of the HTTP Request

- HTTP Method is the first element in the HTTP Request

- HTTP 1.0 had 3 Methods & in HTTP 1.1, "Five New Methods" got introduced. So in total HTTP 1.1 has 8 different methods & every Http Request should consist of "ONE of the 8 HTTP Methods"

1. HEAD (part of HTTP 1.0)
2. TRACE
3. PUT
4. DELETE
5. OPTIONS
6. POST (part of HTTP 1.0)
7. GET (part of HTTP 1.0)
8. CONNECT

- **NOTE**: HTTP Methods SHOULD be written in "Uppercase". Code Word : **HTPP DOG C**

HTTP Methods Continued ...

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

- Servlet API has "Default Implementation" for these methods "excluding CONNECT method"
- All these default implementations are present in the Servlet API class by name "javax.servlet.http.HttpServlet"

HTTP Method	Related Servlet Method
HEAD	protected void doHead(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
TRACE	protected void doTrace(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
PUT	protected void doPut(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
POST	protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
DELETE	protected void doDelete(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
OPTIONS	protected void doOptions(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
GET	protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
CONNECT	NO Implementation

HTTP Methods Continued ...

- Since method signature is same apart from Method name, these methods in general called as doXXX(HSR, HSR) methods

- So whenever we create a Servlet by extending HttpServlet

✓ We can inherit all these default implementations

OR

✓ We can override all of them

OR

✓ We can override couple of them (Generally we override doGet() / doPost() Method)

- Depending on the HTTP method present in the request, Webserver invokes the corresponding doXXX(HSR, HSR) method

- In other words, HTTP method indicates the desired action to be performed on the Dynamic Web Resource i.e. Servlet

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

1. Typing a URL in Browser makes request to contain GET method

2. Clicking on a Hyper link in Browser makes request to contain GET method

3. Submitting the form with method="get" form attribute in Browser makes request to contain "GET" method

4. Submitting the form with method="post" form attribute in Browser makes request to contain "POST" method

5. Submitting the form "without method form attribute declaration" in Browser makes request to contain "GET" method

<form method="get/post"> ??

1. If a form collects
   - "Sensitive Data" like Password (ex: Gmail Login Page)
     OR
   - Very Large Data but in-sensitive (ex: YouTube Video Upload)
     OR
   - Both of them (ex: Naukri Profile Creation Page) then that form "SHOULD" use method="post"

2. If a form collects "In-Sensitive and Less Amount of Data" like search that form may use
   - method="post" or
   - method="get" (preferred)

3. Hence whenever Servlet gets a request via Submitting the form & that form has method="post" then we have to override doPost() method

4. For rest of the cases, we have to override doGet() method

Differences b/w GET & POST

GET / doGet()	POST / doPost()
GET is a default method	POST is not default. We have to explicitly declare method=”post” in the HTML form
GET requests "have Empty Body"	POST requests "have Non-Empty Body"
Hence incase of GET, Form Data will be present in Header in the form of Query String	Incase of POST, Form Data will be present in Body
Insecure; because form data get exposed to the outside world	Secure; because form data will be present in Body, hence it will not be exposed to the outside world
The amount of data sent using GET is restricted because URL can contain around 2000 characters	There is no restriction on the amount of data sent using the POST
We cannot send the files using GET	we can send entire files using POST. Ex: Resume Upload, Video Files upload, etc.,
We "can bookmark" the GET requests	We "cannot bookmark" POST requests

Servlet Container

- Servlet Container is a sub-component of web server that helps both web server & servlet to communicate with each other
- As the name implies, all Servlets of dynamic web application are directly under the control of Servlet Container

How Servlet Container Works

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

1. Whenever request comes, web server hand over the complete request to servlet container
2. Container by looking at the URL present in the request & referring web.xml it comes to know the servlet which handles that request
3. Container then "creates an instance" of that Servlet
4. Once Instance creation is successful then, it converts the "Raw HTTP Request" to a Java Object of type “HttpServletRequest” & also creates "HttpServletResponse" object
5. Depending on the HTTP Method present in the request, container invokes corresponding doXXX() method by passing these request & response objects
6. Once doXXX() method execution is over, container converts the response object to "Raw HTTP Response" & gives it back to web server
7. Once the response has been give back, Servlet Container garbage collects the request & response objects
8. In other words for every request, container creates new request & response objects
9. i.e. the Life Span of these Objects is Created: once request comes to Servlet & Destroyed: once response is given back

Advantages of Servlet Container

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

1. Communication Support :

Container helps both web server & servlets to communicate with each other

2. Multi-Threading Support :

Container automatically creates a new thread for every incoming request

3. Life Cycle Management :

Container manages/controls the "Life Cycle of a Servlet"

4. JSP Support :

Container takes care of converting JSP into a Servlet

HttpServletRequest

- "HttpServletRequest" object, in short called as "Request Object",is an Object representation of "Raw HTTP Request“
- We should make use of this object to get Info from "Raw Http Request"
- HttpServletRequest is an Interface & it extends an another Interface by name "javax.servlet.ServletRequest"
- Request Object has many "Getter Methods" which helps us to get the information from "Raw Http Request“

Some Methods

1. getMethod(): This method return the HTTP Method present in the request as a String Value
2. getRequestURL(): This method return the URL present in the request as a StringBuffer
3. getProtocol(): This method return the Protocol present in the request as a String Value
4. getParameter(String name):
5. getParameterValues(String name):

✓ Both the above Methods helps us to get the "Form Data / Query String information" from "Request Object"

✓ Both these methods return NULL if the parameter name does not exist

HttpServletResponse

- "HttpServletResponse" Object, in short called as "Response Object", is an Object representation of "Raw HTTP Response"
- We should make use of this object to send Info as part of "Raw HTTP Response"
- HttpServletResponse is an Interface & it extends an another Interface by name "javax.servlet.ServletResponse"
- Response Object has Methods which helps us to send the information as part of "Raw HTTP Response“

Some Methods

1. setContentType(String contentType): This Method helps us to set Content Type info to Response Object
2. getWriter(), println(String response) and print(String response)

✓ These methods helps us to provide "Actual Content“ info in Reponse Object

✓ First we should get "java.io.PrintWriter" Object from Response Object by invoking a method by name "getWriter()"

✓ PrintWriter has "print()/ println()" methods which helps us to add Actual Content to Response Object

HttpServletResponse Methods...

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

1. setContentType(String contentType):
   This Method helps us to set Content Type info to Response Object

2. getWriter(), println(String response) and print(String response)
- ✓ These methods helps us to provide "Actual Content“ info in Reponse Object
- ✓ First we should get "java.io.PrintWriter" Object from Response Object by invoking a method by name "getWriter()"
- ✓ PrintWriter has "print()/ println()" methods which helps us to add Actual Content to Response Object

3. setHeader(String key, String value):
   As the name implies, this method helps us to set the Header info in the Response

4. sendError(int statusCode, String errMsg):
   As the name implies, this method helps us to send the Error Response to Browser

Configuring a URL for a Servlet

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

- Every Servlet must have a URL & web.xml helps to configure a URL for a Servlet
- Container uses this information to identify a specific servlet to handle a given request
- There must be a at least ONE URL configured for a Servlet. Also Servlet can have "More than One" URL
- Below are the different ways to configure the URL for a Servlet

1. Exact Matching
2. Directory Matching
3. Extension/Pattern Matching

GenericServlet

- It's an abstract class, part of Servlet API, a sub-class of GenericServlet is called as Servlet, it can handle "ANY Protocols" including HTTP & HTTPS

- In other words, it becomes "Protocol-Independent Servlet“

- GenericServlet has "one abstract method", by name service(), hence it's an "abstract class"

```
public abstract void service(ServletRequest req, ServletResponse res)
throws ServletException, IOException;
```

- Hence whenever we create a Servelt by extending GenericServlet we MUST provide an implementation for service(SR, SR) method

GenericServlet vs HttpServlet

GenericServlet	HttpServlet
Protocol Independent	Protocol Dependent. Supports only HTTP & HTTPS protocols
Abstract Class; because service() method is declared as abstract	Abstract class; but none of the methods are declared as abstract
If we extend the GenericServlet then we must provide an implementation for service() method	There is NO restriction on overriding any version of the service method but generally we override one or more doXXX() methods
GenericServlet does not extend any other Servlet API related class	HttpServlet extends GenericServlet which is part of Servlet API

Servlets Few Important Points ...

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

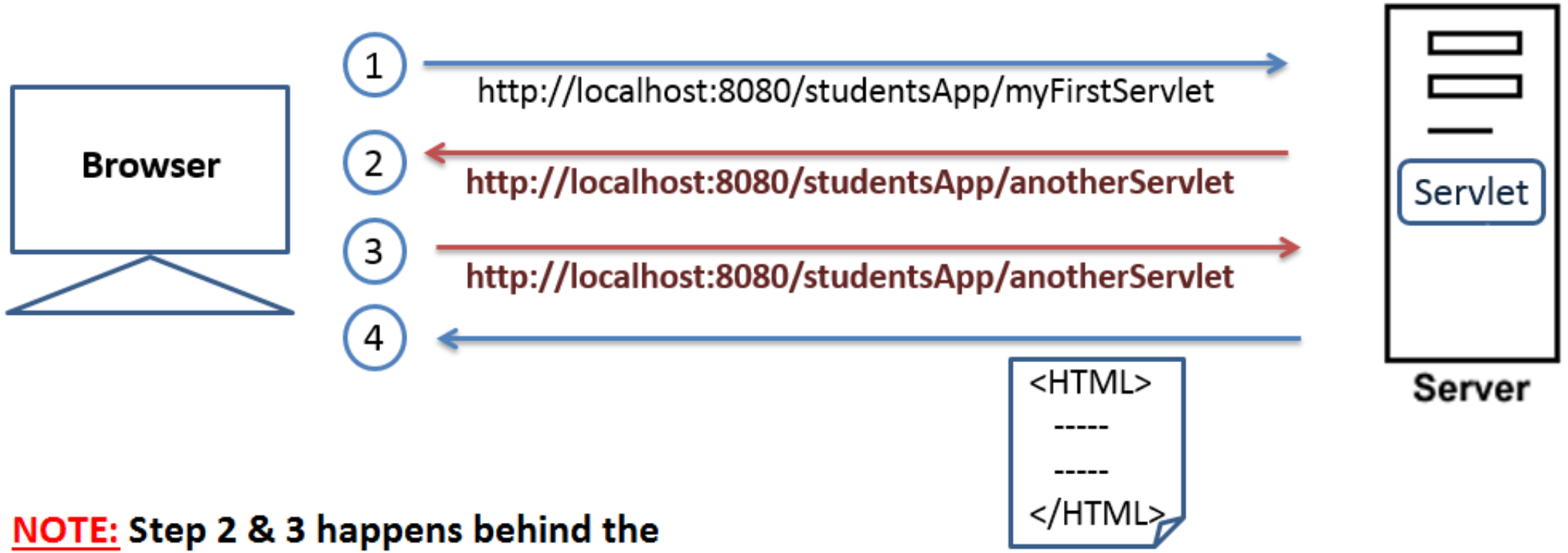
- If a class, extends either HttpServlet or GenericServlet & a "subclass of that class is also be called as Servlet"
- Servlets must be a "concrete class" otherwise they fail at runtime
- Servlets can have, Local Variables (Static / Non-Static), Block of Code (Static / Non-Static), Inner Classes, It's own Methods (Static / Non-Static but non-abstract)
- If we declare abstract method(s) in servlet then we are forced to declare that servlet as "Abstract Class" & we MUST sub-class it. Otherwise it fails at runtime during Instantiation
- We can have "main() method" in the servlet but its of "no use". Servlets are directly under the control of Container & Container will not execute main method
- Servlets MUST have public default Constructor OR combination of any other constructor along with public default constructor
- Servlet API is protocol independent in nature, but are most often used with HTTP & HTTPS protocols
- There is only one instance exist for any servlet i.e. Servlets are "Singleton in nature“
- At any point of time there will be multiple threads acting on servlet instance. Hence by default servlets are Multi Threaded in Nature. In other words Dynamic Web Applications are "Multi Threaded environment"

ServletContext vs ServletConfig

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

- ServletContext is an Interface and "an Object of ServletContext" is used by container to pass information to ALL the servlets which are part of an application
- ServletConfig is an Interface and "an Object of ServletConfig" used by a container to pass information to a particular Servlet
- In web.xml, context parameters are declared under <context-param> tag (one / more)
- In web.xml, servlet config parameters are declared under <init-param> tag (one / more) which is a subtag of <servlet> tag

Redirect



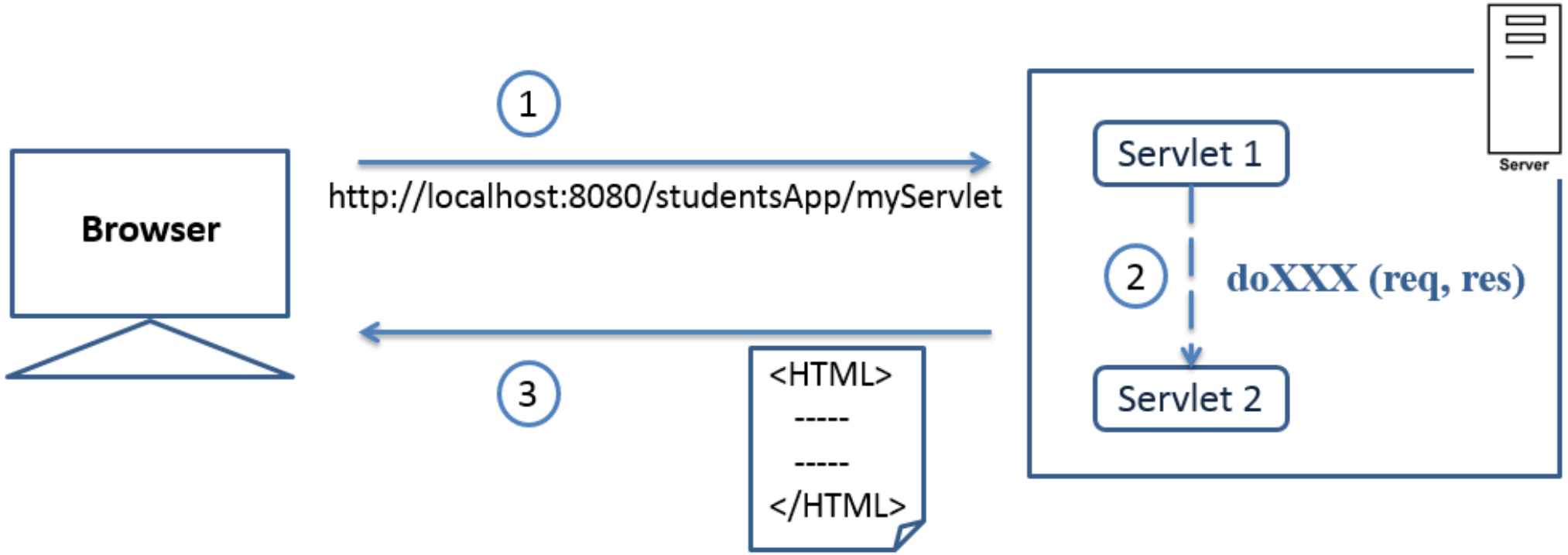
NOTE: Step 2 & 3 happens behind the scene without User Intervention

Redirect Continued ...

- Using Browser, user makes the request to a servlet
- Servlet redirects that request to another resource (Internal / External). In this case Browser gets the URL as a response
- "Browser makes a New Request to this URL“
- Incase of redirected request, HTTP Method present in that request will always be a "GET"
- Browser displays the whatever the response given by new URL
- Redirect happens at Browser side & incase of Redirect "URL in the browser changes“
- To redirect the request call sendRedirect() on HttpServletResponse object

  public void HttpServletResponse.sendRedirect(String url)
  throws IOException

Forward


Browser
1
http://localhost:8080/studentsApp/myServlet
3
<HTML>

</HTML>
Server
Servlet 1
2
doXXX (req, res)
Servlet 2

Forward Continued ...

- Using Browser, user makes a request to a servlet
- Servlet internally forwards that request to another Internal Resource (Static/Dynamic) by passing Request & Response objects
- This internal resource handles that request & gives back the response
- Browser displays the response. In this case Browser will not have any clue on what went behind the scene (i.e. forward happens at Server side)
- Also, URL in the browser doesn't change
- To forward the request call forward() on RequestDispatcher object

Redirect vs Forward

Redirect	Forward
Redirect happens @ "Browser side"	Forward happens @ "Server side"
URL in the browser changes	URL does not change
We can Redirect the request to "both Internal & External Web Resources"	We can forward the request "ONLY to Internal Web Resources"
Redirect contains "More Than One" request & response cycle	Forward contains "ONLY ONE" request & response cycle
Slower in operation	Faster in operation
Redirect happens on "Response Object"	Forward happens on "Request Object"

Redirect & Forward Few Cents ...

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

- If URL in Browser Changes means it MUST be a Redirect (especially Domain Name)
- For Internal URL either we can make use of Redirect OR Forward but "Forward" is preferred
- For External URL we left with no choice we MUST make use of Redirect
- In other words, If one web application wants to communicate with another "via Web Browser without any user intervention", then Redirect is the ONLY Way
- In this case both the web applications can transfer the data using "Query String"
- We cannot Redirect / Forward the request to more than one URL at a time

Include

- "include() Method" of RequestDispatcher helps us to Include the Response of an another Internal Resource (Static / Dynamic) into Servlet
- When we Include the content of one servlet into an another, it will include the response of "corresponding overridden version of doXXX() method" in that servlet
- We Cannot Include the response of "External Resource" into the Servlet
- Unlike Redirect & Forward, we can include more than one resource/URL at a time
- Include helps us to reuse the Internal Resources with that its "easy to maintain the Web Application"

Cookies

- Cookies are little piece of information in the form of name-value string pair exchanged between browser & server

- Cookies are Created @ Server/Servlet side but Present @ Browser side. Hence Cookies are "Browser Dependent“

- To Create a Cookie @ Servlet side, simply create an instance of "javax.servlet.http.Cookie" by passing name & value as String to the constructor

```
Cookie nameCookie = new Cookie("myName","Praveen");
```

- To send a Cookie as part of the Response, invoke addCookie() method on Response Object

```
resp.addCookie(nameCookie);
```

- Browser before sending the Request, it performs the below activities (automatically)

1. It takes "Domain Name" from request
2. looks out for Cookies within it's own memory
3. If present, it attaches cookies to Request
4. If NOT present, it simply sends the Request to Server without Cookies

Cookies Continued …

- Hence Cookies "optional" both in HTTP Request & Response. If present, they will be present in “Header" part of HTTP Request / Response

- To get Cookies from Request, invoke getCookies() method on Request Object

```
Cookie[] receivedCookies = req.getCookies();
```

- Cookies are the One & Only way in Servlet API, If we want to store any information in "User Browser"

- There are two types of cookies,

1. Non-Persistent Cookie
2. Persistent Cookie

- **Non-Persistent Cookie (it's default):-** Non-Persistent Cookie lives as long as Browser is kept open. Once the Browser is closed, cookie disappears

- **Persistent Cookie:** Persistent Cookie will be present in Browser even after Browser is Closed till it meets the expiry time

Attributes

- Attributes helps us to pass "Object" has an information from One Servlet to an Another Servlet

- Attributes are "name/key = value" pair where "name/key" is a "String" and "value" is a "java.lang.Object" i.e, "any java object" can be an attribute value

- Attributes are the "One and Only way", in Servlet API, to pass information in the form of "Java object" from one servlet/JSP to another servlet/JSP

- Attributes are of 3 types

1. Context Attributes (Application Scope)
2. Request Attributes (Request Scope)
3. Session Attributes (Session Scope)

- Following methods are present in above 3 Objects, which can be used to "get, set or remove" attributes

1. setAttribute(String name, Object value)
2. getAttribute(String name)
3. removeAttribute(String name)

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

- Servlets helps us to develop "Dynamic Web Application“ & there are 2 types of Dynamic Web Application

1. User Dependent Dynamic Web Applications
2. User Independent Dynamic Web Applications
3. Both User Dependent & User Independent Dynamic Web Applications

1. User Dependent Dynamic Web Applications:-

- These are the Web Applications where "Response Data is specific to a User”. Ex:- Gmail, Facebook, Twitter, Linked-In, etc.,
- Login & Logout is "Must" for these Web Applications
- Also these web applications MUST know "from which User it's getting the request“ to generate response

2. User Independent Dynamic Web Applications:-

- These are the Web Applications where "Response Data is independent of the User“. Ex:- Google Search, Google Maps, YouTube, Any Company Related Web Applications (www.testyantra.com) etc.,
- Login & Logout is "Optional" for these Web Applications
- Also these web applications Need Not know "from which User it's getting the request“ to generate

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

3. Both User Dependent & User Independent Dynamic Web Applications:-

- These are the Web Applications where "**Some Responses** Data is specific to a user" and "**Some Responses** Data is independent of user". Ex:- BookMyShow, Amazon Prime, Flipkart, etc.,
- Login & Logout is "Must" for some functionality of these Web Applications but Login & Logout is optional/not required for some other functionalities

Thumb Rule :-

- If we must need to Login & Logout to access the web application it MUST be a "User Dependent Dynamic Web Application"

- If we must need to Login & Logout to access certain functionalities of web application then, it MUST be Both User Dependent & User Independent Dynamic Web Applications

Why HttpSession?

- HTTP is a "Stateless Protocol" i.e. it doesn't maintain a relationship/state between requests. Each request is un-related to any previous requests
- Also HTTP don't help web application to uniquely identify the user
- Hence even if user sends sequence of requests to web application then it will not able to identify, those are from the same user. Also web application will not be able to relate them
- To address to this problem, Servlet API provides "HttpSession functionality". With HttpSession we can overcome the above problems

Thumb Rule:

Any web application which has Login & Logout then we MUST make use of HttpSession functionality

HttpSession

- A session, w.r.t web application, is a time difference between Login & Logout
- "HttpSession" is a functionality provided by Servlet API which helps web application to

1. Uniquely identify the user
2. It maintain "State / Relationship" between requests with the help of "Session Attributes"
3. Avoids Authentication for each Request
4. Tightly couples the different pages of the web application

- To make use of "HttpSession" functionality we must follow 3 Steps

1. Create a Session (Only Once; during Login)
2. Validate a Session (Many Times; for every request)
3. Invalidate a Session (Only Once; during Logout)

- HttpSession uses one of the following two mechanisms to handle session:

1. Cookies (it's default)
2. URL Re-Writing

HttpSession Related Methods

1. **HttpSession HttpServletRequest.getSession()** and
2. **HttpSession HttpServletRequest.getSession(boolean create)**

Creating a New Session Object Means:

- Container generates "Unique ID"
- Container maintains the Status of this "Unique ID" as "Active" @ Server Side
- Container sends this Unique ID to User in the form of Cookie along with the Response where, Cookie Name = "JSESSIONID“ and Cookie Value = "Unique ID"
- Container caches this Session Object in it's cache where, Key is "Unique ID" & Value is "Corresponding Session Object"

NOTE:

Container does All the above steps when we create HttpSession Object [i.e. when we invoke getSession() or getSession(true)] & it happens behind the scene

HttpSession Related Methods

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

3. void HttpSession.invalidate()

Invalidates the current session & garbage collects the associated session object

4. void HttpSession.setMaxInactiveInterval(int timeInterval)

- Specifies the valid session time, in seconds
- A negative time indicates the session should never timeout
- This method helps us to set different Session Time Out for different types of users

5. String HttpServletResponse.encodeURL(String url)

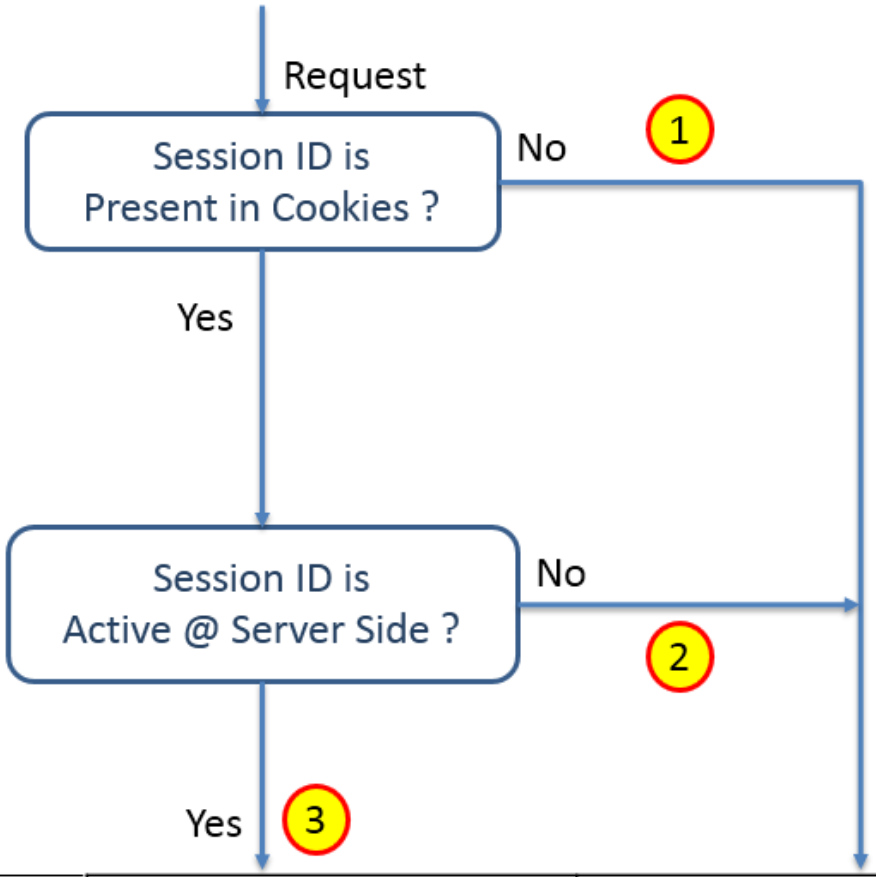
This method encodes the specified URL by appending session ID to it

Session ID Validation

Flow 1 :- Session ID NOT Present

Flow 2 :- Session ID Present but In-Active

Flow 3 :- Session ID Present and Active



getSession(true)	Returns the **Current Session Object**	Returns the **New Session Object**
getSession(false)		Returns **NULL**

Steps to work with Session

TESTYANTRA
SOFTWARE SOLUTIONS (INDIA) PVT. LTD.
EXPERIENTIAL
learning factory

1. Create the Session

Whenever **user Login** we have to create Session & it happens **ONLY ONCE** during the session

2. Validate the Session

After Login for every request we have to validate Session & it happens **MANY times** during the session

3. In-Validate the Session

Whenever **user Logout** we have to invalidate Session & it happens **ONLY ONCE** during the session

Create a Session

Whenever user login & **after successful authentication**, create session for the First Time.

```
HttpSession session = req.getSession(true);
```

OR

```
HttpSession session = req.getSession();
```

Validate the Session

- Once Session is created, for subsequent requests validate the session
- Note: Any Servlet/JSP which get the request from Browser, after successful login, then it should have "Session Validation Logic"

```
//Get the Current Session Object
HttpSession session = req.getSession(false);
if(session == null) {
        //Invalid Session !!!
        //Generate "Login Page With Error Info as Response"
}else{
        //Valid Session ...
        //Generate "Proper Response"
}
```

Invalidate the Session

A Session get Invalidated in Following ways

I. When Application / Server goes down

II. When user logout of the application

When user wants to logout of the application, then invoke invalidate() method on the Current Session Object

```
//Get the Current Session Object
HttpSession session = req.getSession(false);
if(session != null) {
        session.invalidate();
}
//Generate "Login Page with Success Message“ as a Response
```

III. When user is in-active for configured amount of time

There are two ways to configure session time out

1. In web.xml
2. In Program

Invalidate the Session Continued ..

1. In web.xml

```
<session-config>
        <!-- Time in Min (ex: 7 Days)-->
        <session-timeout>10080</session-timeout>
</session-config>
```

2. In Program

```
HttpSession session = req.getSession(true);
//Time in Seconds, ex: 7 Days
session.setMaxInactiveInterval(7*24*60*60);
```

URL Rewriting

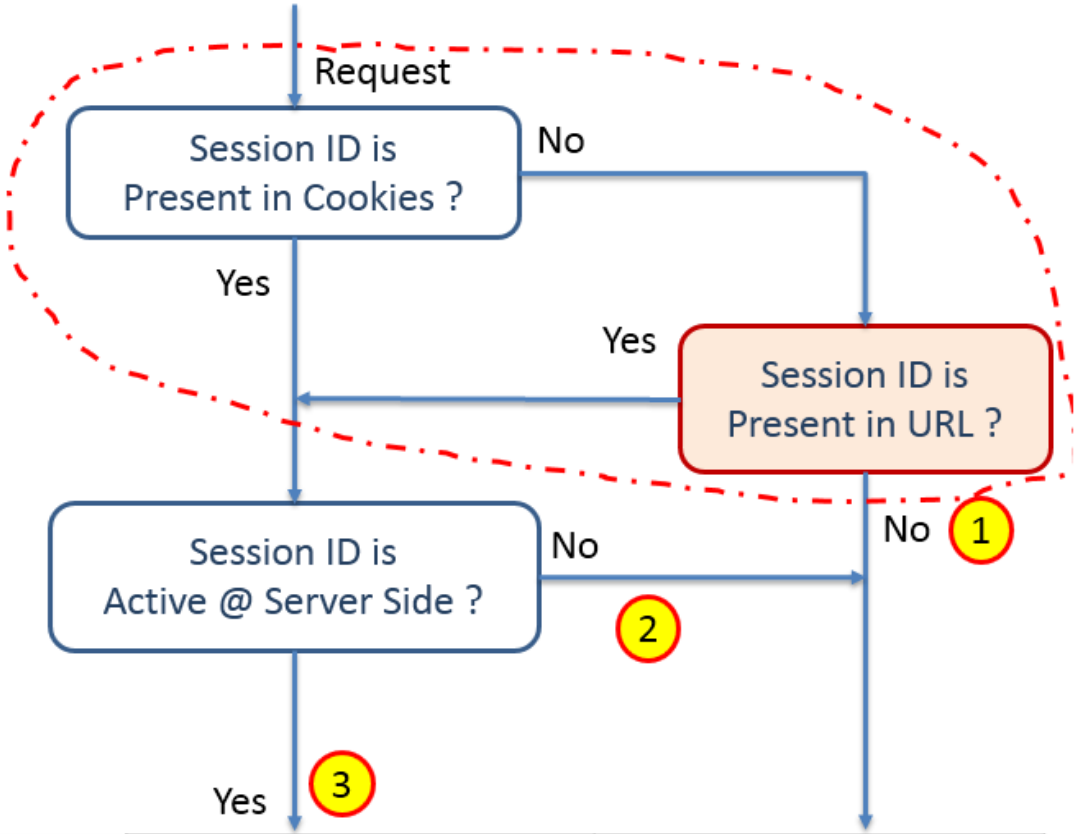
- While handling user session "URL Rewriting" comes into picture ONLY if we encode the URL's with "Session ID"
- If we encode the URL's, then Container ALWAYS first attempt to use the Cookies to get Session ID & fall back to URL Rewriting only if Cookie approach fails
- In real time scenarios we will not make use of URL Rewriting to handle session (because of "Session Hijacking" issue)

Session ID Validation ...

Flow 1 :- Session ID NOT Present in the Request

Flow 2 :- Session ID Present but In-Active

Flow 3 :- Session ID Present and Active



getSession(true)	Returns the **Current Session Object**	Returns the **New Session Object**
getSession(false)		Returns **NULL**

TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Happy Learning !!!

No.01, 3rd Cross Basappa Layout, Gavipuram Extension,
Kempegowda Nagar, Bengaluru, Karnataka 560019

praveen.d@testyantra.com

www.testyantra.com

EXPERIENTIAL
learning factory