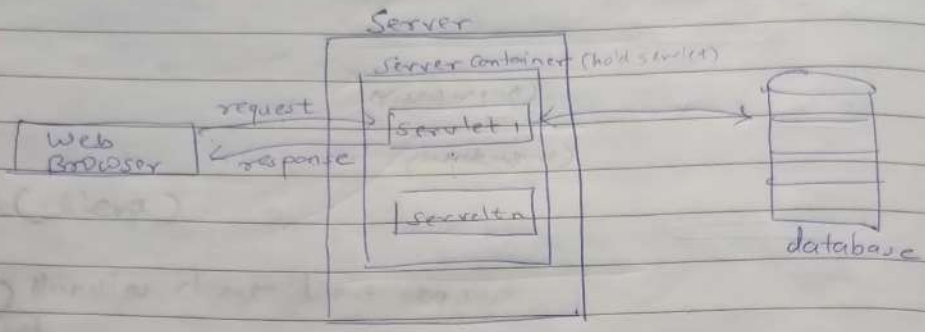
20/12/23

Servlet



- Servlet is a server side technology used to handle client request then process that request, generate response and give it back to the client.
- This response can be static or dynamic

- What is Servlet?

→ * API:

- Servlet is an API, generally used to develop web application.
- All the classes and interfaces related to Servlet API are present in 'javax.servlet' package

* Interface

- We also say Servlet is an Interface API, it has few methods which are present in it any class who implements Servlet interface need to give implementation to all those methods.

* Program

- Servlet is a program which is going to run on Server, meaning we are going to write servlet program while building web application, that web application is going to run on Server, in Server

For building web app → write servlet pgm → going to be run on server → Server → going to write → servlet classes → going to be triggered whenever there is a request related to that particular servlet → produce response & send it back to client

Server we are going to write Servlet classes. and these classes are going to be triggered whenever there is a request related to that particular servlet., then that particular servlet class will produce response and send it back to client

A servlet is a class which acquire property of Server creating response upon request.

21/12/23

Tables

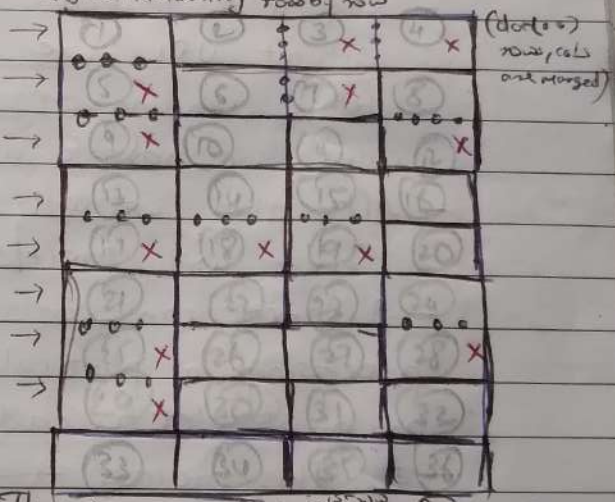
Given Table

①	②	③	④
① Day	Seminar		
②	Schedule		Topic
③	Begin	End	
④ Monday	8:00am	5:00pm	Introduction to XML
⑤			Validity DTD and Relax NG
⑥	8:00 am	11:00 AM	X Path
⑦ Tuesday	11:00 am	2:00 PM	
⑧	2:00 PM	5:00 pm	XSL Transformation
⑨ Wednesday	8:00 am	12:00 PM	XSL formatting Objects

For building this table

① Calculate total no of rows (Vertical) & cols (Horizontal)
Here: total no of rows: 9
total no of cols: 4

② Draw table containing 9 rows & 4 cols & give numbering row by row



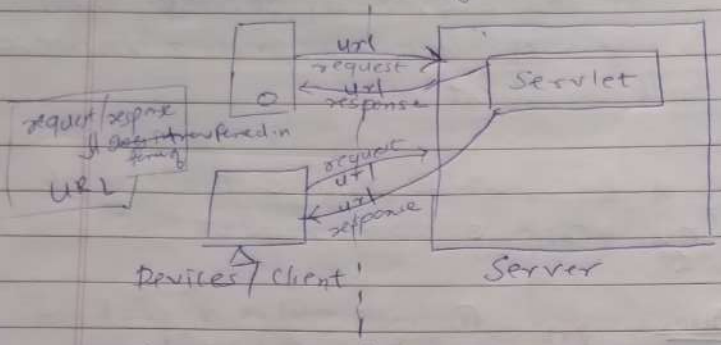
→ Then in above table at (11) we see that ①⑤⑨ get merged (cols are merged) so put dashed lines ① 2 ⑨ (dotted rows)
→ at row ② we see ③ ④ are merged (rows are merged) so put dotted in 3, 3, 4 & (rows ③ ④)
→ Don't consider block which are crossed

22/12/23

How Does Servlet Works

- Protocol set of rules
- Diff btn http & https
& Diff btn other protocols

* Servlet is a program which runs on Server



- This is a client-Server Architecture
- Request & Response are transferred in form of ~~Url~~
- Without URL, we cannot send request
- While sending request, we need to follow some rules; we have to use http protocol
- After this, we will sent request to server,
- Server will process that request
- A Server can contain more than one servlet, processing a particular request means to decide which servlet to call or trigger
- ~~Ex~~ This process ~~to take~~ is handled by servlet Container, ~~servlet container~~ will
- Servlet Container will reciever request and verify the request mapped to which servlet
- Servlet container will take 'http' request and it is going to create a request object. as we are dealing with Java Application (bcoz servlet is Java appl^n)
- 'http' is a protocol so, it need to be converted into java object, then that java object will be passed to servlet

*:- Servlet container will take an http request and convert it into http servlet request object.

Servlet Container will instantiate the servlet object, then it is going to pass http servlet request object into Servlet class

- In return, servlet will return a response & it will be in form of object (http servlet response)

keyword: - http post request
- response
- url
- http servlet
- servlet container
- http
- http servlet request object
- http servlet response

what is Servlet, generic servlet?

Servlet <I> : - It is an Interface
- It is present in javax. servlet package

Servlet <i>
Present: javax. servlet package
↑
implement

solid line = represent 'extends'
dotted line = represent 'implements'

My Servlet <c>
(user defined servlet) (here we implement all methods
(we need to provide implementation of all
Methods of Servlet interface)

Methods in servlet interface: - how many methods - method names - description - Exceptions

1. public void init (ServletConfig config) ServletException
- This method is generally called to initialize members of the class. (user defined class (Servlet))

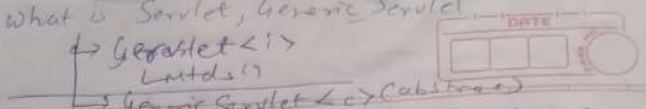
2. public void service(ServletRequest request ServletResponse response) ⟹ ServletException, IOException
- This method will take request and provide response

3. public void destroy()
- This method is called to unload servlet classes.

4. public ServletConfig getServletConfig()
- This method is called to get configuration information
(Configuration object is one for servlet)

5. public String getServletInfo()
- This method is going to return String as servlet info
(gives servlet information in form of String)

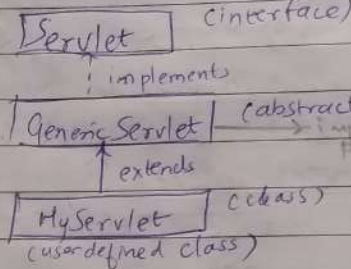
What is Servlet, GenericServlet



- Generally whenever we are building a Servlet class we are not going to implement Servlet interface
- Instead, we are extending 'our' (user defined) servlet class to GenericServlet class

GenericServlet :- 'GenericServlet' class is implementing 'Servlet' interface (abstract class)

- (All Methods of 'Servlet' interface will be inherited in 'GenericServlet',) in ('GenericServlet' implementation to those methods is provided except for service())
- service() stays as abstract Method in 'GenericServlet')



- If you want to create a servlet it is recommended to inherit 'GenericServlet' class & override service()

- In 'GenericServlet' along with 'Servlet' interface methods, we have few more methods to get more information about 'Servlet'

• Methods in GenericServlet

- There are total 12 Methods in GenericServlet'

1. public void init() throws ServletException
   - It is used to initialize the members of servlet class
   - If exception occurs: interrupts Servlet normal operation

2. public ServletContext getServletContext()
- Returns a reference to the servlet context in which servlet is running
NOTE: ServletContainer: one for the application
- Returns ServletContext object passed to this servlet by init method
- This method is present in the GenericServlet class
- It can be called by Servlet itself
- It can be called at any point during the servlet's life cycle

3. public String getInitParameter(String name)
- Whatever request we send (like form information (name, email, etc)) we can read all those parameters by calling this method
- Parameters: name - a String specifying the name of initialization parameter
- Return: a String containing the value of initialization parameter

4. public Enumeration<String> getInitParameterNames()
- All parameter names can be retrieved using this method
- Parameters: —
- Returns: Enumeration of String type
Enumeration: It is a string object containing names of servlet's initialization parameter

5. public String getServletName(): returns the name of Servlet instance (object)
Return: String information of servlet instance

6. public void log (String msg)
- Writes the specified message to the servlet log file
- Parameters: String specifying the message to be returned in log file

- How to create Servlet Maven project:
1. Right click on project explorer > select Other (search for maven) > select maven project (click on next) > again click next > filter: maven-archetype-webapp (select 1.4) > click on next > provide artifact id, group id and finish
2. open 'pom.xml' > find dependencies tag > here we have to add servlet dependency > on internet search for 'MVN repository' > in that search for 'servlet api' > in that look for Java Servlet API > in that select version 4.0.1 > in that copy code under Maven tab > paste it in pom.xml > save the file > then, project will build
3. Go to java resources folder, here we might not see 'src main' folder (1st place var as to fo) > so right click on project > click on properties > java build path > order & export tab > check (select all) maven dependencies jre System libraries > apply and close > now you can see 'src/main/java' folder.
4. Right click on project > maven > update project > check on force update release Snapshot > click ok

the servlet log file
message to

ct:
select Other
project (click on
aven-archetype-
ovide artifact id,

tag> here we
internet search
-api'>
in that select
Maven tab
file >

e might not
)>so right
ra build path>
ndencies
> now you

e project >
t> click ok

- Create, register and run servlet
1. first follow the four steps from 'how to create servlet Maven project'
2. Now, we are going to create a servlet class
3. To create servlet class: open 'java resources' folder → inside that select & right click on 'src/main/java' folder → create new class file inside desired package or the class will be stored inside default package → provide class details.

Add unimplemented Method

Abstract class where service mtd is abstract so we need to override service method

```
public class ServletClass extends GenericServlet {
    @Override
    public void service(ServletRequest req,
    ServletResponse res) throws ServletException,
                                IOException {

    }
}
```

4. After creating servlet we need to register that servlet
5. To register servlet, goto project → 'src' → 'main' → 'webapp' → 'WEB-INF' → 'web.xml' → open

*web.xml

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, INC.//DTD Web Application 2.3
//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <display-name>Archetype Created Web Application
  </display-name>
  <servlet>
      <servlet-name>ServletClass Name</servlet-name>
      <servlet-class>Qualified Class Name</servlet-class>
  </servlet>
  <servlet-mapping>
    <!--servlet name should be same as servlet name
                                in servlet tag-->
```

<servlet-name>ServletClassName</servlet-name>
<url-pattern>/msg</url-pattern>
</servlet-mapping>
</web-app>

* To run servlet program:
- right click on project → run as → run on server
(it will ask you select which server to run on)
→ select server apache tomcat server version
→ finish

* PrintWriter: - PrintWriter is a class in java that is a part of java.io package
- It provides methods to write formatted text

* Methods for writing data
- PrintWriter provide a variety of print() & println() for writing different kind of data including integer, floating point number, character & string
- Eg: print(int a);
print(String s);
print(Double d);

* Read data in GenericServlet

servlet-name>
>

run on server
to run on)
rver version

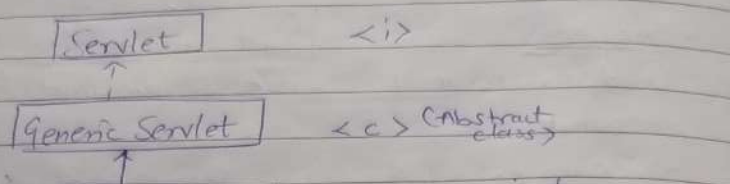
in java that is

rmatted text

print() &
of data
umber,

- Read form data in Generic Servlet (02-project)
* request.getParameter(String name)
→ • Description: - The request.getParameter(String name) Method is used for retrieval of values of parameter from the client's request
- In context of a servlet, this Method is beneficial for extracting data submitted through html forms.
• Argument (Parameter name):
- The Method takes parameter name as an argument, representing the name of the parameter whose value you want to retrieve
• Return type:
- The Method returns a string that represents the value of specified parameter.
- Null is returned for absent parameter. The Method returns null if the specified parameter is not present in the request
■ Use in HTML forms:
• Description: - In context of html forms the parameter names are typically associated with the name attribute of form element
- Eg <input name = "parameter name">

HTTP Servlet and its Methods

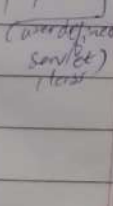


- HTTP Servlet is a abstract class
- It is present in javax.servlet package

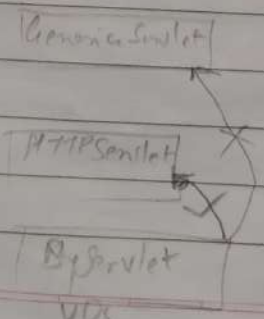
#NOTE *HTTP Servlet is an (abstract class) even though it (does not have any abstract method) in it

(why: ??

Ans: - The HTTP Servlet class in java is (designed to be base class for creating servlets) that (handle HTTP request)
- Although it does not contain any abstract methods, it is (marked as abstract class) because, it is not meant to be instantiated (creating object) directly
- Instead, (developers are expected to create subclass by extending to HTTP Servlet) and (providing their own implementation (complete implementation) for the methods they want to override)



* If we want to (create Servlet that handles HTTP request particularly, generally it is not recommended to inherit GenericServlet, instead we need to inherit (ability of) HTTP Servlet class)



Q. If we already have Generic Servlet, then why to opt for HTTP Servlet?

→ - Generic Servlet and HTTP Servlet serves different purpose in Java Servlet API and choice between them depends on the requirement of web application

* Generic Servlet: Generic Servlet is a more General purpose class, that can handle request and response in a protocol in generic manner
- It is not specific to HTTP & can be used for other protocols as well

* HTTP Servlet: - HTTP servlet is a subclass of Generic Servlet
- It provides specific support for handling HTTP request and response
- It includes Methods such as doGet, doPost, doPut, doDelete for handling different HTTP Methods for handling request and responses.
- HTTP Servlet is designed to simplify the development of web application, that specifically deals with HTTP
- It abstracts away many details of the HTTP protocol, making it easier to work with higher level concepts such as request parameters, session management and url patterns

• Methods

* Methods:

1. doGet

Header: protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException

def: - This method is useful when dealing with HttpGetRequest
- Data will be attached to url resulting in low security

2. doPost

header: protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws: ServletException, IOException

- Data will be pasted into HTML body, ensuring better security

#NOTE: We may replace service() with doGet, doPost, doTrace, doPut ... but init() & destroy() will only be executed once

- In HTTP Servlet doGet & doPost methods are designed to handle HTTP GET and POST request respectively
- The main difference between these methods lies in the type of HTTP request they are intended to process and the way they handle request parameter

* doGet(): - doGet() is used to handle HTTP GET request
- It is called when the client sends a request using HTTP GET method
- Typically, GET request are used for operations that do not have side effects, such as retrieving information or displaying a resource
- In GET request data is appended to the URL as query parameters
- These parameters are visible in URL itself

* doPost() :- doPost() is used to handle HTTP POST request
- It is called when client sends request using HTTP Post()
- POST request are often used for operations that have side-effects, such as submitting form or updating data on server.
- The data is sent in the body of the request making it more suitable for large amount of data
- In POST request data is not visible in URL but is instead send in the request body

doGet()	doPost()
- handle HTTP GET request	- handle HTTP POST request
- client sends request using HTTP GET()	- Client sends request using HTTP POST()
- It is used for operations which doesn't (Modifies data in database) have side-effects	- It is used for operations which have (Modifies data in database) sideffects.
- Eg: retrieving information displaying a source	- Eg: Submitting form, updating data on server
- Request data is appended to URL as query parameters	- POST Request data is sent to the body of request
- It is not suitable for large data	- It is suitable for large data
- GET request data are visible in URL	- POST request data are not visible in URL

<a> attribute href => URL

"www.example.com?user=abc&pass=123"

- **Query Parameter** :- Query Parameter in context of servlet refers to the parameters that are included in the URL of http request.
- These parameters are added to the end of the URL after a (?) question mark
- These parameters are seperated by ampersands (&)

http://localhost:8080/04-send-form-data-doGet/fetchData?user=abc&pass=12 (maven proj name; servlet name; (key, value) & (key, value))

- Query parameters are commonly used to send data from the client to server in GET request

*** Retrieving Query Parameters in servlet**

- Servlet can retrieve Query Parameters using HTTPServletRequest Object (req) and getParameter()
- The getParameter() is used to retrieve value of specific parameter by providing its name

String user = req.getParameter("user");

*** format of Query Parameter**

- Query Parameters are appended to URL of http request after (?) question mark
- Multiple parameters (multiple names like user, pass) are seperated by ampersands (&) (key/value pairs are joined by &)

eg: http://example.com/servlet?name=value1&age=value2

(by using @WebServlet, @WebServlet(value="/page-a") => we dont need to write any tags <servlet> <servlet-mapping> in web.xml (Enterprise Edition))

@WebServlet :- This is a standard annotation in Java EE for declaring servlets

- It is a part of the javax.servlet.annotation package

Pgm: 06-query parameter

@WebServlet(value="/page-a") :- The 'value' attribute specifies the URL pattern (servlet should response) to which servlet should respond. In this case, servlet is mapped to the URL pattern "/page-a"

(url pattern [a href and the servlet name is default] same @WebServlet("/ forward slash) (value=/name) => same)

This annotation provides a convinient way to configure servlet without the need of explicit enteries in web.xml
This commonly used in Modern Java web application.

Servlet Life Cycle

- From the Birth to Death of servlet all operations are taken care by web container

1. Load Servlet class
   - The first step of servlet life cycle is to load servlet class into memory
   - The class loader is responsible for loading servlet into memory, when servlet class recieves its first request.
   - From second request onwards, it will not load again
2. Creation of Instance
   - Web container will create instance for servlet class using which, we can access all methods of the object.
   - Object creation will take place only once in entire lifecycle of servlet
3. Initialization :- The web container will initialize the parameter of instance of servlet by calling init()

#NOTE: All above steps (Load servlet class, Creation of Instances, Initialization) will happen only 'once' in Servlet life cycle

In Servlet to put img => src → main → webapp → make folder resources & in that put image

while giving src in img src = "/resources\...jsp"

Task: Escape Sequence character → Java, absolute path, relative path

2/1/24

4. Invoking ~~Service~~ service ()
- Web container will (invoke) service().
- The service() will be (invoked) for every request

5. Destroy the Servlet
- Web container will (invoke) destroy() (before unloading Servlet class) i.e Servlet will be destroyed
- This operation will happen (only once) in Servlet Lifecycle

(default => index.jsp) → to change this welcome file use this

- Welcome ~~@~~ File list
- Whenever we are launching web application and want a specific page to be opened ~~and~~ at that time, we need to do some modification in 'web.xml'

- <welcome-file-list> tag will help to set which page needs to be opened when ~~an~~ an application is launched.
- In <welcome-file-list> we can configure multiple "welcome" files (<welcome-file>)

```
<web-app>
    <welcome-file-list>
        <welcome-file> A.html </welcome-file>
        <welcome-file> B. html </welcome-file>
        <welcome-file> C. html </welcome-file>
    </welcome-file-list>
</web-app>
```

Margin notes: launch a web app → specific page opened; some configuration in web.xml; welcome-file-list → help specify which page to be opened; multiple welcome file → work else-if ladder; None of the files mentioned in welcome file is present → Error 404; Don't specify welcome file list. Use case → Provide default landing page, Entry pt → User friendly URL

- If we have multiple 'welcome file' in 'welcome file list', It will work as 'if-else' ladder (else-if ladder).

- If page A, B, & C are not available then it will throw 'ERROR 404 - Page Not Found'

- If we don't specify welcome-file-list at all, by default 'index.jsp' page will open when an web application is launched.

* Use case of welcome file list
  - It is primarily use to provide a default entry point or landing page for a web application

  → Default Landing Page:
  - The welcome file list allows you to specify one or more files that should be considered as default landing pages; when a user accesses the root URL
  - for eg: If you have 'index.html' file as one of welcome files, accessing 'http://example.com/' would display the content of 'index.html'.

  → User friendly URL
  - It helps in creating user friendly URL without specifying exact file name in the URL
  - Instead of 'http://example.com/home.html' we can configure 'home.html' as a welcome file and user can access it via 'http://example.com/' link

3/1/24

Load on Startup (tag used in web.xml) (configuration)

- In servlet based-web application, the load on startup element in web.xml is used to specify that the servlet shall be loaded, when the web application starts, rather than waiting until the servlet is first accessed.

[PCI-AU]

* Purpose:
- load on startup is used to ensure that Servlet is initialized and its init() is called when the web application starts.

* Configuration in web.xml
- You can specify load on startup for a Servlet (tag used inside servlet) in web.xml by providing integer value

in web.xml

```
<servlet>
    <servlet-name> Your Servlet </servlet-name>
    <servlet-class> com.example.YourServlet </servlet-class>
    <load-on-startup> 1 </load-on-startup>
</servlet>
```

- The integer value (1/0) in load-on-startup tag indicates the order in which servlet should be loaded. (cannot load two servlets at a time)
- Servlet with lower numbers / lower value are loaded first (lower no: higher priority)
- Here, we cannot give 'negative' or '0' number/integer (natural nos.)
- It only accepts positive integers.

, the load on
is used to specify
aded, when the
- than waiting
d.

ure that
init() ~~met~~
n starts.

for a
integer value

et-name>
et</servlet-class>
up>

artup tag
should be

alue are

umber/integer

Servlet version: 4.0.1
" 3.0
" 1.0 & further

* Initialization order
- Servlet with lower load on startup order value are loaded and initialize before servlets with higher order

* If multiple servlet have same load on Startup value, the container determines the order.

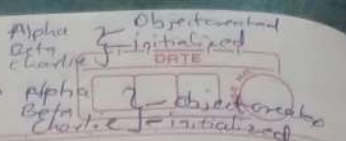
* Advantages:- Using load on startup, can be benificial for servlet that need to perform initialization task or setup resources when application starts. (time saving)

* Common Use Case
* Initializing database connections, establishing communication with external resources or loading configuration settings can be common usecases for utilizing load on startup.

- In Summary, load on startup is used to control the initialization order of a servlet in web application
- It ensures that specific Servlets are loaded and initialize when the application starts. allowing them to perform setup tasks before handling any client request
- load on startup is optional, if not used then, a particular servlet will be loaded & initialized only when requested

08- load on Startup web.xml: only one time

09- load on startup web.xml : two times



Q. Can we do load on startup using annotation?

→ - Yes, we can achieve load on startup using annotation

(How) - The @WebServlet annotation provides `loadOnStartup` attribute, that allows you to specify the order in which Servlet should be initialized during application startup

```
@WebServlet(value = "/your-url-pattern", loadOnStartup = 1)
public class YourServlet extends HttpServlet {
    // Your servlet code
}
```

Q. What happens if we give negative or zero number in load on startup

→ - Setting negative or zero value for load on startup in servlet configuration is generally considered invalid and may result in unpredicted behaviour

- following are scenarios that might happen

- negative value:
  - Servlet container expects positive integer values in load on startup tag.
  - If negative value is provided, it might be ignored or treated as an error or interpreted in an unexpected way
  - The behaviour is not standardize across the servlet containers and different containers might handle negative values differently

- ~~Zero value~~

- Zero value: = Zero value is not a valid value for servlet configuration for load on startup tag
  - Servlet takes zero value internally as error, ignored
  - Behaviour can vary in different servlet containers
  - It is best practice to provide positive integer values.

- Request Dispatcher: Project: BMI
- RequestDispatcher (what?) → RequestDispatcher interface in java servlet (provides a way to forward the request from one servlet to other servlet.) or (include the content of another resources (servlet, JSP, html) in the response.)
  - It is mainly used for server-side forwarding during request processing
  - It is basically helps us to dispatch the request

Q. Dispatch the request from where to where?

→ - From one servlet to other servlet (S - S)

or

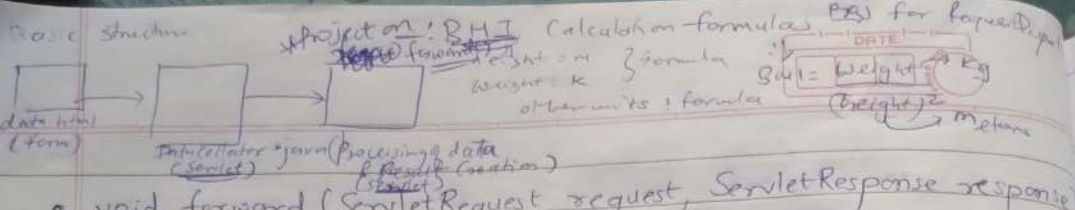
- From one servlet to other JSP / HTML (S - J)

or

- From JSP to JSP (J - J)

or

- From JSP to Servlet (J - S)



- void forward (ServletRequest request, ServletResponse response) throws ServletException, java.io.IOException

- forwards a request from a servlet to another resource (servlet, JSP file, HTML file) on the server

(It has to get RequestDispatcher obj)

* Obtaining a RequestDispatcher:

- The RequestDispatcher can be obtained using getRequestDispatcher() using ~~HTTPRequest~~ HttpServletRequest 'req'

RequestDispatcher requestDispatcher = req.getRequestDispatcher ("Target_Servlet_Name");

- This method take relative path of resource as an argument (Servlet Target)

* Use cases:

- Suppose you have Servlet 'A' that process some data then forward the request to another Servlet 'B' to display those results.
- This helps in Modularizing application logic and concerns.
- Forwarding is used when you want another Servlet or resource to handle the request entirely

* forwarding a request

- ~~forwarding~~ is achieved using forward() of RequestDispatcher

requestDispatcher.forward(req, resp);

#NOTE: - After the forward the processing of original Servlet stops and control is transferred to the destination Servlet

* ID- RequestDispatcher-include() (projector includes)

include

- void include(ServletRequest request, ServletResponse response) throws ServletException, IOException

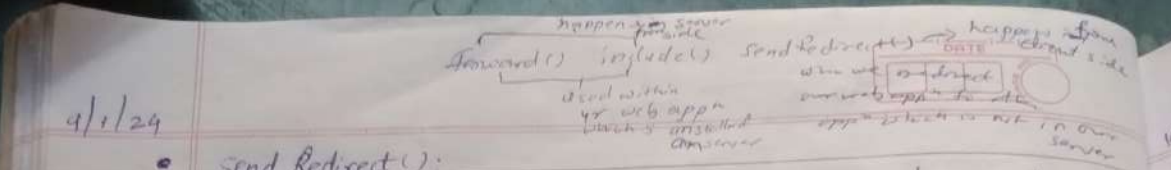
- Includes the content of a resource (servlet, JSP page, HTML file) in the response.

- Way to get RequestDispatcher object (same as forward) Obtaining

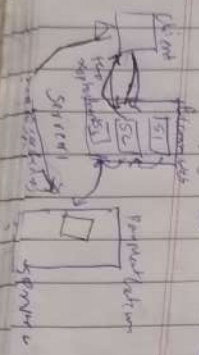
* Use case:

- include() is used when you want to include the content of another servlet or resource in the response

9/1/24



- Send Redirect():
- The Send Redirect method in servlet is used to redirect a client's request to another resource, either within the same web application or external web application (External URL).
- This method is a part of 'HTTPServletResponse' interface



* Redirecting to another resource:
- The SendRedirect() sends a temporary redirect response to the client, instructing the client's browser to make a new request to the specified URL.

```
(HTTPServletResponse resp)
resp.sendRedirect("url");
```

* URL's: - The URL can be 'relative' to current servlet context or an 'absolute' URL including external URLs.

```
resp.sendRedirect("/context/newPage.html"); //Relative URL
```
(this URL is for resource which is in same appn)

```
resp.sendRedirect("https://www.example.com"); //Absolute URL
```
(this URL is for external webpage present in other server

* Client side Redirect: - Unlike Server side forward (RequestDispatcher.forward()), SendRedirect() triggers a new request from the client's browser

10/1/24

Send Redirect
Redirect to new resource
URL
clientsideRedirecting → RequestParameter → Common Mistakes → not to do considerations

10/1/24

* Request parameter :- Request Parameters can be included in the URL and they will be available to redirect the resource
- Eg: "https://www.google.com/search?q=" + cars + "https://www." + urlname + ".com"
request parameter

* Common Mistake :- Attempting to use 'requestDispatcher.forward(req, resp)' after 'sendRedirect(req, resp)' in the same request response cycle may lead to unexpected behaviour, these two Methods serve different purpose.

(Task: create the issue of forward and sendRedirect together)

* Consideration :- Use sendRedirect() when you want client's browser to initiate new request to different URL.
- Be careful, about URL paths, context path & whether URL is relative or absolute.

• Diff b/w request parameter & request attribute.

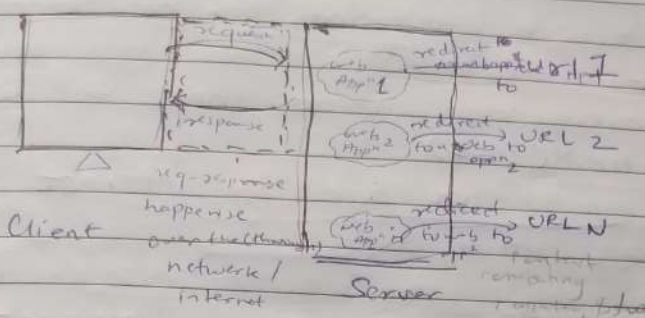
CROU - BR - PUC)

Difference between RequestDispatcher & sendRedirect

Features	RequestDispatcher (forward(), include())	sendRedirect()
Purpose	- Used for server-side forward and include operations. Requests are forwarded within same Servlet	- used for client-side redirection to a new URL.
Operates	Entirely on server-side	Initiates request from client-side
URLs	Works with internal URLs (within the same web-application)	Works with both internal as well external URLs (within the same web-app or even outside the web-app)
Browser URL	(base) URL in client's browser does not change	- URL in client's browser changes to redirected URL
Request Attribute (Shared b/w source & destination)	Allows sharing of request attributes between the source & destination	Request attributes are not shared between the source & destination
Performance	Generally faster as it involves internal processing within server	- Involves additional round trips to the client that may have slight performance overhead
Use case	- Suitable for forward and include operations where the destination is within the web-application	- Suitable for scenarios where you want to redirect client to a different URL, specially when the URL is outside of current web application
Client Involvement	Does not involve client's browser in redirection	Involves client's browser in making new request

live 2 proj simultaneously: start server w/o restarter

Client Server Interaction



* Server:- A server in context of computing generally refers to a (computer or software) that (provides services, resources or other functionalities) to (other computers or clients) through network

- The term 'server' can be used in various context including hardware, software and services

- Hardware Server:- A physical computer or a dedicated device, (designed to run server software) and (provide services to other devices) (through network)
  - Eg: web server, file server, dlb server & Mail server etc.

- Software Server:- A (software application or a program) that (runs on computer) and (provides specific services to client)
  - Eg includes web server software like (Apache or Nginx), Database Server software (PostgreSQL or MySQL.

- further, we can have multiple definitions of server as services, web server, application server, server farm, or data center

- Services
- Web Server
- Application Server
- Server farm, - Data Center

12/1/24

in client server interaction

- Multiple web applications can be stored on the server
- for every application, there will be an identifier (URL)
- Any application access by internet is called as web application

- whenever client is sending request it can contains URL*, Method type* & data*.

request side: URL, Method, Data

response side: Status code*, content type*, actual content*

- Response contains status code*, content type* and actual content*

* Content type :- It is an http header that indicates type of data contained in body of HTTP Response
- It specifies the media type of content being send

* Status code :- A status code is a 3 digit numeric code returned by server to indicate outcome of client's request made through web browser)
- These codes are part of HTTP (HyperText Transfer Protocol) Standards and are included in the response message header send by the server)
- Status code provide information about success, failure or other conditions related to request
- HTTP Status codes are divided into different classes (different series) each representing specific category of response

stored on the

ill be an

et is called as

uest it can

content type

that indicates

HTTP Response

nt being send

digit numeric

te outcome

browser)

xtTransferProtocol)

response

success,

request

different classes

esponse)

13/1/24

JSP - CS

x=(0-99)(x: includes digit from 0 to 9)

1XX (Informational) :- This status codes indicates that the server has recieve the client's request and it is (continueing to poressing it and will provide final response later

2XX (Successful) :- This status codes indicates that client's request was success-fully recieve, understood and aclepted

3XX (Redirectional) :- This status code indicates that further action is required to complete the request.
The client typically needs to take additional steps such as redirection.

4XX (Client Error) : This status code indicates that client seems to have made an error
- It could be due to issues like request with malfunction or unauthenized access.

5XX (Server Error) :- This status code, indicates that server has encountered an error while fulfilling the request
- It is (not client's fault) and server is responsible for resolving the issue)

Commonly encounteered HTTP Status codes

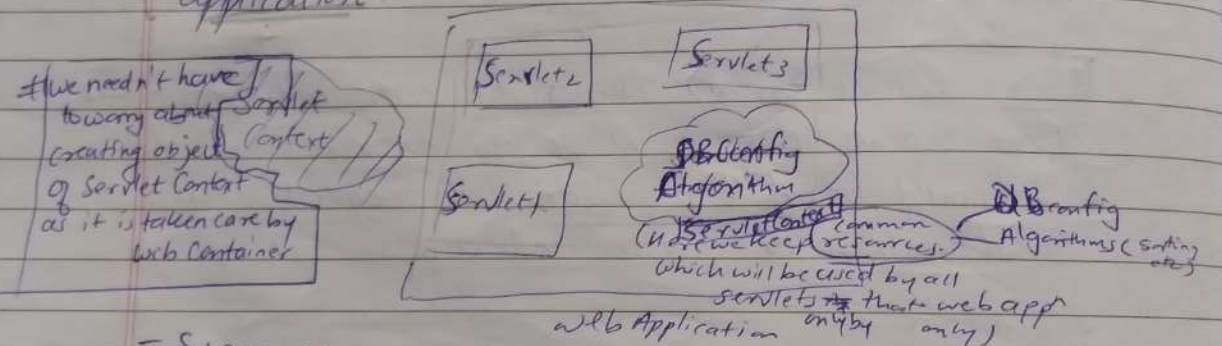
404 NOT FOUND :- The requested resource is not found on server

500 INTERNAL SERVER ERROR - A Generic Error Message indicating server has encountered an error or an un excepted condition.

* Actual content:
- It refers to meaningful information or data within a document, file, webpage or any other form of communication.

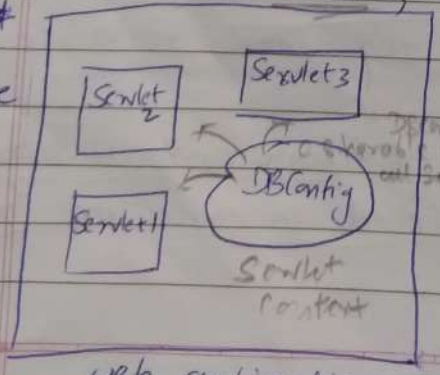
12/1/24

What is Servlet Context
- Servlet Context is an interface that represents a communication channel between servlet and the Servlet Container.
- It provides information about web applications environment and servers as central repository for sharing configuration informations, resources and communication among servlet in same web application.

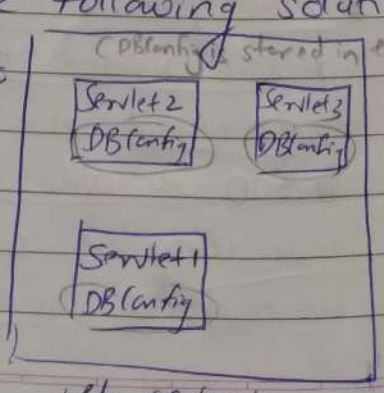


- Suppose (Explaination of diagram)
- Suppose we have web-app in that we have multiple servlets
- If we want to share same information among all Servlet (information like DB Configuration) which can be common through all servlets.
- In such case, we can have following solutions:

Soln 1st (Appropriate sol.)



Soln 2



- In web app ServletContext is an object

[1.0]

Q. what kind of object ServletContext is ⇒ ServletContext kind of object

[2.0]

Q. What is ServletContext and who is responsible for it?

→ (It is an object in webapp) and (web container will create ServletContext object.)

- As a programmer, we neednot to ~~worry~~ about object creation of it

* How to store value in ServletContext

- Whenever we are storing something in servletContext object we are going to store it in ~~key, value~~ form of key and value pair (key, value pair)

- These key value pairs are stored in 'web.xml'

~~<context-param>~~

```
<web-app>
    <context-param>
        <param-name> key </param-name>
        <param-value> value </param-value>
    </context-param>
</web-app>
```

- We can Multiple context-param (we can store Multiple key-value pairs)

- We can make difference bth these key-value pair by 'key'

Q. When does ServletContext object is going to be created?

→ As soon as we deploy application, ~~ServletContext~~ ServletContext object is going to be created.

11/1/24

Servlet Config

- It is an interface present in java servlet API that provides servlet with its instantiation parameters.
- It allows a servlet to access configuration information specified in web.xml during its instantiation.
- ServletConfig object is specific to that particular object

* Considering a webapp with Multiple servlets:
- Suppose I need to have some information which needs to be stored in a particular servlet in such case we can make use of ServletConfig
- Servlet Config object is one for a servlet
- Servlet Config object of Servlet 'A' is not visible to Servlet 'B' and Servlet 'c'.
- Servlet Config object of Servlet 'B' is not visible to Servlet 'C' and servlet 'A'
- Servlet Config object of Servlet 'c' is not visible to Servlet 'A' and servlet 'B'.
- for eg: If Servlet 'B' tries to access a key specified in servlet-config of Servlet 'A' then it returns 'null' to Servlet 'B'.

- web container creates ServletConfig objects which are available to that particular servlet.

- web container creates Servlet Config object when it instantiates servlet class

Web container → create ServletConfig object which is available to that particular servlet
When it creates servletConfig object
When it instantiates Servlet class

* How to store & retrieve values from the ServletConfig object:

1. Using web.xml

```
<web-app>
  <display-name> Archetype Created Web Application</display-name>
  <servlet>
    <servlet-name> PageA </servlet-name>
    <servlet-class> edu.ty.PageA </servlet-class>
    <init-param>
      <param-name> key </param-name>
      <param-value> value </param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name> PageA </servlet-name>
    <url-pattern> /page-a </url-pattern>
  </servlet-mapping>
</web-app>
  <servlet>
    <servlet-name> PageB </servlet-name>
    <servlet-class> edu.ty.PageB </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name> PageB </servlet-name>
    <url-pattern> /page-b </url-pattern>
  </servlet-mapping>
</web-app>
```

Diff b/h ServletContext & ServletConfig

2. Annotation Based Servlet-Config:

```
@WebServlet (
    urlPatterns = {"/MyServlet"}
    initParam = {
        @WebInitParam(name = "paramName",
            value = "paramValue")
    }
)
```

* What if we use both Web.xml & Annotation
- when using both configurations for a servlet the setting provided by @WebServlet annotation takes precedence over corresponding settings provided in web.xml

14/1/24

Session Management in Java

- Session Management is crucial aspect of web dev. that allows server to maintain stateful information about user across multiple request
- In Java Servlet, session are typically used to store user specific data such as user preferences, shopping cart items or authentication details.
- A session is way to track user's interaction with website across multiple request
- It allows server to store user's specific information and associate with unique session Id.

Session Life Cycle (CTT)

Phase/steps ⇒
1. Creation: A session is created at the first time when user accesses the website/webapp
2. Tracking: The SessionID is used to associate subsequent request with same session.
3. Timeout: Session have timeout, if user is inactive for specified duration, the session expires.

* Session Tracking

- There are multiple ways by which we can track a session

1. Cookie (Default Mechanism):
   - The Server sends unique sessionID in a cookie stored on client's side.
2. URL Rewriting:
   - SessionID is appended to URL

Q. Why we need Sessions?

- Every request to a Server is treated as new request
- Hence, Server won't remember who the client is, in the next request (Tw) above 2 pts desc^{n} is of stateless behaviour)
- So again, client needs to introduce to server for every request to server (if Server is following stateless behaviour)
- To avoid this, we need to take help of session
- By which stateful behaviour can be achieved (Server will recog client throughout Multiple request)

Q what are Cookies?

→ - Cookies are small piece of data stored on user's or client's device by the web browser
- They are created when client visits a website and are designed to hold modest amount of data specific to a particular client and website.

[SP-TH ST]

* Purpose of cookies

1. Session Management: - Cookies help in managing user's sessions
- They store session information, allowing users to stay logged in as they navigate through different pages of a website.

2. Personalization: - Cookies are often used to remember user preferences and settings providing a personalized browsing experience
- This includes language preferences, theme choices etc.

3. Tracking and Analytics: - website owners use a cookie to collect data on user behaviour such as page views, time spend on site and most viewed content
- This information is valuable for analytics and improving website

4. Authentication: - Cookies plays a crucial role in user authentication
- Cookies stores information that allows a user to remain authorized so they don't have to login every time in a particular web page/ web app

5. Shopping Cart Management: - In Ecommerce websites cookies are used to store items in user's shopping cart ensuring that cart contents are persistant as user navigate throughout website.

6. Targeted Advertising :- Advertisers use Cookie to track user's interest and display targeted ads. , this depends on is based on user's browsing history and behaviour

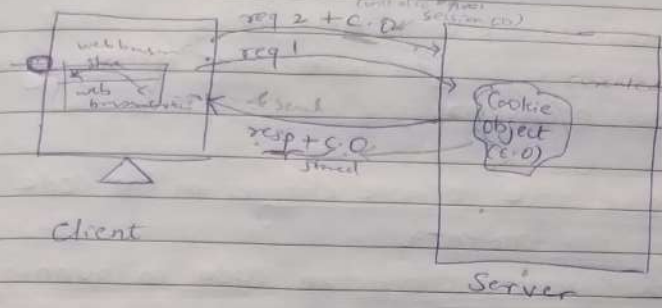
- A 'Cookie' in Java is a 'Class'
- Cookie class is present in 'javax.servlet.http.Cookie'
- Initially cookie is created in Server by Server
- Then that cookie will be stored on response and sent to the client.
- the client will take those cookies and stored it in browser

- When client makes another request, client will sent request along with cookie

#NOTE - In Cookie, we store data in (key, value) pair.

- We can create a Cookie object with help of Constructor
- In cookie class we have 2 constructors
  1. Cookie()
  2. Cookie(String key, String value)

- After creating cookie, we can attach it to response
- Then, browser will read response in which we have cookie and store it in browser (which is client side)
- From Next consecutive request onwards, browsen will send request along with cookie.

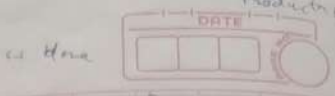


Session ID (Cookies)

1. When we access Page A for the first time, a new session is created and we get a session ID
2. Page B retrieves and prints the cookie along with session ID
3. When you access Page A again it is still part of same session. The server recognizes sessionID and cookie from client side indicating that it is the same session and therefore new session is not created
4. Page B in the second run will still print existing session ID and cookie details.

Q. Can we have Duplicate keys?

→ - In the context of Http Cookies, each cookie is uniquely identified by its name (key)
- If you try to add new cookie with same name as existing cookie it will effectively override existing cookie.
- This behaviour is designed in Http Cookie specification
- So while you can have multiple cookies with same name, the last one set (key value) will be the one which is sent which will be appended to the response
- The browser does not keep cookies with same name



- It replaces old one with new value.
- Having multiple cookies with same name might lead to unexpected behaviour.

Q. Is it good practice to make changes in the code in running environment?

→ - In general, making change in code in running environment is not considered as good practice.
- This is specially true in production environment where any changes can have immediate direct impacts to user.

- Below are some reasons why it is not recommended
1. Unpredicted Behaviour:
- Changing code in running environment can introduce unexpected issues or bugs.
- It is difficult to predict all possible consequences of changes made in without testing.

Q. When and how we shall do changes in code of web application.

→ - Code changes in web application needs to be done in structured and controlled process to minimize the rate of issues.)
- It involves following phases:
D * Development Environment
T * Testing
S * staging Environment
R * Review
A * Approval
D * Documentation
D * Deployment to Production
* Monitoring
* Rollback plan (If there is issue in Monitoring)

* Deployment to Production
- Scheduled a plan deployment windows during which we have less user activity
- It is also known as Maintainance window

• Setting Expiry to Cookie / Expiring a Cookie
- In Servlets you can set expiration of a cookie to control how long a cookie will persist on client's side.

Pgm: Expiring-Cookie

1. Creating a cookie with expiration
- When creating a Cookie we can set Maximum Age which is the time the cookie will remain at the client side before expiring.
- To set expiry to a cookie, we need to use method name setMaxAge() to set the expiration time
- The specified time in the argument of the above method is in seconds

```
// create a cookie object
Cookie cookie = new Cookie("pi", "3.142");
// Set Expiry time of cookie
cookie.setMaxAge(3600); // 1hr = 3600 sec
// Adding cookie to response
resp.addCookie(cookie);
```

2. Client side expiration
- keep in Mind that the expiration time is maintained on client side (how: by changing system time)
- Users can manipulate cookie, so it is better not to rely completely on cookie expiration for security

#NOTE: If you want a cookie to be valid for a particular session we can set MaxAge as '-1'

```
Cookie cookie = new Cookie("pi", "3.142");
cookie.setMaxAge(-1);
response.addCookie(cookie);
```

25/01/24

Hidden form fields

- Hidden form fields are HTML elements that are not visible to user but can store values which are submitted with form

* Purpose of Hidden form field

1. Data Transfer :- Hidden fields allows developers to transfer data between client and server without displaying it to user.

2. Session Management :- Hidden form fields can be used to store session related information

* Syntax of Hidden form field

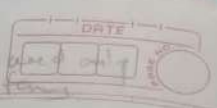
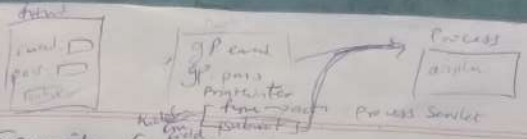
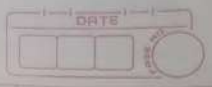
```
<input type="hidden" name="hiddenFormfield"
value="hiddenValue">
```

* Key attributes

- type: Specifies the type of input is set to hidden for this field
- name: It provides key for identifying the form field
- value: It holds the data for the particular name (key)

* Eg:

```
<form action="servlet">
    <input type="hidden" name="hiddenField"
    value="hiddenValue">
    <!-- Below are visible fields -->
    <input type="Submit" value="Submit">
</form>
```

* Security Consideration:

- Client Side Modification: - Hidden fields are part of HTML source code and it can be modified by user at client side. (by using Inspect in browser)

NOTE: - This is generally not recommended whenever we want to share/transfer sensitive data.

- (Same concept with Query Parameter)

o URL Rewriting

→ URL Rewriting is a technique used to include additional data in the URL

- Typically it is used for Session Management for passing parameters between pages

* Use Case of URL Rewriting in Session management

1. Session Tracking: - When cookies are disabled or not supported, URL rewriting is used to track sessions by appending session id to URL.

* How URL Rewriting works for Session Management:

→ - An unique session ID is appended to URL and as per requirement for session management logic Session ID is fetched from URL

* Why URL Rewriting for session Management is not recommended?

→ 1. Security Concern: ~~Exposed Session ID~~

(a) Exposed Session ID: - Session ID in URL are exposed in browser's address bar, raising the security concern which might cause session hijacking & security threats

Task: How to fetch session ID by using URL Rewriting
How many Http session objects.
(how to make obj of interface)

Q. Diff b/w ServletContext & ServletConfig (Imp)
(length for Browser etc & server URL)

2. Browser & Server Limitation:
a) Length Limitation - Some browsers & servers impose limit on URL length.
- Long URL's may be truncated (ignored) leading to session Data Loss (Session Data Loss)

30/1/24

HttpSession Management:

- HttpSession Management is a mechanism to maintain data across multiple request from the same client within web application

- There are a interface name 'HttpSession' present in 'javax.servlet' package

IQ: Q. Why do we need HttpSession when we have other ways to do Session Management? (Cookies, Hidden form field, URL Rewriting)

Cookies: Client side Manipulation is possible. Hidden form field: Security Concern: Data transferred b/w client & server is visible to user & user can manipulate that data (inspect browser). URL Rewriting: Security, length limitation, plain limitation of 3 ways. So to do session management using HTTPSession - we need to get HttpSession obj.

Q. How to Create HttpSession object

- Mainly there are 2 ways to get HttpSession object

Method 1 / Way 1: To get HttpSession object we have method name getSession() (implicit type working) (Here HttpSession will decide whether to make new object or not)

a) If session object already exist it will return existing object

b) If no session object is existing then it will return new session object

(When to use getSession() & getSession(boolean value))

Method 2 Way 2: getSession(boolean value) (explicit working) (pgm will decide whether to make new obj or not)

In this method if we pass true it is going to create new session object even if session object already exists

If we pass false it is going to create new session object only if session object does not exists otherwise, if session object exists, it will return existing object

l Servers

to Session

to maintain
e client.

ther ways
, URL Rewriting
manipulate that data
of HttpSession obj

n object
method

turn

will return

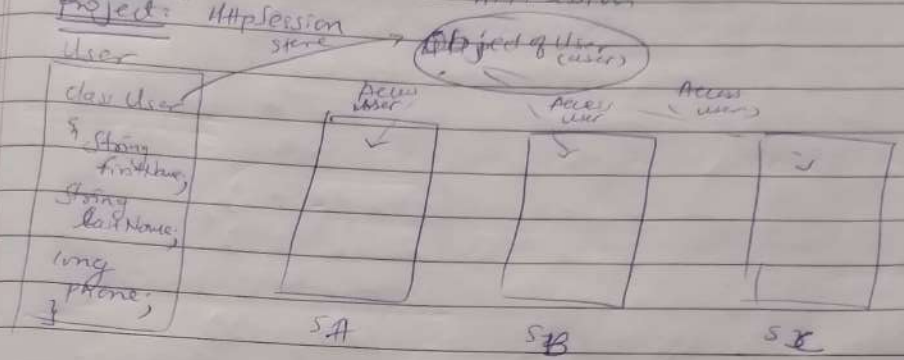
o create
ready exits
new
exists

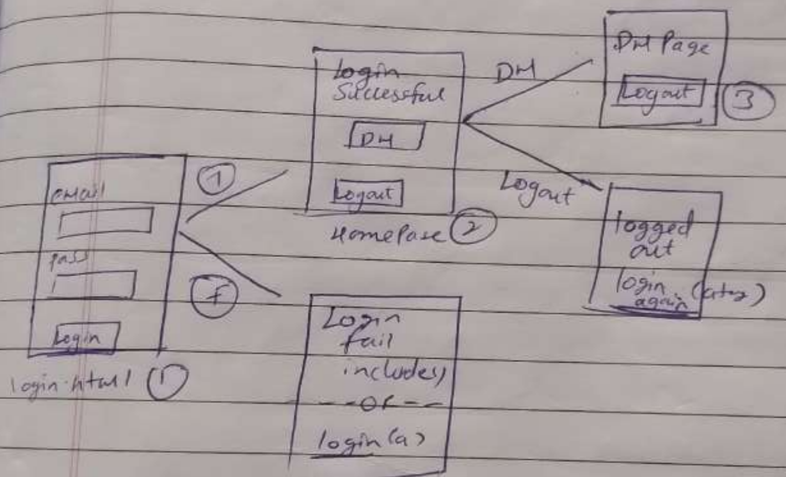
return
ect

basically: input tag: hidden
in read Servlet
<input name id value= ... />
(hidden fields)

#NOTE: - We can store any information (any type of data (String, class, int)) in HttpSession object
- Session object is visible for every Servlet class and JSP within web application.
Session: In one web application how many HttpSession object (ServletContext: one for webapp)
- (For one user for one web application there will be one HttpSession object.)

* Eclipse Project: HttpSession
User





URL modify: baseURL/target

Scenario ▷ (1) - (1) - (2) - (3) - logout