(12) Difference between Arrays and collection.

Arrays	collection
-> Arrays is fixed in size static.	-> collection is Growable in Nature (Dynamic).
-> Arrays is best in case of performance vice.	-> collection is Not that best in case of performance
-> Memory Management wise Arrays is worst.	-> Memory Management wise collection is best.
-> we can store only homo-genous type of data.	-> we can able to store both homogenous & hetro-genous type of data.
-> It can Able to hold both primitive and object type.	-> It can able to hold only object type.
-> underlying data struct-ure are not available.	-> collections are implem-nted on standard data structures

(13) Difference between collection and collections

collection	collections
-> It is an interface.	-> It is an utility class.
-> It belongs to java.util package.	-> It belongs to java.util package.
-> It represents group of objects known as its elements	-> It provides static utility methods for various collection
-> It has static & default method (from java 8)	-> It has only static method.

->What is Application.

It is a list of programs which helps us perform some a particular task/action.

Eg: WhatsApp
MS office
Web browser [chrome. Mozilla] Facebook.
Instagram.
GTA vice City.

=> Types of Application

1. StandAlone/Desktop Application.
2. Client server/Mobile Application.
3. Web Application
4. Enterprise Application.

=> StandAlone Application:-

Software installed in One Computer and used by only one person.

Eg:
Installing s/w of a Calculator, Adobe photoshop MS office, AutoCAD, paint ect.

Advantages:

-> Faster in access.
-> Secured from Data hacking and virus.

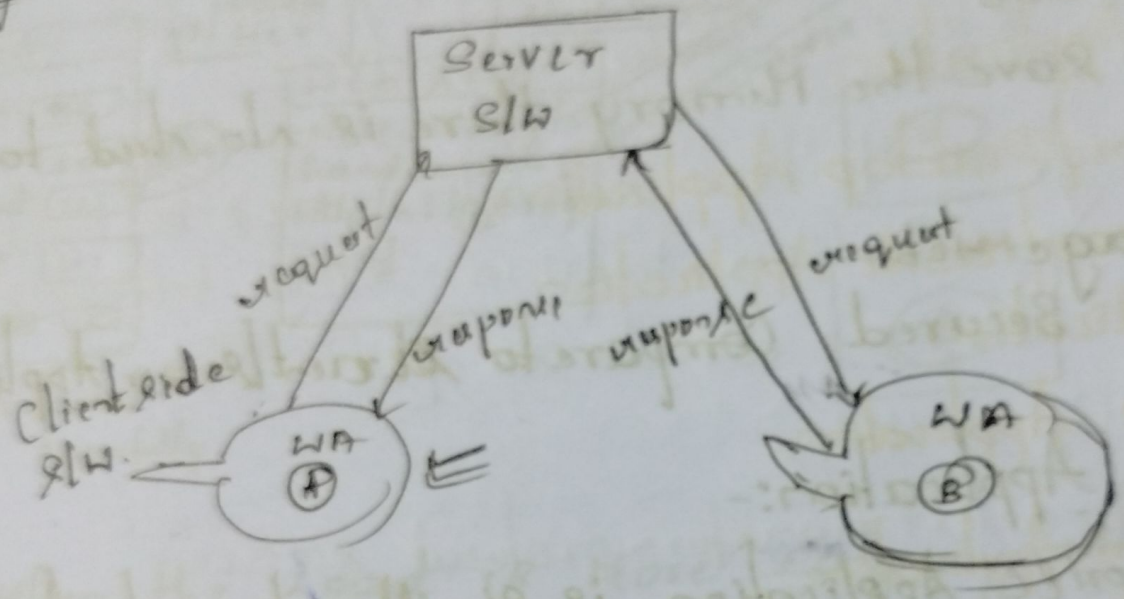
Disadvantage:-

+> Single user access at a time.

=> Client Server Application:-

do client Server Application, unlike standalone Application part of application installed on the client system and the remaining part installed on the server machine.

Eg: Facebook, WhatsApp, Instagram, Wiki, youTube...etc.



=> Web Application:-

Any Application which is opened through a browser is known as WEB APPLICATION.

Eg:- facebook, wiki, Gmail, SkillRary, amazon etc..

->

Advantages:-

-> Easy to access and faster in access if the bandwidth is more.
-> Data Security from the data hacking and virus.
-> Data Sharing is possible.
-> Maintenance is Not so tough.

-> Multiple users can access the Application.

=> Dis Advantages:-

-> Installation is still required at client's place.

-> If the server goes down no one can access the Application.

-> Advantage web

We can save the Memory there is no need to download.

-> Disadvantage web

Less Secured Compare to Client/Server Appli

=> Enterprise Application:-

Enterprise Application is similar to web Appli-cation but it is restricted to Organizations.

=> What is Server:-

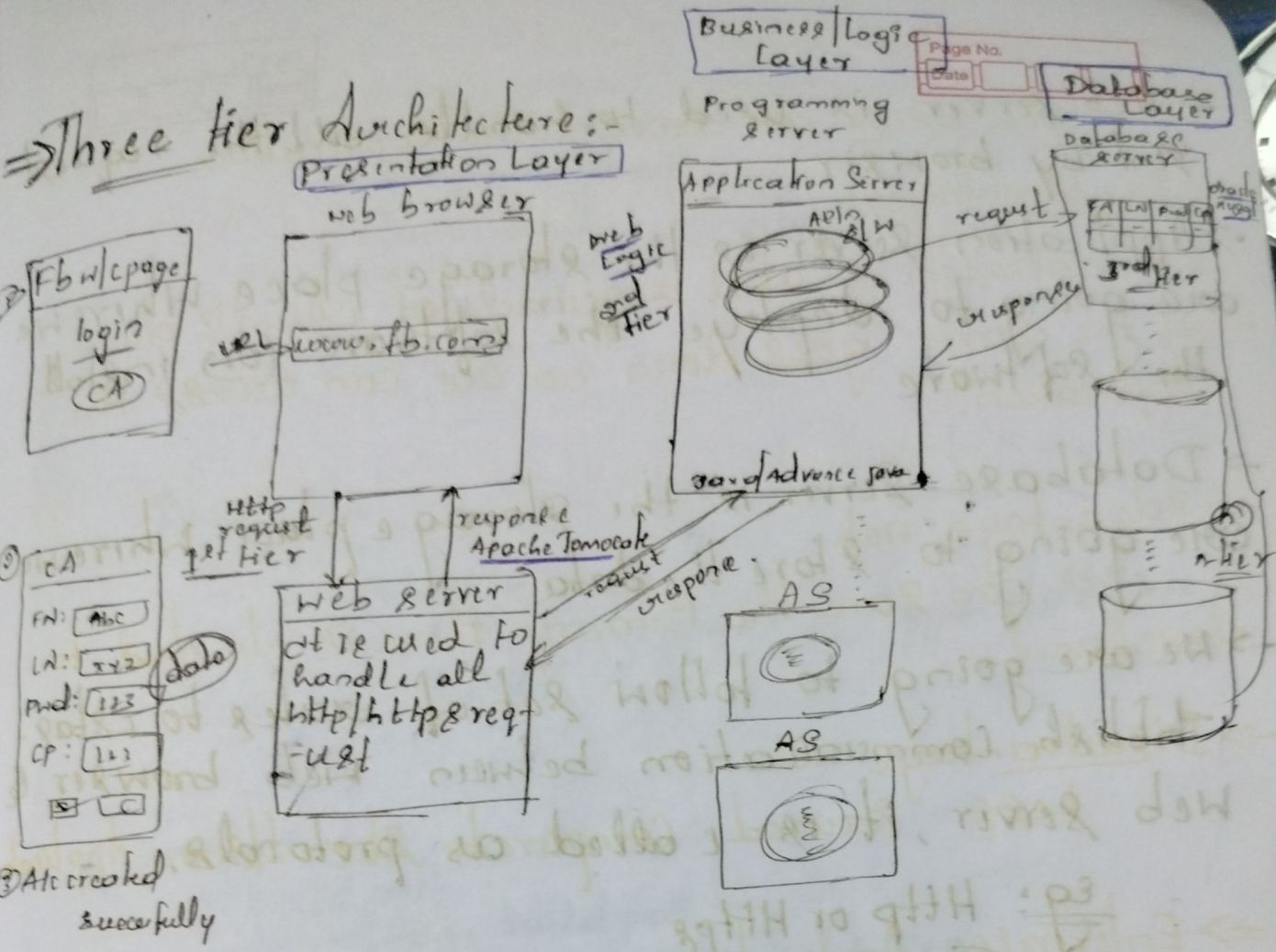
It is Nothing but a super Computer where in all the applications are installed and can be accessed by anyone. [High Configuration].

-> what is the difference between interface & Abstract class

-> How to call static Members within the class (or) between class.

-> How to call/access Non static Members within the class and between class.

⇒ Three tier Architecture :-



-> When the load is More company will go for the 'N' tier Architecture.

-> When the load is Less company will go for three tier Architecture.

-> When the communication happens between One Web Server, One Application Server, One Database server it is called as three tier Architecture.

-> When the communication happens between one web server, 'n' Number of Application Server & 'N' Number of Database server it is called as three tier Architecture.

-> Web server is used to handle all the requ
sent by browser

-> Application server is the storage place where
are going to deploye the software. (or) install
the software

-> Database server is the storage place where
are going to store the data.

-> We are going to follow set of rules to estab
-blish communication between web browser
web server, it is called as protocols.

eg: Http or Https
(Hyper text transfer protocal secure)
FTP (File transfer protocal)
SMPT (Simple mail transfer protocal)

=> How to create Servlet project:-

Step 1: go to file
↳ Create Maven project
↳ Don't check the checkbox click Next
↳ select All catalog
type org.apache.maven.
↳ select
↳ click Next
↳ Group_id
↳ Artifact_id

How to create a Servlet project:

-> File
  L> Maven project
    (uncheck simple project) -> next
      L> catalog (all catalogs)
        L> org.apache.maven
          L> maven-archetype-webapp
            L> groupid
            L> Artifactid

L> console :Y : Y [type]

① project
  L> Right click
    L> Maven
      L> update project
        L> check (Force update of snapshots/Release)
          L> ok

② -> project
  L> Right click
    L> Build path
      L> configure build path.
        L> check two of the checkboxes
          L> apply and close.

③ Project
  L> Right click
    L> properties
      L> Targeted Runtimes
        L> Apache Tomcat v9.0
          L> apply & close

=> Server
  L> Windows
    L> show view
      L> Others
        L> Server
          L> Add Server
            L> Browse (open folder)
              L> ok.

=> How to download apache tomcat server.

-> goto Any browser
  L> apache tomcat download
    L> go to apache tomcat website
      L> click on Tomcat9 under download
        L> click on zip link under core
          L> once zip file is downloaded
            L> go to downloads & Extract file

=> How to Add/configure apache tomcat server to eclipse

Step 1:- in eclipse goto window -> show view -> Servers (if server option is Not available go to other and search for servers). and double click on Server.

Step 2:- we will be getting No servers are available link, double click on that link
  L> select apache and version 9.0
    L> click on Next

Step 3:- click on Browse
  L> go to download &
    L> select apache tomcat extracted file.
      L> click on ok/finish.

NOTE:-

=> Once we create the Servlet project we should follow three steps.

Step 1:- Select the project (Right click)
  L> go to Maven
    L> update project
      L> check force update of Snapshots/Releases checkbox.
        L> click on ok.

step 1: Select the project (Right click)
↳ go to build path
↳ configure build path
↳ select all the checkboxes
↳ click on Apply & close

step 2: Select the project (Right click)
↳ go to properties
↳ select Targeted Runtimes
↳ select the Server
↳ click on Apply and close.

(This is nothing but deploying the project to Server)

-> If we want to run the Servlet project select the project Right click on the project -> go to Run as -> click on Run on Server.

-> If we are running Servlet project by default index.jsp file will get executed.

-> Inside web App folder we are going to create the frontend pages. (driver)

-> Inside Source/Main/java we are going to write logic (Java code) (conductor)

Page No.
Date

Applets:

->Applet was first used for developing java web app.
->When client/browser sents a requests, in server respective files will be downloaded after that respo is sent back to client (late response)

CGI (common gateway interface)

-> for every client request new process will be creat it consumes cpu memory & performance will become slower.
-> it uses c language & it is platform dependent.

Servlet:-

-> for every client request thread will be created Once response is sent back it will be destroyed.

Server side component, resides on server side an gives dynamic web pages.

=> What is Servlet:

-> Servlet is a discrete (individual/seperate) java class which resides into the container of web server which follows "component architecture" and "request-respons Model".

-> Servlet means "Server-side Component".

-> Servlet are the server side programs that run on a web server or application server and acts as a middle layer between web browser and database server.

-> Servlet technology is robust and scalable because of java language.

-> Servlet is a technology which is used to create a web Application.

-> Servlet is an API that provides many interfaces & classes.

-> Servlet is an interface that must be implemented for creating any servlet.

-> Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.

-> Servlet is a web component that is deployed on the server to create a dynamic web page.

=> Ways to creating a Servlet class:

1. Implementing Servlet Interface.
2. Extending GenericServlet.
3. Extending HTTPServlet.

(1) Servlet is an interface in javax.servlet package
It has 5 abstract methods.

init()
service()
destroy()
getServletInfo()
getServletConfig()

-> you can create a servlet by implementing the Servlet interface and providing the implementations for all these methods

(2) Generic Servlet is an abstract class in javax.servlet package.

-> Generic Servlet implements Servlet, ServletConfig and Serializable interfaces. It provides the implementation of all the methods of these interfaces except the service method.

-> GenericServlet class can handle any type of request so it is protocol-independent.

-> you can create a servlet by inheriting the GenericServlet class and providing the implementation of the service method.

Page No.
Date

③ HTTP Servlet is an abstract class in Javax.Servlet.Http package

-> The HttpServlet class extends the GenericServlet class and implements Serializable interface.

-> It provides http specific methods such as doGet, doPost, doHead, doTrace etc.

-> It is an abstract class with no abstract methods.

-> A subclass of HttpServlet must override at least one method, usually one of these:

① doGet:-
If the servlet supports HTTP GET requests, used in the case of fetching data.

② doPost:-
For HTTP POST requests. used in the case of inserting data @ Sending data.

③ doPut:
For HTTP PUT requests. Used in the case of doing update Operations.

④ doDelete:-
For HTTP DELETE requests. used when we want to perform delete Operations.

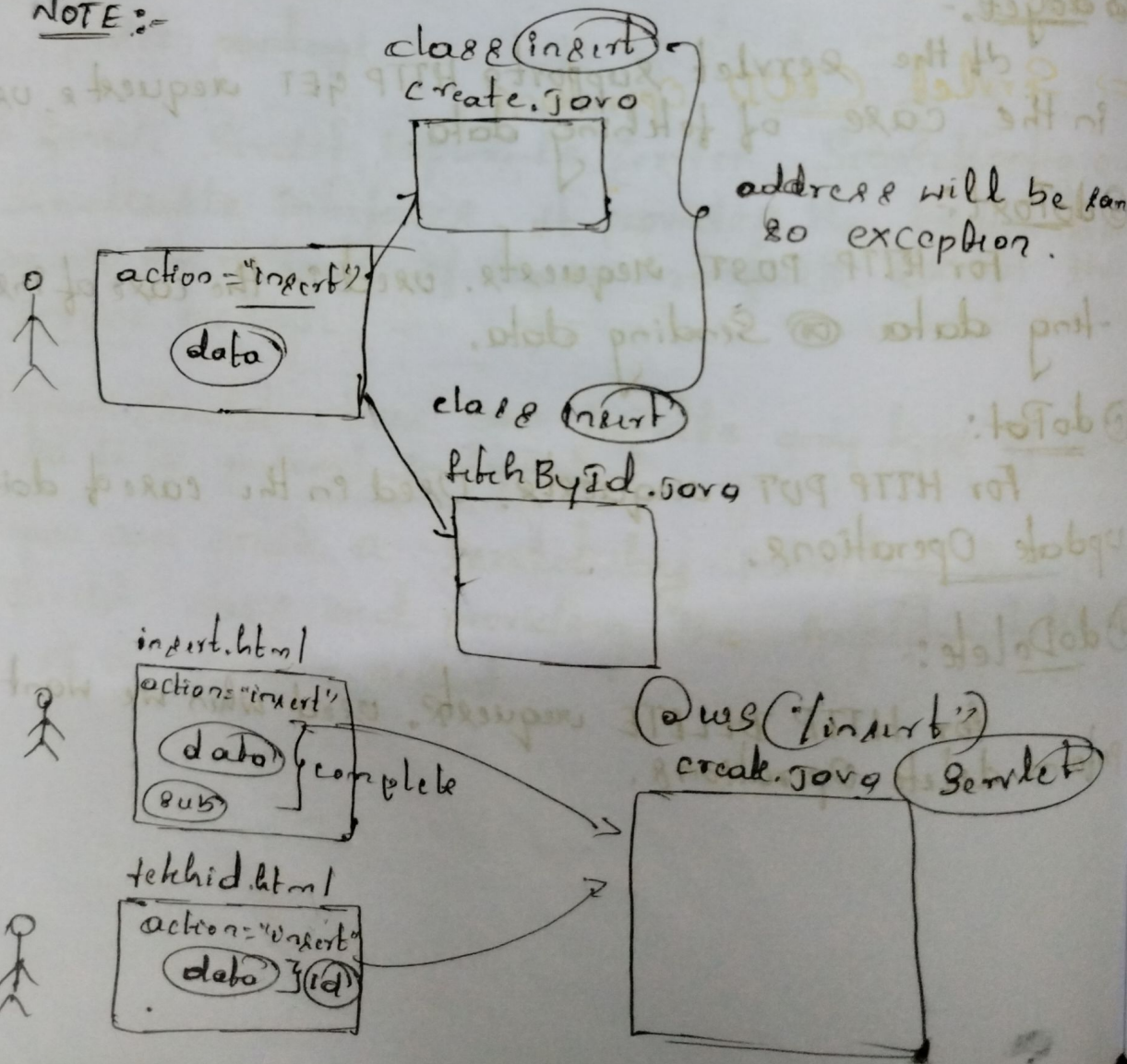
=> Web container / Servlet Container / Deployment Descriptor:

-> A web container is the component of a web server that interacts with the Java Servlet.

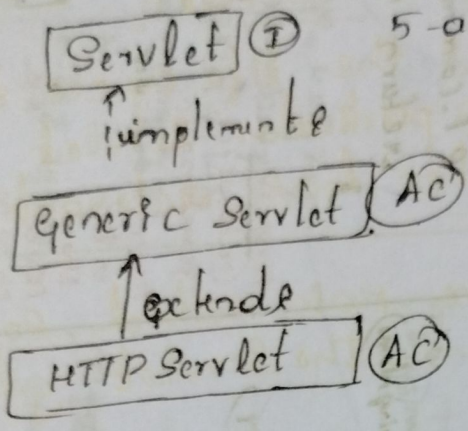
-> A web container manages the life cycle of servlets and it maps a URL to a particular servlet while ensuring that the requester has relevant access rights.

-> Java servlet do not have main method, so a container is required to load them. The servlet gets deployed on the web container.

=> NOTE :-



-> In how many ways we can create Servlet class?
(or)
Explain Heirarchy of Servlet?



=> Servlet CRUD Operations

Insert/Create
Front End — insert.html

```
<!DOCTYPE html>
<html>
<head><title>Insert</title>
</head>
<body>
<form action="insert">
<input type="number" name="id">
<input type="text" name="name">
<button>SUBMIT</button>
</form>
</body>
</html>
```

① req
data
resp

controller/ServletClass — Insert.java

```
@WebServlet("/insert")
class Insert implements
        Servlet
{
Public void service(req, resp)
{
String id=req.getParameter("id");
String name=req.getParameter("name");
int aid=Integer.parseInt(id);

StudentDto dto=new StudentDto();
  dto.setId(aid);
  dto.setName(name);

StudentDao dao=new StudentDao();
String msg=dao.insert(dto);
  resp.getWriter().println(msg);
}
}
```

StudentDto — StudentDto.class

```
@Entity
class StudentDto
{
@Id
Private int id;
Private String name;
public void setId(int id)
{
this.id=id;
}
public int getId()
{
  return id;
}
public void setName(String name)
{
  this.name=name;
}
public String getName()
{
  return name;
}
@Override
public String toString()
{
return "StudentDto [id="+id+
  ", name="+name+"]";
}
}
```

② req

StudentDao — StudentDao.class

```
class StudentDao
{
EntityManagerFactory entityManagerFactory=Persistence.cre-
    -ateEntityManagerFactory("DM");
EntityManager EM=entityManagerFactory.createEntityMa-
    -nager();
EntityTransaction ET=EM.getTransaction();

Public String insert(StudentDto d1)
{
  ET.begin();
  EM.persist(d1);
  ET.commit();
  return "data inserted";
}
```

object → RDBMS (ORM Tool)

③ req

persistence.xml
(database Configuration file)
persistence unit: dev
url
Uname
Pwd
db name

database

StudentDto

Id	Name
1	abc

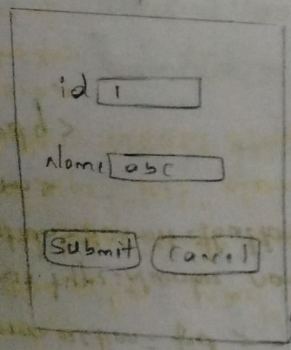
Page No.
Date

Insert/Create

Frontend

webApp insert.html

```
<body>
<form action="create">
id: <input type="number" name="id">
name: <input type="text" name="name">
<button> Sumbit </button>
<button> Cancel </button>
</form>
</body>
```



1 Request data

response

controller/Servlet class

create.java

```
@WebServlet("/create")
public class create implements Servlet(I)
{
Public void service(ServletRequest(I) req,
  ServletResponse(I) res)
 {
 String id=req.getParameter("id");
 String name=req.getParameter("name");
 int cid=Integer.parseInt(id);
 StudentDto dto=new StudentDto();
   dto.setId(cid);
   dto.setName(name);
 StudentDao dao=new StudentDao();
 String msg=dao.insert(dto);
   res.getWriter().print(msg);
 }
```

2

DTO/Entity class

StudentDto.java

```
@Entity
class StudentDto
{
 @Id
  Private int id;
  Private String name;
Public void setId(int id)
 {
  this.id=id;
 }
Public int getId()
 {
  return id;
 }
Public void setName(String
                    Name)
 {
  this.name=name;
 }
Public String getName()
 {
  return name;
 }
@Override
Public String toString()
{
return "StudentDto [id="+id+
       name="+name+"]";
```

Response

DAO class

daoStudentDao

```
EMF emf=Persistence.CreateEMF("dev");
EM em=emf.CreateEntityManager();
ET et=em.getTransaction()
Public String insert(StudentDto d1) //ob
 {
  et.begine();
  em.persist(d1); // object
  et.commit();
  return "data inserted";
 }
```

3

persistence.xml

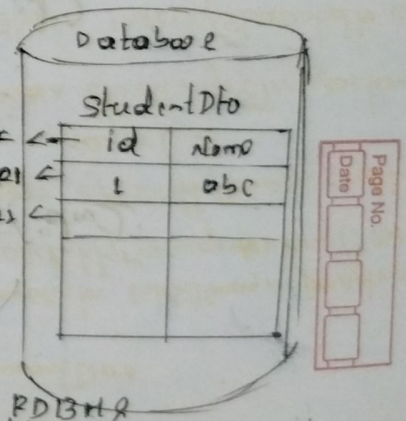
DatabaseConfigurationfile

Persistence unit="dev"

URL

UN dbname.

pwd



ORM tool [hibirnate]

FetchById

frontend

webApp fetchbyid.html

```
<body>
<form action="fid">
<input type:"Number" name="pk">
<button>Submit</button>
</form>
</body>
```

Controller/Servlet class

Fetchbyid.java

```
@WebServlet("/fid")
Public class Fetchbyid implements
    Servlet
{@Override
Public void Service(ServletRequest
 req, ServletResponse res)
{
String id=req.getParameter("pk");
Int cid=Integer.parseInt(id);
StudentDto dto=new StudentDto();
StudentDao dao=new StudentDao();
StudentDto sd=dao.fetchById(cid);
   res.getWriter().print();
 }
}
```

Dao class

StudentDao.java

```
class StudentDao
{
EMF emf=Persistence.createEMF("dw");
EM  em=emf.CreateEM( );
ET  et=em.getTransaction();

public StudentDto fetchById(int id)
 {
 StudentDto s1=em.find(StudentDto.class, id);
    return s1;
 }
}
```

FetchAll

controller/Servlet class

FetchAll.class

```
@WebServlet("/fall")
public class FetchAll extends HttpServlet
{
  public void doGet(HttpServletRequest req,
                    HttpServletResponse resp)
  {

Query q=
   StudentDto dto=new StudentDto();
   StudentDao dao=new StudentDao();
List<StudentDto> list=dao.fetchAll();
   resp.getWriter().print(list);
  }
}
```

Dao class

StudentDao.java

```
class StudentDao
{
EntityManagerFactory emf=Persistence.CreateEntityManagerFactory("dev");
EntityManager em=emf.createEntityManager();
EntityTransaction et=em.getTransaction();
public List<StudentDto> fetchAll()
 {
Query q=em.createQuery("Select a from StudentDto a");
 List<StudentDto> list=q.getResultList();
  return list;
 }
}
```

DeleteById

frontend

deletebyid.html

```
<body>
<form action="did">
<input type="number" name="pk">
<button>SUBMIT</button>
</form>
</body>
```

controller/Servlet class

```
@WebServlet("/did")
class Deletebyid extends HttpServlet
{
public void doGet(HttpServletRequest req,
    HttpServletResponse resp)
{
String sid = req.getParameter("pk");
int cid = Integer.parseInt(sid);
StudentDao dao=new StudentDao();
String msg=dao.deleteById(cid);
resp.getWriter().println();
}
}
```

Dao class

StudentDao.java

```
class StudentDao ext
{
EMF emf=Persistence.CreateEMF("dev");
EM em=emf.CreateEM();
ET et=em.getTransaction();
Public String deleteById(int pk)
{
StudentDto s=em.find(StudentDto.class, pk);
if(s!=null)
{
et.begin();
em.remove(s);
et.commit();
return "data deleted";
}
else
return "no data found";
}
}
```

DeleteAll

controller/Servlet class

```
@HebServlet ("/dall")
class DeleteAll extends HttpServlet
{
public void doGet (HttpServletRequest req,
                   HttpServletResponse resp)
{

StudentDao dao=new StudentDao();
   String msg=dao.deleteAll();
   resp.getWriter().print(msg);
}
}
```

Dao class

StudentDao.java

```
class StudentDao
{
  EMF  emf=Persistence.CreateEntityManagerFactory("dev");
  EM   em=emf.CreateEntityManager();
  ET   et=em.getTransaction()

  Public String deleteALL()
  {

Query q=em.createQuery("select a from StudentDto a");
List<StudentDto> list=q.getResultList();
 if (list.isEmpty())
   {
     for(Object o:list)
        {
          et.begin();
          em.remove(o);
          et.commit();
        }
     return "data deleted";
   }
  else
   return "no data found";
  }
}
```

=> Difference between RequestDispatcher & SendRedirect.

-> RequestDispatcher is used to pass the request to another resource for further processing "within the same server", another resource could be any servlet, JSP page or any kind of file where in SRD is used to redirect client request to some other location for further processing, the new location is available on "different Server or different conte

-> In RD the process of transferring the request taken care by web container without the client being informed.
Where in SRD our web container handles this request is visible in browser as a new Request.

-> RD is called as Server-side Redirect.
Where in SRD is called as Client-side redirect

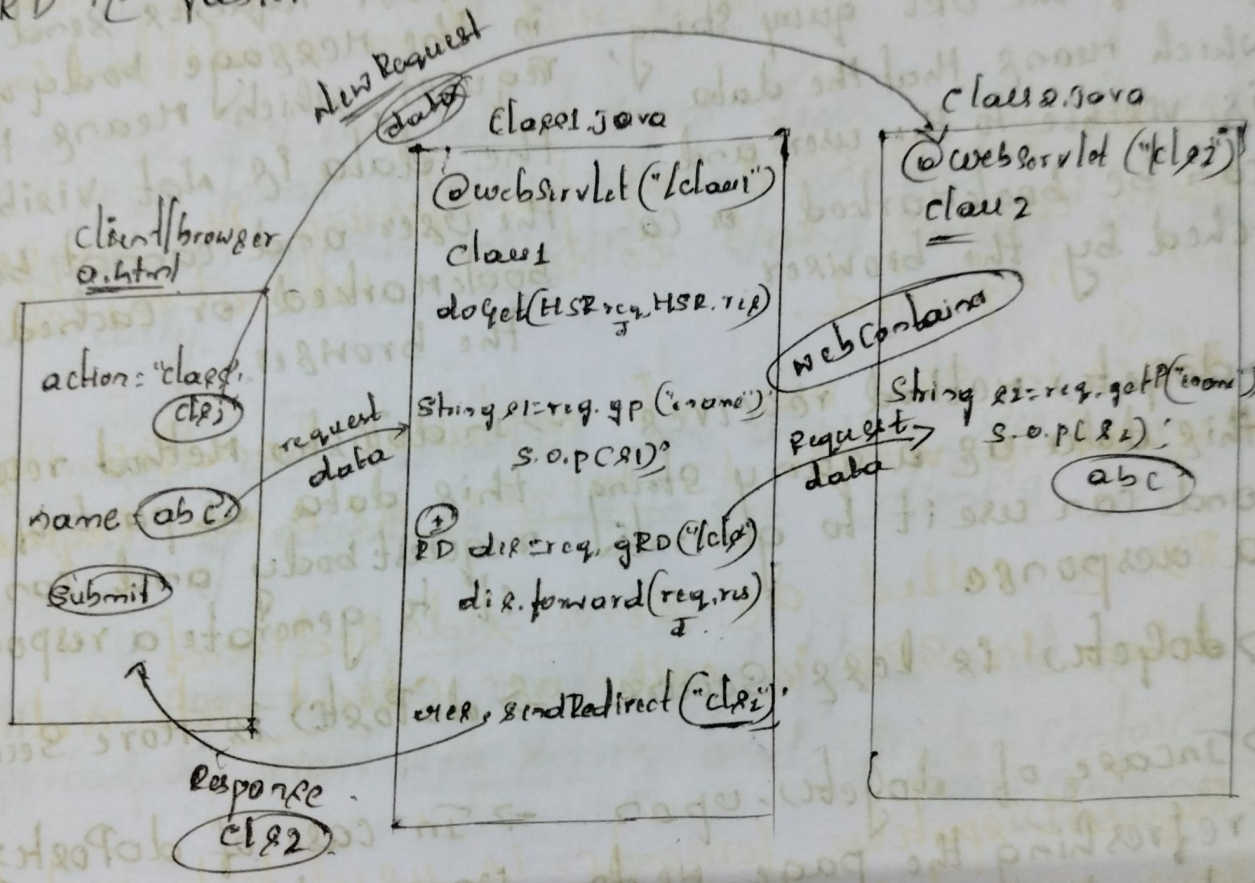
-> RequestDispatcher forward() is used to forward the same request to another resource where in,

In SendRedirect, web application returns the response to client with status code 302 (redirect) with URL to send the request. The request sent is a completely new request.

-> RequestDispatcher forward() is handled internally by the web container where in sendRedirect() is handled by browser.

-> In forward() browser is unaware of the actual processing resource and the URL in address bar remains same.
Where in sendRedirect() URL in address bar can change to the forwarded resource.

-> RD is faster than SRD.

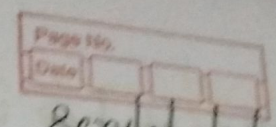


-> Http is an Stateless protocal Because Each and Every request acts like a new Request

=> Difference between doGet() & doPost()

doGet()	doPost()
-> doGet() is used to handle HTTP GET requests, which are typically used to retrive data from a server.	-> In doPost() is used to handle HTTP POST requests which are typically used to send data to server.
-> GET Requests send data as part of the URL query string, which means that the data is visible to the user and can be bookmarked or cached by the browser.	-> In POST requests send data in the message body of the request, which means that the data is not visible to the user and cannot be bookmarked or cached by the browser.
-> doGet() method receives this data as a query string and can use it to generate a response	-> In doPost() method receives this data as part of the request body and can use it to generate a response
-> doGet() is less secure.	-> doPost() is more secure.
-> Incase of doGet(), upon refreshing the page we do not get any security popup alert	-> In case of doPost(), upon refreshing the page we get security popup alert to avoid duplicate transaction.

Java Server Pages (JSP)



-> JSP is an improved extended version of Servlet technology.

-> Java Server Pages (JSP) is a technology that allows developers to create dynamic web pages using a combination of HTML, XML, and Java code (or) it is a technology which is used for developing dynamic web pages which helps developers insert Java code in HTML Pages by making use of special JSP tags.

-> Using JSP, you can collect input from users through web page forms & also existing records from a database (or) another resource, and create web pages dynamically.

-> JSP pages are executed on a web server and the resulting output is sent to the client's web browser.

-> Jsp is part of the Java EE platform and is supported by most web servers and servlet containers. (web container)

=> Advantages of JSP:-

-> It is more convenient to write (and to modify) regular HTML than to have plenty of println statements that generate the HTML. (against servlet).

-> Jsp allows developers to follow the Model-View-Controller (MVC) design pattern which separates web applications' Presentation, logic, and data.

-> This makes it easier to create scalable and Maintainable web applications. (same as servlet)

-> It provides good security features like session tracking, user authentication, and access restriction (same as servlet).

=> Difference between JSP and Servlet:-

-> JSP Pages allow web designers to work with HTML (or) XHTML Markup,

JSP	Servlet
-> Jsp pages allow web designers to work with HTML (or) XHTML Markup	-> Servlets requires a deep understanding of Java.
-> Jsp are text files that contain HTML, Java code, and JSP tags and are compiled into servlets by the Web container.	-> Servlets are Java classes that run on the server and handle requests and responses.
-> JSP can Mix HTML and Java Code.	-> Servlets are written in pure Java.
-> JSP is slower because JSP Need to be translated and compiled into servlets before execution	-> Servlets are faster than JSP.

-> JSP can only handle HTTP requests.	-> Servlet can handle any type of requests, such as HTTP, FTP (file transfer protocol) or, SMTP (Simple Mail transfer protocol.
-> JSP are more suitable for presentation and user interface.	-> Servlets are more suitable for complex business logic and data processing.
-> JSP can be used as views.	-> Servlet can be used as controllers in the MVC pattern.

=> Life Cycle of Servlet:-

-> The life cycle of a servlet is the process that describes how the servlet container manages the servlet from its creation to its destruction.

The life cycle of a servlet consists of the following phases:

-> Loading:

The servlet container loads the servlet class into the memory when the first request for the servlet is received.

The servlet class is loaded only once in the life cycle.

-> Instantiation:

The servlet container creates an instance of the servlet class by invoking its no-argument constructor.

-> The Servlet instance is created only once in the life cycle

=> Initialization:

The Servlet container calls the init() Method of the servlet instance to perform any initialization tasks, such as opening database connections or reading configuration files.

The init() Method is called only once in the life cycle.

-> Service:-

The Servlet container calls the service() Method of the servlet instance for each request that is sent to the Servlet. The s

-> The service() Method handles the request and generates the response by executing the java code and the JSP tags in the Servlet.

-> The service() Method is called repeatedly in the life cycle.

-> Destruction:

The Servlet container calls the destroy() Method of the Servlet instance when the Servlet is no longer needed or the web application is shut down.

-> The destroy() method performs any cleanup tasks such as closing database connections or releasing resources.

-> The destroy() method is called only once in the life cycle.

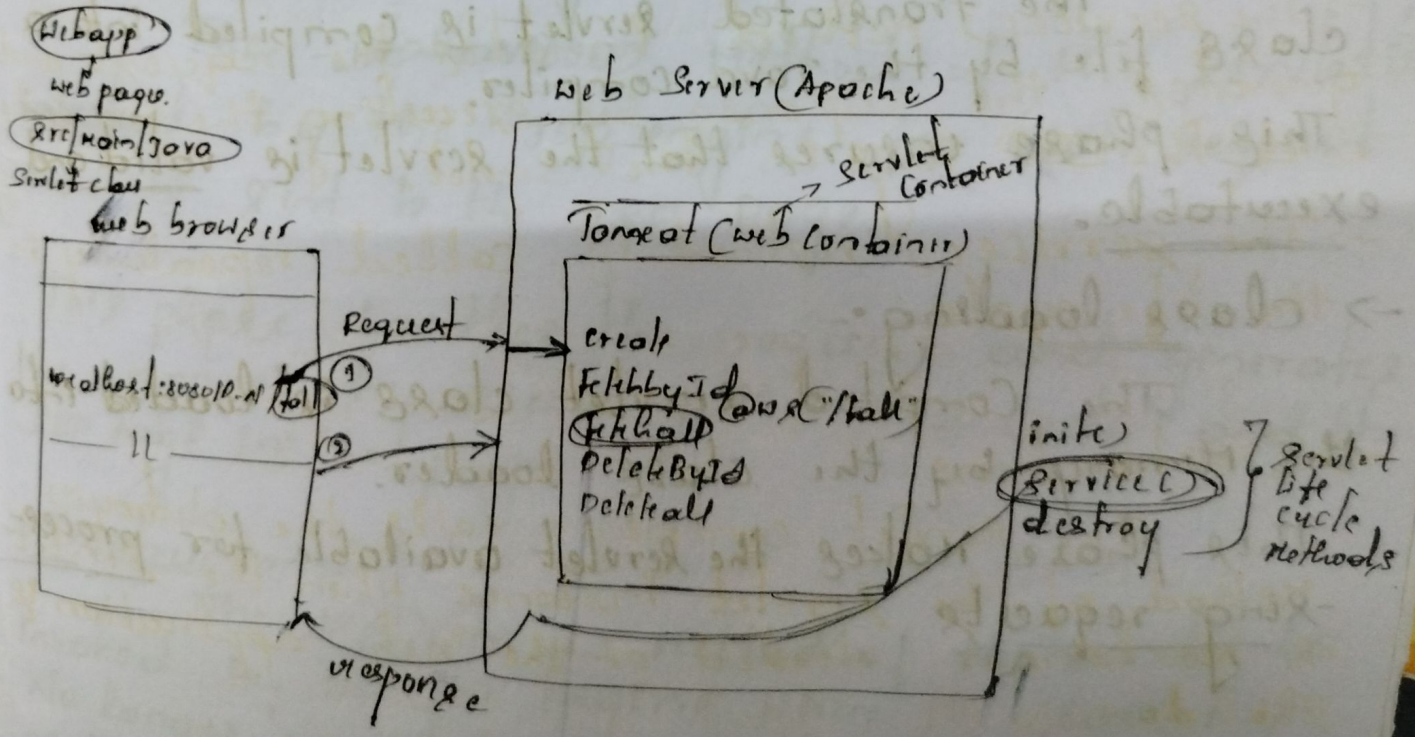
=> What is the difference between static web pages and dynamic web pages.

Servlet life cycle:-

=> 5 phases/Stages

1. Loading
2. Instantiation
3. Initialization -> init()
4. Request handling -> service()
5. destruction -> destroy

} Servlet life cycle methods Executes 'N' Number of times based on Requests.



⇒ Life cycle of JSP:-

→ The life cycle of a JSP page is the process that starts with its creation and ends with its destruction. The life cycle of JSP page consists of the following phases.

→ Translation:- (.JSP to .Java (JSPEngine))

The JSP page is translated into a Java servlet by the JSP engine.

This page phase checks the syntax and the structure of the JSP page and generates the corresponding Java code.

→ Compilation:-

The translated servlet is compiled into a class file by the Java compiler.

This phase ensures that the servlet is valid and executable.

→ Class loading:-

The Compiled servlet class is loaded into the Memory by the class loader.

This phase makes the servlet available for processing requests.

Page No.
Date

→ Instantiation:-
An instance of the servlet class is created by the JSP engine.
-> This phase initializes the servlet object and its fields.

-> Initialization:-
The jspInit() method of the servlet is invoked by the JSP engine. This phase performs any initialization tasks that are required for the servlet, such as opening database connections or reading configuration files.

-> Request processing:-
The _jspService() method of the servlet is invoked by the JSP engine for each request that is sent to the JSP page.

This phase handles the request and generates the response by executing the java code and the JSP tags in the page.

-> Destruction:-
The jspDestroy() method of the servlet is invoked by the JSP engine when the servlet is no longer needed or the web application is shut down.

-> This phase performs any cleanup tasks that are required for the servlet, such as closing database connections or releasing resources.

=> JSP tags:

1. Scriptlet tag:
   used to write java code
   <% %>

2. Expression tag
   used to print the values.
   <%= %>

3. Declaration tag.
   used to declare the members.
   <%! %>

4. Directives tag.
   to import the packages and define error handling pages or the session information of the JSP page.
   <%@ %>

5. comment tag.

JSP comment marks to text or statements that the JSP container should ignore.

A JSP comment is useful when you want to hide or "comment out", a part of your JSP page.

<%-- JSP comment --%>

=> without using @WebServlet() :- (web.xml)

```
<web-app>
<display-name>Archetype Created web Application</display-name>

<servlet>
<servlet-name></servlet-name>
<servlet-class></servlet-class>
</servlet>

<servlet-mapping>
<servlet-name></servlet-name>
<url-pattern></url-pattern>
</servlet-mapping>

</web-app>
```

Eg:

```
<web-app>
<display-name>Archetype Created web Application</display-name>

<servlet>
<servlet-name>a</servlet-name>
<servlet-class>session_tracking.httpsession.Login
                </servlet-class>
</servlet>
```

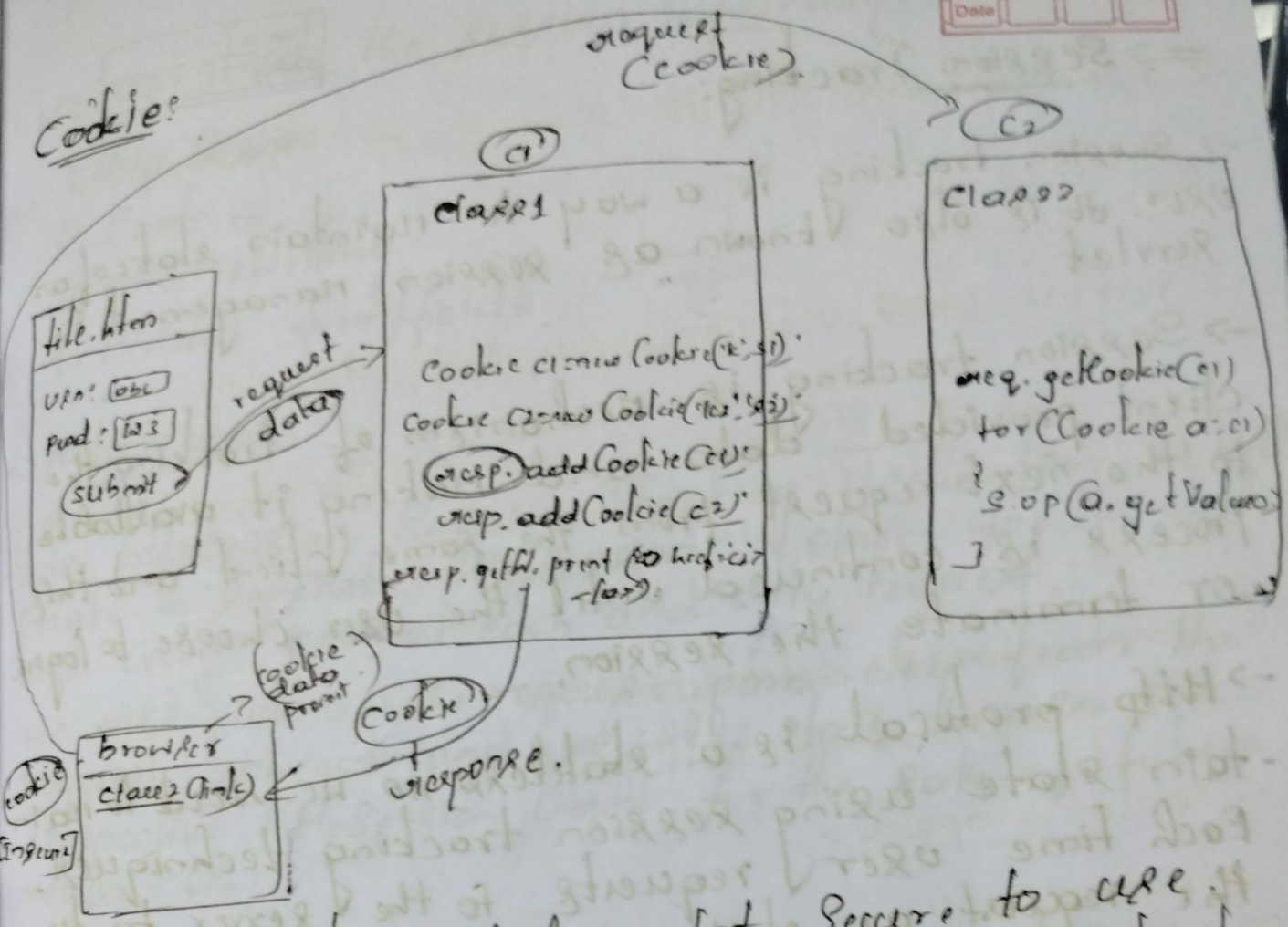
↓ Packagename.classname

```
<servlet-mapping>
<servlet-name>a</servlet-name>
<url-pattern>/login</url-pattern>
</servlet-mapping>
</web-app>
```

=> 4 ways of Session Tracking:

1. Http_Session
2. Cookie
3. Url Rewriting
4. hiddenform.

Cookies



-> Cookie is a class which is Not Secure to use. because it holds the data in the browser. [client side]

->In http.session. data will not be hold in the browser handled by the server. [Server side]

-> cookies stored on the user completter	-> Sessions are stored on the server.
-> Cookie can stored a limited amount of data, a maximum of 4kB.	-> Session can be stored an unlimited amount of data.
-> Cookie does not depend on the session.	-> Session depends on the cookie.
-> Cookies expires according to the time of expiry set for.	-> It expires after the user closes the web browser.
-> It has many security issues as it can be accessed by anyone easily.	-> It is secure as it cannot be accessed by anyone easily.

=> Session Tracking:-

-> Session tracking is a way to maintain state of an user. It is also known as session management in servlet.

-> Session tracking is mechanism of tracking the client provided data and making it available to the next request from the same client. and this process is continued until the user choose to logout or terminate the session.

-> Http protocal is a stateless so we need to maintain state using session tracking techniques. Each time user requests to the server treats the request as new request, so we need to maintain the state of an user to recognize the particular user.

-> Session tracking is a mechanism that servlets use to maintain state about a series of requests from the same user (that is request originating from the same browser) across some period of time.

NOTE:-

Http is stateless that means each request is considered as the new request.

We can track the session in four ways:

1. Http Session.
2. Cookies
3. URL rewriting
4. Hidden form fields.

HttpSession:

→ In HttpSession Session Tracking Mechanism we will create a seperate HttpSession object for the each and every user at each and every request.
We will pick up the request parameters from the request object and we will store them in the respective HttpSession Object for the sake of future reusability.

→ To get HttpSession Object if we use getSession() Method, then Container will check whether any HttpSession object existed for the respective user or Not,
if any HttpSession Object is existed then Container will return the existed HttpSession object reference.

→ If no object is existed for the respective user then Container will created a new HttpSession Object and return its reference.

=> HttpSession Methods:

(1) public void invalidate():
To destroy the HttpSession object we will use this method.

(2) public void setMaxInactiveInterval(int time):
If we want to destroy the HttpSession object after a particularly ideal time duration then we have to use this method.

(3) public void setAttribute(String name, Object value):
To set an attribute onto the HttpSession object we have to use this Method.

(4) public Object getAttribute(String name):
To get a particular attribute value from the HttpSession object we have to use this Method

(5) public void removeAttribute(String name):
To remove an attribute from the HttpSession object we have to use the following Method.

⇒ Advantages of HttpSession Session Tracking

-> There is no restriction on the size of the object, any kind of object can be stored in a session.

-> The usage of the session is not dependent on the client's browser.

-> It is secure and transparent.

⇒ Disadvantages of HttpSession Session Tracking:

-> We will create a HttpSession object for each and every user where if we increase the number of users then automatically the number of HttpSession objects will be created at the server machine, it will reduce the Overall performance of the web application.

-> We are able to identify user-specific HttpSession Objects ~~will be~~ among multiple number of HttpSession Objects by carrying session-id value from client to server and from server to client.

Cookies :-

-> Cookies are the mostly used technology for session tracking. Cookie is key value pair of information, sent by the server to the browser. This should be saved by the browser into space in the client computer

-> Session tracking is easy to implement and maintain using the cookies.

-> Disadvantage is user can disable cookies in their browser in such case the browser will not save the cookie at client computer and session tracking fails.

-> Cookies are the small files which are stored on the user's computer. They are designed to hold a modest amount of data specific to a particular client and website.

-> Javax.servlet.http.Cookie class provides the functionality of using cookies.
It provides a lot of useful methods for cookies like setName, setMaxAge, setPath, setValue etc.

-> Creates a cookie, a small amount of information sent by a servlet to web browser, saved by the browser, and later sent back to the server. A Cookies value can uniquely identify a client, so Cookies are commonly used for session management.

-> A cookie has a name, a single value and optional attributes such as comment, path and domain qualifiers, maximum age and a version number.

-> Servlets sends cookies to the browser by using the HttpServletResponse #resp.addCookie() method, which adds fields to HttpResponse headers to send cookies to the browser, one at a time.

-> The browser returns cookies to the servlet by adding fields to HTTP request headers. Cookies can be retrieved from a request by using the HttpServletRequest. #req.getCookie() method. several cookies might have the same name but different path attributes.

* => Write the difference between HttpSession and Cookies, As a developer which one will prefer for session tracking.

* -> Explain types of Cookies.

⇒ Url rewriting:

-> When a request is made, additional parameter is appended with the url. In general, added additional parameter will be session_id or sometimes the user_id. It will sufficient to track the session.

-> This type of session tracking doesn't need any special support from the browser.

-> Disadvantage is, implementing this type of session tracking is tedious. We need to keep track of the parameter as a chain link until the conversation completes and also should make sure that, the parameter doesn't clash with other application parameters.

⇒ Hidden Form Field:

```
<INPUT TYPE="hidden" NAME="technology" VALUE="Servl
```

-> Hidden fields are like the above can be inserted in the webpages and information can be sent to the server for session tracking.

-> These fields are not visible directly to the user but can be viewed using view source option from the browsers. This type doesn't need any special configuration from the browser of server and by default available to use for session tracking.

-> This cannot be used for session tracking when the conversation included static resources like html pages.

=> Difference between Cookies and HttpSession.

Cookie	HttpSession
-> It is a class.	-> It is an interface.
-> It is stored in the client browser/User Computers.	-> Session are stored in the server side.
-> cookies stored data in textfile.	-> Session stored a ~~unlimited~~ data ~~in~~ encrypted form.
-> cookies stored data upto 4KB.	-> Session we can store the unlimited data.
-> Cookies are not secured.	-> Session are more secured.
-> In cookie keep information until the the deleted by user Can set as per timer)	-> It is available until the browser is opened.

=> Types of Cookie:-

① Non-Persistent cookie:-
It is valid for single session only.
It is removed each time when user closes the browser.

② Persistent Cookie:-
It is valid multiple session.
It is not removed each time when user closes the browser. It is removed only if user logout or signout.