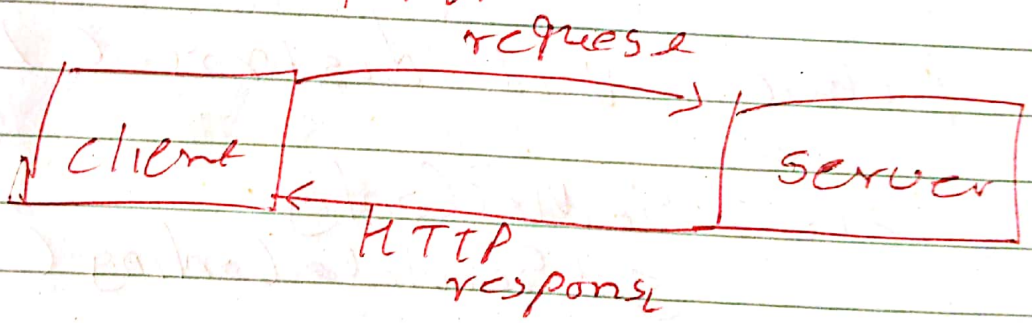
Advance Java Programming

- Servlet is use for web application development

Web Technology

1. Website
   Static vs dynamic
2. HTTP
3. HTTP Request & HTTP response
4. Get vs POST
5.



- The Servlet container is a part of webserver which can be run in seperate Process

* Server type
  1. web Server
     Apache Tomcat, Resin
  2. Application server
     JBoss, Glassfish, Weblogic

(1) Servlet Interface

→ Servlet Interface provide common behaviour of all the Servlet.

Servlet Interface Method

1. public void init(Config con)
2. public void service(ServletRequest req, ServletResponse res)
3. public void destroy()
4. public ServletConfig getServletConfig()
5. public String getServletInfo()

```
import java.io.*;
import javax.servlet.*;

public class IT implement Servlet
{
    all above method
}
```

2) GenericServlet class

GenericServlet class implements Servlet, ServletConfig & Serializable

List Method

Public void Init()

{

}

Public class It extend GenericServlet

{

}

② HttpServlet class extends
HttpServlet GenericServlet class
the and implements Serializable

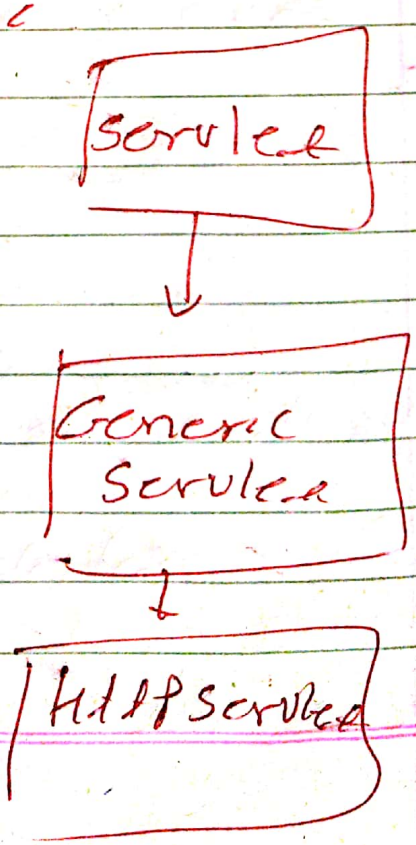
Important Method

① public void doGet
(HttpServletRequest req
HttpServletResponse res)

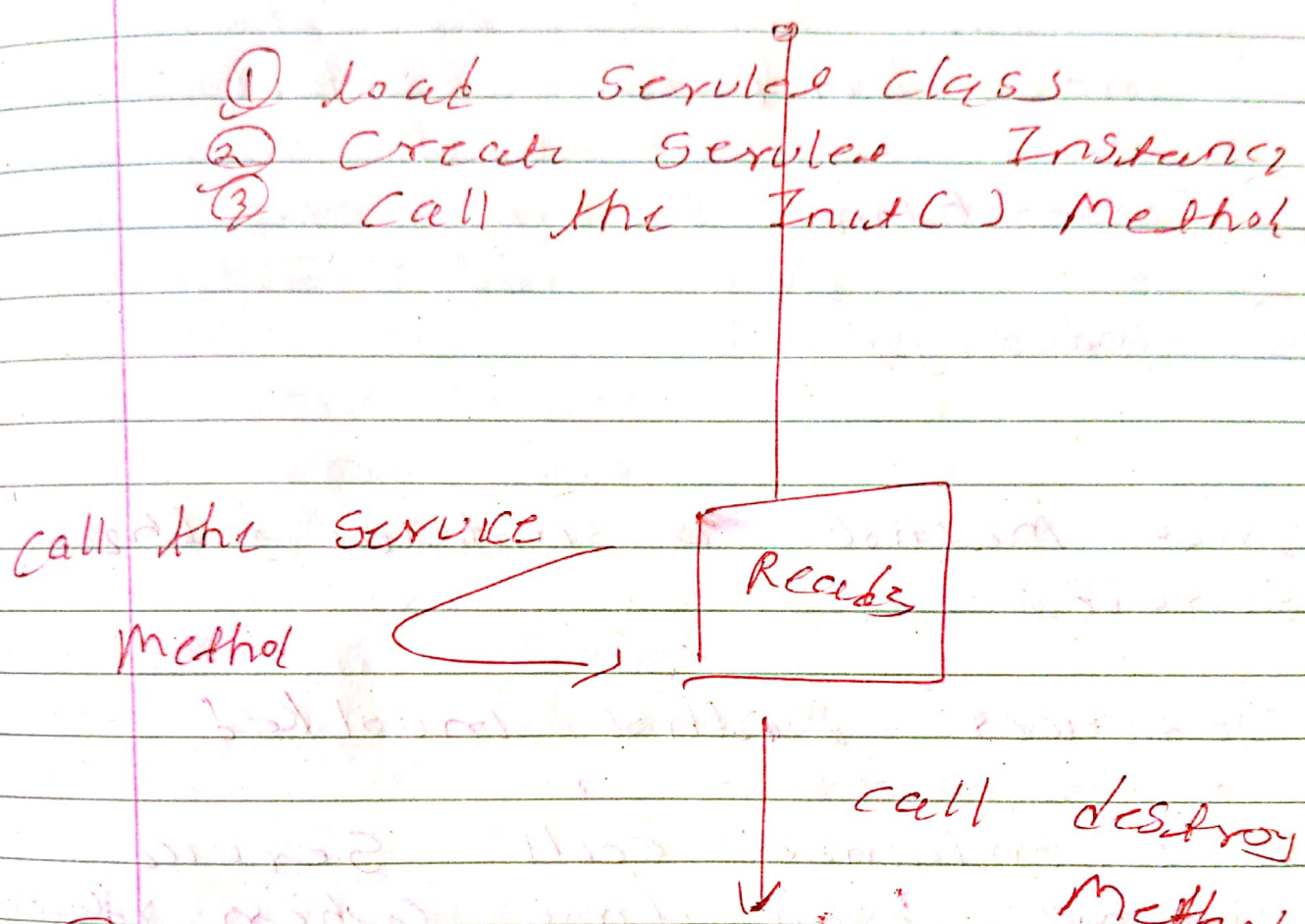
② doPost
(doHead, doOption
, doPut, doDelete

③ public void Service
{
ServletRequest res,
ServletResponse req

}



* Servlet lifecycle



1. Servlet class load
   → The classLoader is responsible to load the Servlet class
2. Servlet Instance Created
   → it is created after load Servlet class. The Servlet Instance created only in Servlet life cycle
3. Init method is Invoked
   Web container call Init Method after creating instance of Servlet

Prporuts execution of Prosed

① welcome - file list in web.xml

② index.html

③ index.htm

④ index.jsp

Init Method is initialize the Servlet

⑤ Service Method invoked

- web container call Service Method each time when request for the Servlet is received. If instance is not created then follow step 1 to 3

⑥ destroy

- The web container call the destroy method before removing the Servlet Interface from service it gives the Servlet opportunity to clean up the resource from memory

web.xml

Servlet Code to Print Hello

```
import javax.servlet.http.*;
import javax.servlet.*
import java.io.*;

public class demo extends HttpServlet
{
Public void doGet( HttpServletRequest req
, HttpServletResponse res) throw
      Exception
{
   res.setContentType("text/html");
   PrintWriter pw=res.getWriter();
   pw.println("<html><body>");

   pw.println("welcome to Servlet");

   pw.println("</body></html>");
   pw.close();
}
}
```

01 web.xml
URI pattern Match

Servlet Request Interface

- object of Servlet Request is used to provide the client request information to a Servlet such as content type, length, parameter name value header attribute etc

Common basic Method of Servlet Request

1) public String getParameter(String name)
2) public String[] getParameterValues(" ")
3) public int getContentLength();
4) public String getCharacterEncoding()
5) public abstract String getServerName()
6) public int getServerPort()

String s = req.getParameter("name");

RequestDispatcher in Servlet

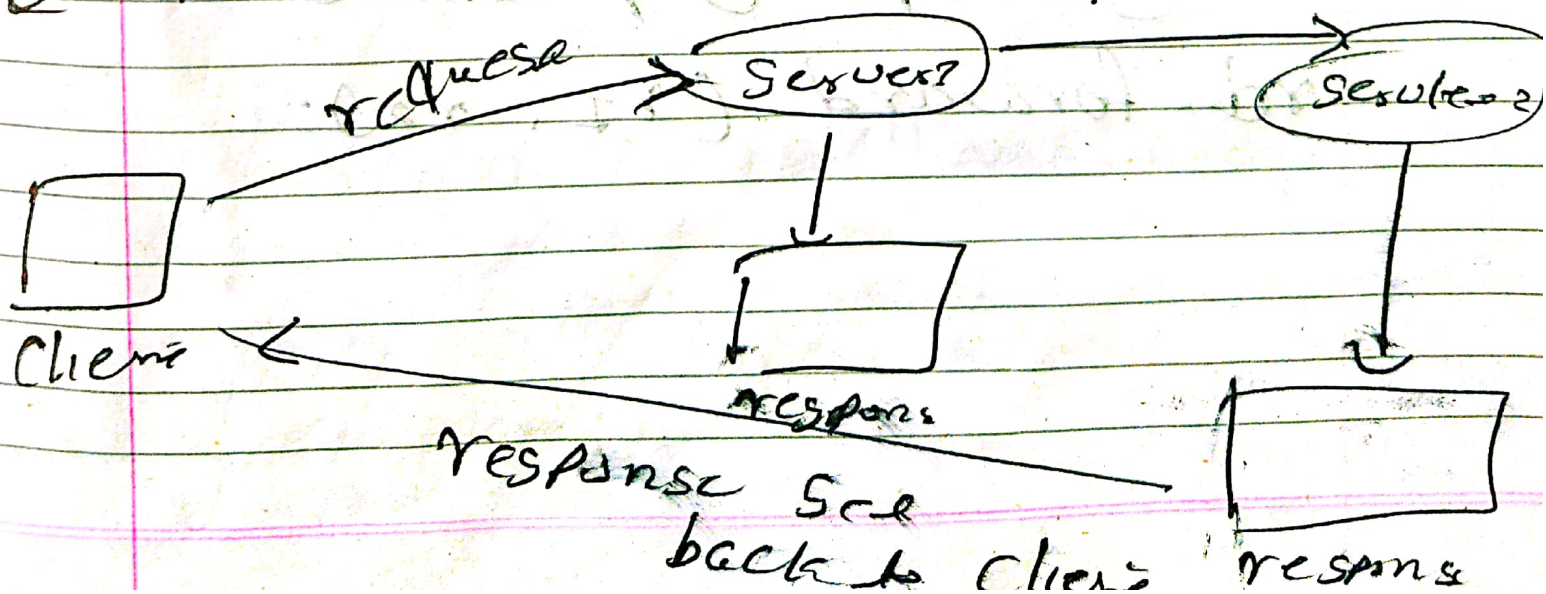
(1) RequestDispatcher Interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp

* Method of RequestDispatcher Interface

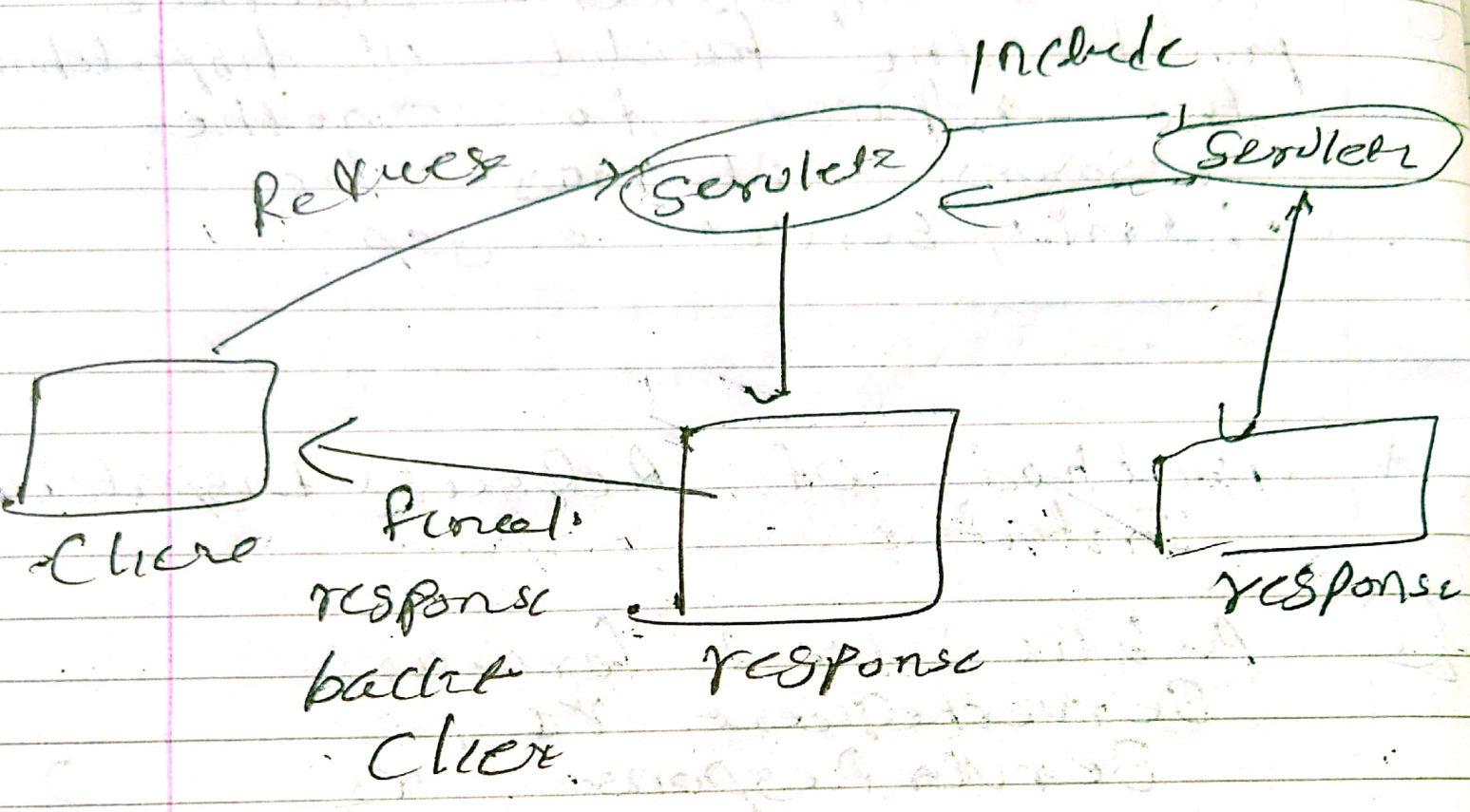
(1) public void forward (ServerRequest r1, ServerResponse r2)

(2) public void include (ServerRequest r1, ServerResponse r2)

(1) forward forward



(2) Include (w



Method

RequestDispatcher rd = request
.getRequestDispatcher("second");
rd.forward(r1, r2);

index.html

web.xml

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class first extends HttpServlet
{

public void Service
(HttpServletRequest r1
 HttpServlet Response r2 )
{
    r2.setContentType("text/html");
PrintWriter pw = r2.getWriter();
String s1 = r1.getParameter("uname");
if( s1.equals("admin") )
  {

>

}
else{ out.println("Not found");
RequestDispatcher rd
     = request.getRequestDispatcher
       ("/index.html");
rd.response( r1, r2);
       include.
}
```

Second Servlet

```
{ response.setContentType("text/html");
  PrintWriter pw = response.
                   getWriter();

String s1 = r1.getParameter
            ("Uname");
pw.println("welcome"+s1);

}
```

web.xml


```
<web-app>
<Servlet>
<Servlet-name>      JP          </Servlet-na
<Servlet-class>     first       <Servlet-class

</Servlet>
<Servlet-mapping>
<Servlet-name>      JP          <Servlet-na
<url-pattern>       /first      < URL-Pa -->

</Servlet-mapping>
```

ServletConfig

ServletConfig object is used to get Configuration information from web.xml file

Advantage

You don't need to edit servlet file if information modified in web.xml

* Methods of ServletConfig Interface

1. public String getInitParameter(String)
2. public Enumeration getInitParameterNames()
3. Public String getServletName()
4. public ServletContext getServletContext()
   return object of servlet context

* getServletConfig() Method of Servlet Interface return object of ServletConfig

*
```
ServletConfig con = getServletConfig();
String s = con.getInitParameter("driver")
                                "username"
                                "password"
```

~~ServletContext~~

<web-app>

<Servlet>

<init-param>

<param-name> driver </param-name>

<param-value> &q; </param-value>

</init-param>

</Servlets>

</web-app>

-> printWriter pw = response.getWriter();

pw.println("driver " + s);

* Servlet Context

An object of ServletContext is created by the web container at time of deployment the project

- It is also used to get configuration information of web

→ There is only one servlet context object per web application

→ It is created in context.xml

* Advantages of Servlet Context

- Easy to maintain:
If any information is shared to all the servlet then it is better to store in servlet context.

Common Method
Same as previous
+
public void setAttribute(String name, Object o)
public Object getAttribute(String name)
public void removeAttribute(String name)

web.xml

<web-app>

<context-param>

<param-name> AAkash </param-name>

<param-value> 78.1 </param-value>

</context-param>

<context-param>

<servlet>

<servlet-name>

<servlet-class>

</servlet>

</servlet-mapping>

<servlet-name>

<URL-pattern> / first

</servlet-mapping>

</web-app>

```
{
        ServiceMethod

{
    r2.setContentType("text/html");
PrintWriter pw = r2.getWriter();
ServletContext sc = getServletContext
String s1 = sc.getInitParameter
                Names("Akash");
pw.println("per" + s1);

}
    above in Command
```

Set Attribute

```
ServletContext sc = getServletContext();
sc.setAttribute("IT", "Eng");
out.println("<a href='second'> visit </a>");
```

Get Attribute

second

```
ServletContext sc = getServletContext();
String sz = (String) sc.getAttribute("IT");
out.println("category" + sz);
```

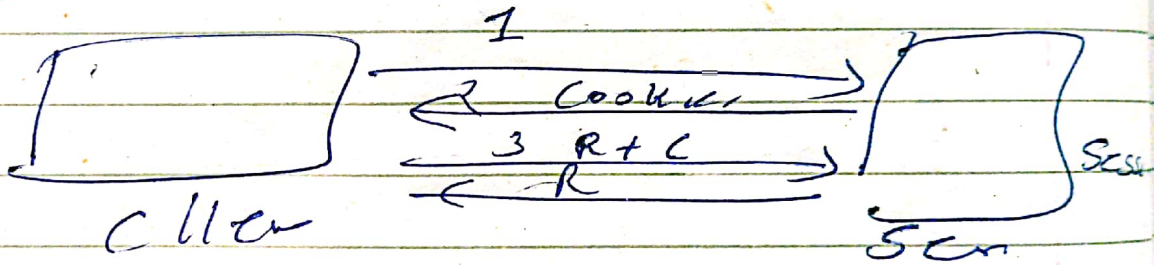
Session

Session

Session Mean particular Interval of time

Session Tracking is Concept of maintain state of an user-id is also known as Session Management in Server

HTTP



X Session Tracking Technique

1. Cookies
2. Hidden form field
3. URL Rewriting
4. Http Session

Fidter & Types of fidter