Q1. What is Servlet? List and Explain various stages of Servlet life cycle. Explain role of web container.

Ans. **What is Servlet?**

Servlet is java class which extends the functionality of web server by dynamically generating web pages. Servlet technology is used to create Dynamic web application.

List and Explain various stages of Servlet life cycle

In the life cycle of servlet there are three important methods. These methods are

1. init()
2. service()
3. destroy()

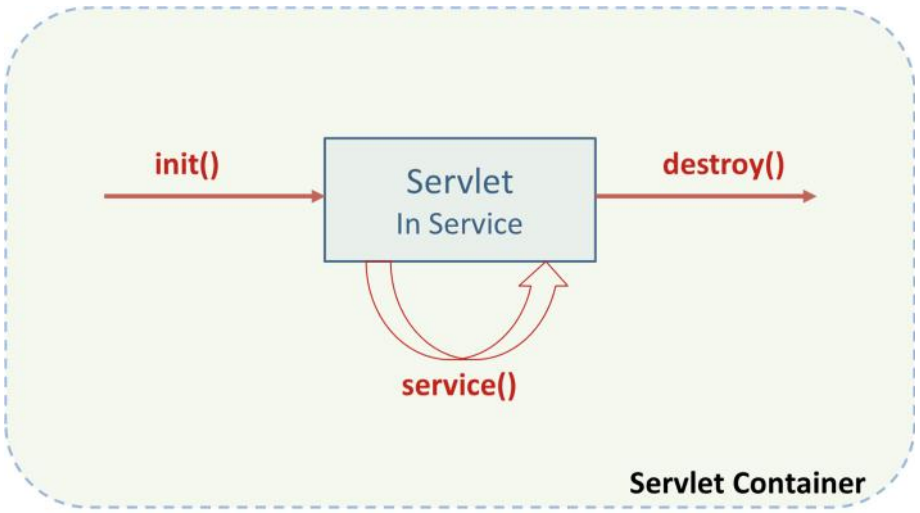


Figure: Servlet Life Cycle

- The client enters the URL in the web browser and makes a request. The browser then generates the HTTP request and sends it to the Web server.
- Web server maps this request to the corresponding servlet.

init()

- The server basically invokes the **init()** method of servlet. This method is called only when the servlet is loaded in the memory for the first time.
- The class loader is responsible to load the servlet class.
- The servlet class is loaded when the first request for the servlet is received by the web container.
- The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

- The web container calls the init method only once after creating the servlet instance. The init() method is used to initialize the servlet.

```
public void init(ServletConfig config)throws ServletException
{
          //Servlet Initialization…
}
```

- A servlet configuration object used by a servlet container to pass information to a servlet during initialization.

service()

- The service() method is the main method to perform the actual task.
- The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the response back to the client.
- Each time the server receives a request for a servlet, the server spawns a new thread and calls service.

```
public void service(ServletRequest request,
                    ServletResponse response)
                         throws ServletException, IOException
{
     //Servlet Task
}
```

destroy()

- Finally server unloads the servlet from the memory using the **destroy()** method.
- The destroy() method is called only once at the end of the life cycle of a servlet.
- This method gives your servlet a chance to close
  1. database connections,
  2. halt background threads,
  3. write cookie lists or hit counts to disk, and
  4. perform other such cleanup activities.
- After the destroy() method is called, the servlet object is marked for garbage collection.

```
public void destroy()
{
     // Finalization code...
}
```

Example

```
import java.io.*;
import javax.servlet.*;
public class MyServlet1 extends GenericServlet
{
public void init() throws ServletException
{//Initailization Code
}
public void service(ServletRequest request,
                    ServletResponse response)
                         throws ServletException,IOException
{    //Servlet code
     }

     public void destroy()
     {//Finalization Code
 }}
```

Q2. Differentiate Servlets and CGI

Ans.

CGI(Common Gateway Interface)	Servlet
CGI is not portable (as CGI programs written inside the native language).	Servlets are portable (written in java).
In CGI each request is handled by heavy weight OS process.	In Servlets each request is handled by lightweight Java Thread.
CGI is more expensive than Servlets, because For each request CGI Server receives, It creates new Operating System Process.	Servlets is inexpensive than CGI because In Servlet, All the requests coming from the Client are processed with the threads instead of the OS process.
Session tracking and caching of previous computations cannot be performed.	Session tracking and caching of previous computations can be performed
CGI cannot handle cookies	Servlets can handle cookies
CGI does not provide sharing property.	Servlets can share data among each other.

Q3. Differentiate GenericServlet and HttpServlet

Ans.

GenericServlet	HttpServlet
javax.servlet.GenericServlet (abstract class)	javax.servlet.http.HttpServlet (abstract class)
It is the immediate subclass of Servlet interface.	The immediate super class of HttpServlet is GenericServlet.
It defines a generic, protocol-independent servlet. it can be used with any protocol, say, SMTP, FTP, CGI including HTTP etc.	It defines a HTTP protocol specific servlet.
GenericServlet is a super class of HttpServlet class.	HttpServlet is a sub class of GenericServlet class.
All methods are concrete except service() method. service() method is abstract method.	All methods are concrete (non-abstract). service() is non-abstract method. service() can be replaced by doGet() or doPost() methods.

Q4. Differentiate doGet() vs doPost()

Ans.

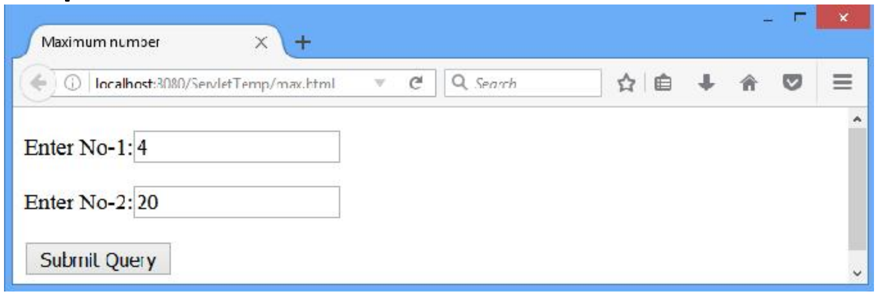
doGet()	doPost()
In doGet(), parameters are appended to the URL and sent along with header information	In doPost(), parameters are sent in separate line in the body
Maximum size of data that can be sent using doGet() is 240 bytes	There is no maximum size for data
Parameters are not encrypted	Parameters are encrypted here
Application: Used when small amount of data and insensitive data like a query has to be sent as a request. It is default method.	**Application**: Used when comparatively large amount of sensitive data has to be sent. E.g. submitting sign_in or login form.
doGet() is faster comparatively	doPost() is slower compared to doGet() since doPost() does not write the content length
doGet() generally is used to query or to get some information from the server	Dopost() is generally used to update or post some information to the server
This is default method of http	Not the case

Q5. Write a Servlet program using doPost() to enter two numbers and find maximum among them.

Ans. max.html

```
<html>
    <head>
        <title> Maximum number </title>
    </head>
    <body>
        <form action="/ServletTemp/Max" method="POST" >
            <p>Enter No-1:<input type="text" name="no1"></p>
            <p>Enter No-2:<input type="text" name="no2"></p>
            <p><input type="submit"></p>
        </form>
     </body>
 </html>
```

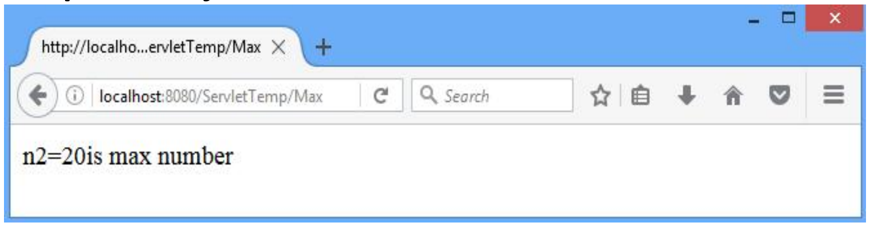
Output: max.html



Max.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Max extends HttpServlet
{   public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
                          throws ServletException,IOException
    {   int n1=0,n2=0;
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        n1=Integer.parseInt(request.getParameter("no1"));
        n2=Integer.parseInt(request.getParameter("no2"));

       if(n1>n2)
          out.println("n1="+n1+"is max number");
       else if(n2>n1)
15.         out.println("n2="+n2+"is max number");
16.      else if(n1==n2)
17.    out.println("n1= "+n1+"and n2="+n2+"are equal numbers");
18.        }
19.    }
```

Output:Max.java


http://localho...ervletTemp/Max
localhost:8080/ServletTemp/Max
Search

n2=20is max number

Q6. Explain ServletConfig with example.

Ans.

- It is used to get configuration information from web.xml file.
- If the configuration information is modified from the web.xml file, we don't need to change the servlet.
  E.g. `String str = config.getInitParameter("name")`

Advantage of ServletConfig

- The core advantage of ServletConfig is that you don't need to edit the servlet file if information is modified from the web.xml file.

How to get the object of ServletConfig

- **getServletConfig() method** of Servlet interface returns the object of ServletConfig.

Usage of ServletConfig

If any specific content is modified from time to time. you can manage the Web application easily without modifying servlet through editing the value in web.xml

E.g. `ServletConfig config=getServletConfig();`

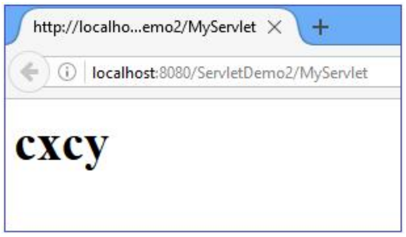
web.xml

```
<web-app>
    <servlet>
        <servlet-name>MyServlet</servlet-name>
        <servlet-class>MyServlet</servlet-class>
        <init-param>
            <param-name>name</param-name>
            <param-value>cxcy</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>MyServlet</servlet-name>
        <url-pattern>/MyServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

MySerlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class MyServlet extends HttpServlet
{   String msg;
    PrintWriter out;
public void init(ServletConfig config)throws
                                             ServletException
{        msg = config.getInitParameter("name"); }
public void doGet(HttpServletRequest request ,
                  HttpServletResponse response) throws
                         ServletException,IOException
    {   response.setContentType("text/html");
    out = response.getWriter();
    out.println("<h1>"+ msg +"</h1>");
    }
    public void destroy()
    {        out.close();    }}
```

Output: MyServlet.java



http://localho...emo2/MyServlet

localhost:8080/ServletDemo2/MyServlet

```
cxcy
```

Methods of ServletConfig interface

1. **public String getInitParameter(String name):**Returns the parameter value for the specified parameter name.
2. **public Enumeration getInitParameterNames():**Returns an enumeration of all the initialization parameter names.
3. **public String getServletName():**Returns the name of the servlet.
4. **public ServletContext getServletContext():**Returns an object of ServletContext.

Q7. Explain ServletContext with example

Ans.

- ServletContext is created by the web container at time of deploying the project.
- It can be used to get configuration information from web.xml file.
- There is only one ServletContext object per web application.
- If any information is shared to many servlet, it is better to provide it from the web.xml file using the <context-param> element.

Advantage of ServletContext

- **Easy to maintain** if any information is shared to all the servlet, it is better to make it available for all the servlet.
- We provide this information from the web.xml file, so if the information is changed, we don't need to modify the servlet. Thus it removes maintenance problem.

Usage of ServletContext

There can be a lot of usage of ServletContext object. Some of them are as follows:

1. The object of ServletContext provides an interface between the container and servlet.
2. The ServletContext object can be used to get configuration information from the web.xml file.
3. The ServletContext object can be used to set, get or remove attribute from the web.xml file.
4. The ServletContext object can be used to provide inter-application communication.

How to get the object of ServletContext interface

1. **getServletContext() method** of ServletConfig interface returns the object of ServletContext.

```
//We can get the ServletContext object from ServletConfig object
   ServletContext context=getServletConfig().getServletContext();
```

2. **getServletContext() method** of GenericServlet class returns the object of ServletContext.

```
//Another way to get the ServletContext object
ServletContext application=getServletContext();
```

Example of ServletContext

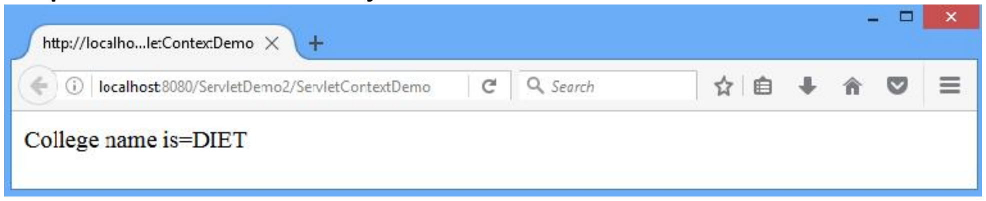
Web.xml

```
<web-app>
    <servlet>
        <servlet-name>ServletContextDemo</servlet-name>
        <servlet-class>ServletContextDemo</servlet-class>
    </servlet>


    <servlet-mapping>
        <servlet-name>ServletContextDemo</servlet-name>
        <url-pattern>/ServletContextDemo</url-pattern>
    </servlet-mapping>
    <context-param>
        <param-name>name</param-name>
        <param-value>DIET</param-value>
    </context-param>
</web-app>
```

ServletContextDemo.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ServletContextDemo extends HttpServlet{
    public void doGet(HttpServletRequest
req,HttpServletResponse                  res)  throws
ServletException,IOException
{   res.setContentType("text/html");
    PrintWriter out=res.getWriter();
    //creating ServletContext object
    ServletContext context=getServletContext();
  //Getting value of initialization parameter and printing it
    String college=context.getInitParameter("name");
    out.println("College name is="+college);
    out.close();
 }}
```

Output: ServletContextDemo.java



```
localhost:8080/ServletDemo2/ServletContextDemo
College name is=DIET
```

Methods of ServletContext interface

1. **public String getInitParameter(String name):**Returns the parameter value for the specified parameter name.
2. **public Enumeration getInitParameterNames():**Returns the names of the context's initialization parameters.
3. **public void setAttribute(String name,Object object):**sets the given object in the application scope.
4. **public Object getAttribute(String name):**Returns the attribute for the specified name.
5. **public Enumeration getInitParameterNames():**Returns the names of the context's initialization parameters as an Enumeration of String objects.
6. **public void removeAttribute(String name):**Removes the attribute with the given name from the servlet context.

Q8. Differentiate ServletConfig and ServletContext Interface.

Ans.

Servlet Config	Servlet Context
ServletConfig object is one per servlet class	ServletContext object is global to entire web application
Object of ServletConfig will be created during initialization process of the servlet	Object of ServletContext will be created at the time of web application deployment
Scope: As long as a servlet is executing, ServletConfig object will be available, it will be destroyed once the servlet execution is completed.	Scope: As long as web application is executing, ServletContext object will be available, and it will be destroyed once the application is removed from the server.
We should give request explicitly, in order to create ServletConfig object for the first time	ServletContext object will be available even before giving the first request
In web.xml – <init-param> tag will be appear under <servlet-class> tag	In web.xml – <context-param> tag will be appear under <web-app> tag

Q9. Explain Methods of HttpServletRequest with appropriate example.

Ans.

1.String **getContextPath**()	Returns the portion of the request URI that indicates the context of the request.

E.g.
```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
{out.println("<p>request.getContextPath():"
                              +request.getContextPath()+"</p>");
}
```
Output:request.getContextPath():/ServletTemp

2.Enumeration **getHeaderNames()**	Returns an enumeration of all the header names this request contains.

E.g.
```
 Enumeration h=request.getHeaderNames();
while(h.hasMoreElements())
{
  String paramName = (String)h.nextElement();
  out.print("<p>" + paramName + "\t");
  String paramValue = request.getHeader(paramName);
  out.println( paramValue + "</p>\n");
}
```

Output:

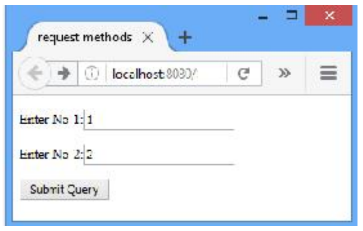
host localhost:8080
user-agent Mozilla/5.0 (Windows NT 6.2; WOW64; rv:50.0) Gecko/20100101 Firefox/50.0
accept text/html,application/xhtml+xml,
application/xml;q=0.9,*/*;q=0.8
accept-language en-US,en;q=0.5
accept-encoding gzip, deflate
connection keep-alive
upgrade-insecure-requests 1

3.String **getHeader**(String name)	Returns the value of the specified request header as a String.

E.g.
```
out.println("<p>request.getHeader():"
                                +request.getHeader("host")+"</p>");
out.println("<p>request.getHeader():"+request.getHeader("referer"
                                                        )+"</p>");
```
Output:
request.getHeader():host=localhost:8080
request.getHeader():referer=http://localhost:8080/ServletTemp/servletmeth.html

4. String **getQueryString**()	Returns the query string that is contained in the request URL after the path.

```
public void doGet(HttpServletRequest request,
              HttpServletResponse response)
{out.println("<p>request.getQueryString():"
         +request.getQueryString()+"</p>");}
```
Output:
request.getQueryString(): no1=1&no2=2



5.String **getServletPath**()	Returns the part of this request's URL that calls the servlet. This path starts with a "/" character and includes either the servlet name or a path to the servlet

E.g.
```
out.println("<p>request.getServletPath():"
                                +request.getServletPath()+"</p>");
```
Output: request.getServletPath(): /ServletMeth

6. String **getMethod**()	Returns the name of the HTTP method with which this request was made, for example GET or POST

E.g.
```
out.println("<p>request.getMethod():"
                                    +request.getMethod()+"</p>");
```
Output: request.getMethod(): GET

Q.10 Write servlet which displayed following information of client.
I. Client Browser
II. Client IP address
III. Client Port No
IV. Server Port No
V. Local Port No
VI. Method used by client for form submission
VII. Query String name and values

Ans.

```
import java.io.*;
import javax.servlet.http.*;
public class ServletInfo extends HttpServlet{
PrintWriter out;
public void doGet(HttpServletRequest req,HttpServletResponse
res)                                          throws
IOException
{
res.setContentType("text/html");
out=res.getWriter();
// I. Client Browser: we use String getHeader(user-agent)
out.println("<p> Client Browser=" +req.getHeader
("user-agent")+"</p>");
//II. Client IP address
out.println("<p> Client IP address= "+req.getRemoteAddr());
//III. Client Port No
out.println("<p> Client Port No= "+req.getRemotePort());
//IV. Server Port No
out.println("<p> Server Port No= "+req.getServerPort());
//V. Local Port No
out.println("<p> Local Port No= "+req.getLocalPort());
//VI. Method used by client for form submission
out.println("<p> Method used by client= "+req.getMethod());
//VII. Query String name and values
out.println("<p> Query String name & values=
                                    "+req.getQueryString());
}}
```

Output:

```
Client Browser=Mozilla/5.0 (Windows NT 6.2; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Client IP address= 0:0:0:0:0:0:0:1
Client Port No= 64779
Server Port No= 8080
Local Port No= 8080
Method used by client= GET
Query String name & values= null
```

Q11. What is Request Dispatcher? What is the difference between Request dispatcher's forward () and include () method?

Ans. javax.servlet.RequestDispatcher Interface

- The RequestDispatcher interface provides the facility of dispatching the request to another resource.
- Resource can be HTML, Servlet or JSP.
- This interface can also be used to include the content of another resource.
- It is one of the way of servlet collaboration.
- The **getRequestDispatcher**() method of ServletRequest interface returns the object of RequestDispatcher.

Syntax

```
        RequestDispatcher getRequestDispatcher(String resource)
```

Example

```
RequestDispatcher rd=request.getRequestDispatcher("servlet2");
rd.forward(request, response);//method may be include/forward
```

There are two methods defined in the **RequestDispatcher interface**

void **forward**(ServletRequest request, ServletResponse response) throws ServletException, IOException	Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
void **include**(ServletRequest request, ServletResponse response) throws ServletException, IOException	Includes the content of a resource (Servlet, JSP page, or HTML file) in the response.

RequestDispatcher: forward()

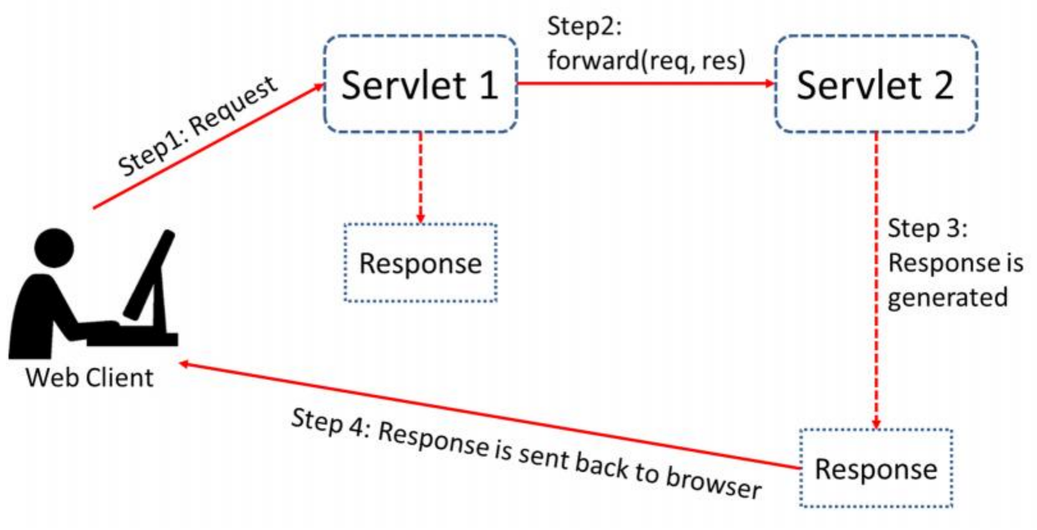


Figure: Working of RequestDispatcher.forward()

Example: forward()
//for java servlet
```
RequestDispatcher rd =   request.getRequestDispatcher("servlet2");
rd.forward(request, response);
```
//for html page
```
RequestDispatcher rd= request.getRequestDispatcher("/1.html");
rd.forward(request, response);
```

RequestDispatcher: include()

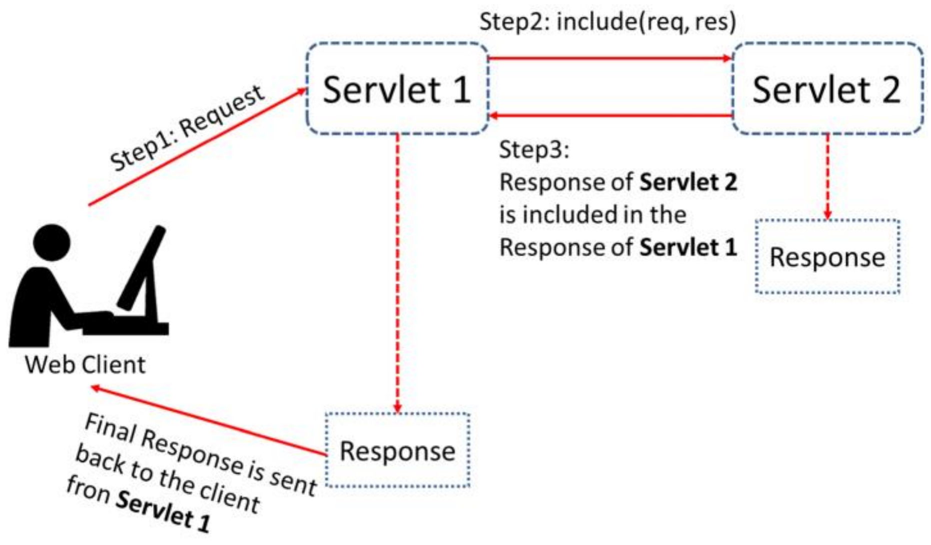

Step2: include(req, res)
Servlet 1
Servlet 2
Step1: Request
Step3:
Response of **Servlet 2** is included in the Response of **Servlet 1**
Response
Web Client
Response
Final Response is sent back to the client fron **Servlet 1**

Figure: Working of RequestDispatcher.include()

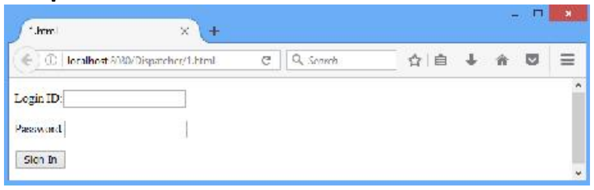
Example: include()
//for java servlet
```
RequestDispatcher rd=request.getRequestDispatcher("servlet2");
rd.include(request, response);
```
//for html page
```
RequestDispatcher rd=request.getRequestDispatcher("/1.html");
rd.include(request, response);
```

Q12. Write a Servlet program to authenticate user with user_id and password, if user is authenticated then forward to welcome page else include message with invalid user_id/password.

Ans. **1.html**

```
<html>
    <head>
        <title>1.html</title>
    </head>
    <body>
        <form action="/Dispatcher/CallServlet" method="POST">
           <p>Login ID:<input type="text" name="login"></p>
           <p>Password:<input type="text" name="pwd"></p>
           <p><input type="submit" value="Sign In"></p>
             </form>
          </body>
     </html>
```

Output:



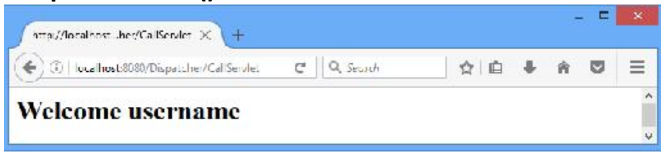
CallServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class CallServlet extends HttpServlet
{   public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
                         throws ServletException,IOException
    {   response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        RequestDispatcher rd;
        String login=request.getParameter("login");
        String pwd=request.getParameter("pwd");
        if(login.equals("java") && pwd.equals("servlet"))
             {   rd=request.getRequestDispatcher("FwdDemo");
                 rd.forward(request, response);
        }//if
        else
        {   out.println("<p><h1>Incorrect Login Id/Password
                                             </h1></p>");
                 rd=request.getRequestDispatcher("/1.html");
                 rd.include(request, response);   }//else
     }//dopost }
```

FwdDemo.java

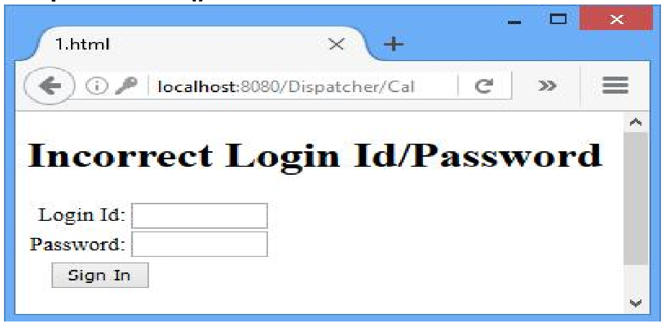
```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class FwdDemo extends HttpServlet{
public void doPost(HttpServletRequest request,
                    HttpServletResponse response)
                          throws ServletException,IOException
     {   response.setContentType("text/html");
         PrintWriter out=response.getWriter();
         String username=request.getParameter("login");
          out.println("<h1>"+"Welcome "+username+"</h1>");
          }
     }
```

Output:forward()



```
Welcome username
```

Output:include()



```
Incorrect Login Id/Password
Login Id:
Password:
Sign In
```

Q13. Explain response.sendRedirect() with appropriate example.

Ans. The **sendRedirect**() method of HttpServletResponse interface can be used to redirect response to another resource, it may be servlet, jsp or html file.

Syntax

```
void sendRedirect(String location) throws IOException
```

Example

```
response.sendRedirect("http://www.darshan.ac.in");
response.sendRedirect("/1.html");//relative path
response.sendRedirect("http://localhost:8080/1.html");
```

```
                                                    //absolute path
```

Program: sendRedirect()

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Redirect extends HttpServlet
{   public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                         throws ServletException,IOException
    {   response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        String login=request.getParameter("login");
             String pwd=request.getParameter("pwd");
             if(login.equals("java") && pwd.equals("servlet"))
             {  response.sendRedirect("/Dispatcher/Welcome");
             }
          else
          response.sendRedirect("/Dispatcher/redirect.html");
         } //doGet
     }
```

Q14. Differentiate forward() and sendRedirect()

Ans.

forward()	sendRedirect()
forward() is method of RequestDispatcher interface	sendRedirect() is the method of HttpServletResponse
In forward(), redirect happens at server end and not visible to client.	In sendRedirect() ,redirection happens at client end and it's visible to client.
It is faster than the redirect.	It is slower than a forward, since it requires two browser requests(one for actual request and another for redirected request).
In case of forward() original URL remains unaffected.	While in case of sendRedirect() browser knows that it's making a new request, so original URL changes.
The request is transfer to other resource within same server. Thus, it can be used within server.	The request may transfer to other resource to different server. Thus, it can be used within and outside the server.
When forward is called on RequestDispather object we pass request and response object so our old request object is present on new resource which is going to process our request.	In case of SendRedirect call old request and response object is lost because it's treated as new request by the browser.
Syntax: forward(ServletRequest request, ServletResponse response)	**Syntax:** void sendRedirect(String url)

Q15. What is Session? Why we require Session? List the different ways to manage the session.

Ans. Define Session

"A session refers to the entire interaction between a client and a server from the time of the client's first request, which generally begins the session, to the time of last request/response."

Why we require Session?

- HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.
- Session is required to keep track of users and their information.

List the different ways to manage the session.

- Session Management is a mechanism used by the Web container to store session information for a particular user.
- There are four different techniques for session management.
    1. Hidden form field
    2. URL Rewriting
    3. Cookies
    4. HttpSession

Q16. Explain Hidden form field with appropriate example.

Ans.

- Hidden Form Field, a hidden (invisible) textfield is used for maintaining the state of an user.
- In such case, we store the information in the hidden field and get it from another servlet.

  E.g. `<input type="hidden" name="session_id"   value="054">`

Real application of hidden form field

- It is widely used in comment form of a website.
- In such case, we store page id or page name in the hidden field so that each page can be uniquely identified.

Advantage of Hidden Form Field

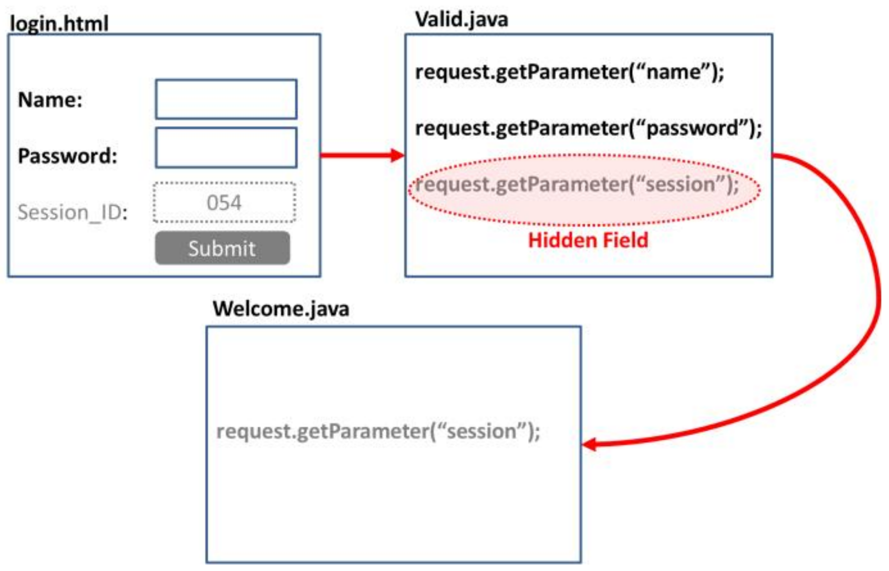
- Easy to implement
- It will always work whether cookie is disabled or not.

Disadvantage of Hidden Form Field:

- It is maintained at server side.
- Extra form submission is required on each pages.
- Only textual information can be used.
- It does not support hyperlink submission.
- Security

- Hidden field will be visible with GET method
- User might view page source and can view hidden field

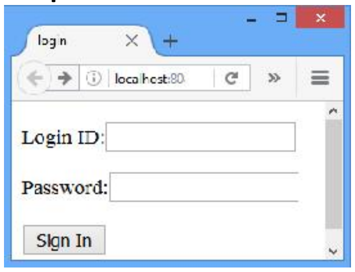
Program: Hidden form field


login.html
Name:
Password:
Session_ID:
054
Submit
Valid.java
request.getParameter("name");
request.getParameter("password");
request.getParameter("session");
Hidden Field
Welcome.java
request.getParameter("session");

login.html

```
<html>
    <head>
        <title>login</title>
    </head>
 <body>
 <form action="/Session/Valid" method="POST">
   <p>Login ID:<input type="text" name="login"></p>
   <p>Password:<input type="text" name="pwd"></p>
   <p><input type="hidden" name="session_id"
               value="054"></p>
          <p><input type="submit" value="Sign In"></p>
      </form>
     </body>
    </html>
```

Output:



Valid.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Valid extends HttpServlet
{   public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
                          throws ServletException,IOException
  {
     response.setContentType("text/html");
     PrintWriter out=response.getWriter();
          RequestDispatcher rd;
          String login=request.getParameter("login");
          String pwd=request.getParameter("pwd");
     String session=request.getParameter("session_id");
     if(login.equals("java") && pwd.equals("servlet"))
     {
          rd=request.getRequestDispatcher("Welcome");
          rd.forward(request, response);
     }//if
     else
     {
          out.println("<p><h1>Incorrect LoginId/Password
                                          </h1></p>");
          rd=request.getRequestDispatcher("/login.html");
          rd.include(request, response);
      }//else
     } }
```

Welcome.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Welcome extends HttpServlet
{   public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
                    throws ServletException,IOException
   {   response.setContentType("text/html");
       PrintWriter out=response.getWriter();
       String session=request.getParameter("session_id");
            String username=request.getParameter("login");
            out.println("<h1>"+"id:"+session+"</h1>");
            out.println("<h3>"+"Welcome "+username+"</h3>");
        }
     }
```

Q17. Explain URL Rewriting with appropriate example.

Ans.

- In URL rewriting, a token or identifier is appended to the URL of the next Servlet or the next resource.
- We can send parameter name/value pairs using the following format:

  `URL ? Name1 = value1 & name2 = value2 &…`
- Here, A name and a value is separated using an equal (=) sign and name/value pair is separated from another parameter using the ampersand(&).
- When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server.
- From a Servlet, we can use **getParameter()** method to obtain a parameter value.

Advantage of URL Rewriting

- It will always work whether cookie is disabled or not (browser independent).
- Extra form submission is not required on each pages.

Disadvantage of URL Rewriting

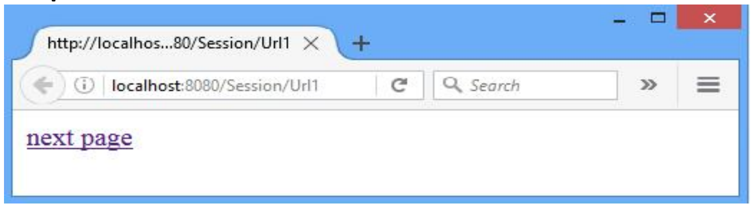
- It will work only with links.
- It can send only textual information.
- URL header size constraint.
- Security
  - name/value field will be visible with URL followed by '?'.

Program: URL Rewriting

Url1.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Url1 extends HttpServlet
{   public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException,IOException
 {  String url;
    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
   //for URL rewriting
   url= "http://localhost:8080/Session
              /Url2?s_id1=054&s_id2=055";
   out.println("<a href="+url+">next page</a>");
 } }
```

Output:

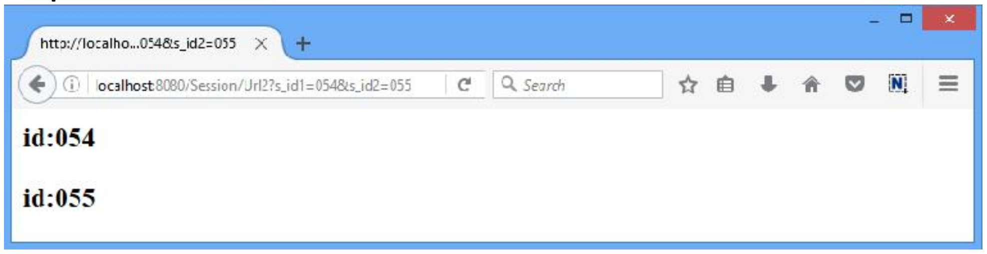

http://localhos...80/Session/Url1
localhost:8080/Session/Url1
Search

next page

Url2.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Url2 extends HttpServlet
{   public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                         throws ServletException,IOException
    {   response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        String session1=request.getParameter("s_id1");
        String session2=request.getParameter("s_id2");
        out.println("<h3>"+"id:"+session1+"</h3>");
        out.println("<h3>"+"id:"+session2+"</h3>");
    }
}
```

Output:


http://localho...054&s_id2=055
localhost:8080/Session/Url2?s_id1=054&s_id2=055
Search

id:054

id:055

Q18. What is Cookie? What does the cookie contains? Explain methods of Cookie class.

Ans. What is Cookie? What does the cookie contains?

- A cookie is a small piece of information that is persisted between the multiple client requests.
- A cookie has a
  1. Name
  2. Single value
  3. Optional attributes such as
     i. comment
     ii. path
     iii. domain qualifiers
     iv. a maximum age
     v. Version number etc.

Types of Cookie

There are 2 types of cookies in servlets.

1. **Non-persistent cookie/Session cookie:** It is **valid for single session** only. It is removed each time when user closes the browser.
2. **Persistent cookie:** It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Cookie class(javax.servlet.http.Cookie)

- This class provides the functionality of using cookies.
- It provides a lots of useful methods for cookies.

Constructor

Cookie(String name, String value): constructs a cookie with a specified name and value.

Example

```
Cookie c = new Cookie("session_id","054");
                                     //creating cookie object
```

Methods of Cookie class

void setMaxAge(int expiry)	Sets the maximum age in seconds for this Cookie
int getMaxAge()	Gets the maximum age in seconds of this Cookie. By default, -1 is returned, which indicates that the cookie will persist until browser shutdown.
String getName()	Returns the name of the cookie. The name cannot be changed after creation.
void setValue(String newValue)	Assigns a new value to this Cookie.
String getValue()	Gets the current value of this Cookie.

Other Methods of HttpServletRequest & HttpServletResponse	
void addCookie (Cookie cookieObj)	Method of HttpServletResponse interface is used to add cookie in response object.
Cookie[] getCookies()	Returns an array containing all of the Cookie objects the client sent with this request. This method returns null if no cookies were sent.

How to create Cookie?

```
//creating cookie object
     Cookie c= new Cookie("session_id","054");

//adding cookie in the response from server to client
     response.addCookie(c);
```

How to retrieve Cookies?

```
Cookie c[]=request.getCookies();
for(int i=0;i<c.length;i++)
{
     out.print(c[i].getName()+””+c[i].getValue());
          //printing name&value of cookie
}
```

How to delete Cookie?

1. Read an already existing cookie and store it in Cookie object.
2. Set cookie age as zero using setMaxAge() method to delete an existing cookie
3. Add this cookie back into response header.

```
//deleting value of cookie
  Cookie c = new Cookie("user","");
//changing the maximum age to 0 seconds
  c.setMaxAge(0);
//adding cookie in the response
  response.addCookie(c);
```

Advantage of Cookies

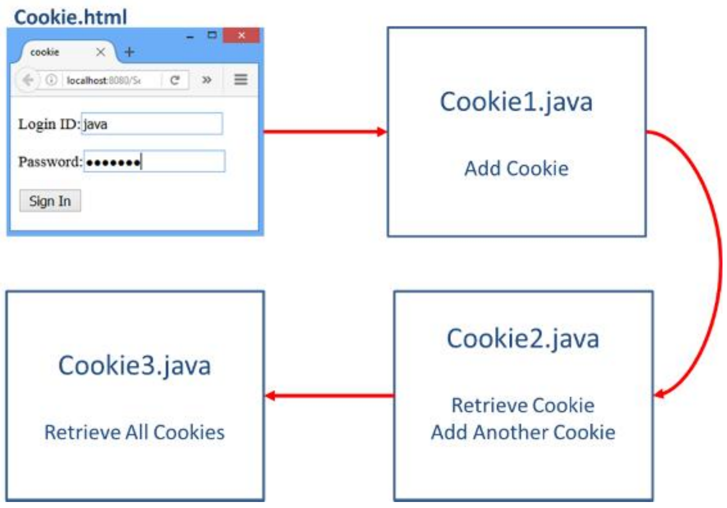
- Simplest technique of maintaining the state.
- Cookies are maintained at client side.

Disadvantage of Cookies

- It will not work if cookie is disabled from the browser.
- Only textual information can be set in Cookie object.

Q19. Write a Servlet program for Session Management using Cookie.

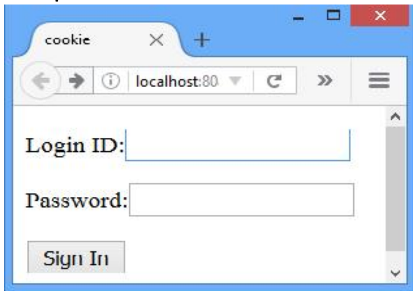
Ans.



cookie.html

```
<html>
<head>
<title>cookie</title>
</head>
<body>
<form action="/Session/Cookie1" >
      <p>Login ID:<input type="text" name="login"></p>
<p>Password:<input type="password" name="pwd"></p>
<p><input type="submit" value="Sign In"></p>
</form>
      </body>
      </html>
```

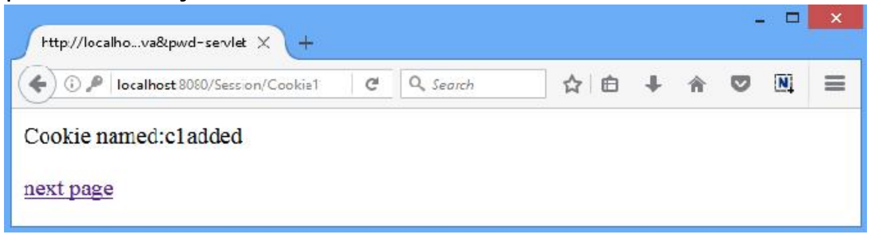
Output: **cookie.html**



Cookie1.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Cookie1 extends HttpServlet
{   public void doGet(HttpServletRequest request,
             HttpServletResponse response)
                             throws ServletException,IOException
    {   response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        String login=request.getParameter("login");
        String pwd=request.getParameter("pwd");

        if(login.equals("java") && pwd.equals("servlet"))
        {
          Cookie c = new Cookie("c1",login);//create cookie
           response.addCookie(c);//adds cookie with response

           out.println("Cookie named:"+c.getName()+" added");
           String path="/Session/Cookie2";
           out.println("<p><a href="+path+">next page</a></p>");
             }
         else
     {
     out.println("<p><h1>Incorrect Login Id/Password </h1></p>");
                 rd=request.getRequestDispatcher("/cookie.html");
                 rd.include(request, response);}
     } }
```

Output: **Cookie1.java**

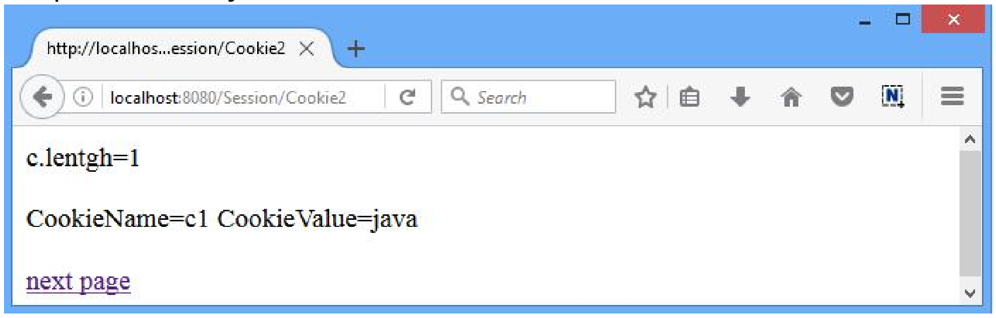

http://localho...va&pwd-servlet
localhost:8080/Session/Cookie1
Search

```
Cookie named:c1added

next page
```

Cookie2.java

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

public class Cookie2 extends HttpServlet
{   public void doGet(HttpServletRequest request,
                        HttpServletResponse response) throws
                        ServletException,IOException
    {
     response.setContentType("text/html");
      PrintWriter out=response.getWriter();
       Cookie c[]=request.getCookies();
       out.println("c.length="+c.length);
       for(int i=0;i<c.length;i++)
       {    out.println("CookieName="+c[i].getName()+
                         "CookieValue="+c[i].getValue());
       }
        //to add another cookie
           Cookie c1 = new Cookie("c2","054");
           response.addCookie(c1);
           String path="/Session/Cookie3";
           out.println("<a href="+path+">next page</a>");
 }}
```

Output: **Cookie2.java**



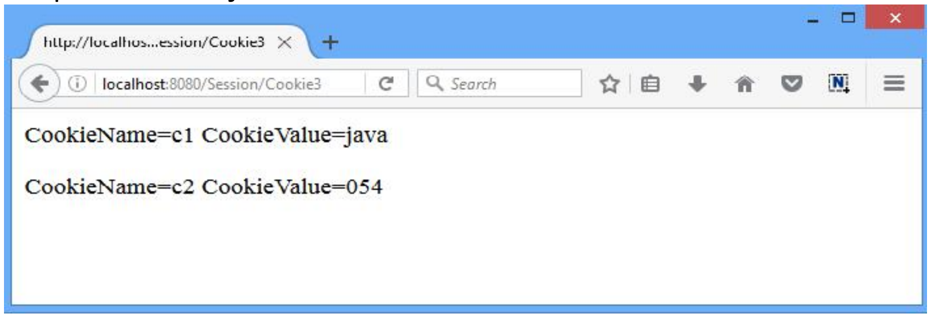
```
c.lentgh=1

CookieName=c1 CookieValue=java

next page
```

Cookie3.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Cookie3 extends HttpServlet
{   public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
                        throws ServletException,IOException
    {  response.setContentType("text/html");
       PrintWriter out=response.getWriter();
       Cookie c[]=request.getCookies();
            for(int i=0;i<c.length;i++)
```

```
                { out.println("<p>");
                  out.println("CookieName="+c[i].getName()+
                              "CookieValue="+c[i].getValue());
                  out.println("</p>");
             }
          }
       }
```

Output: **Cookie3.java**



```
CookieName=c1 CookieValue=java

CookieName=c2 CookieValue=054
```

http://localhos...ession/Cookie3
localhost:8080/Session/Cookie3
Search

Q20. Explain Session Management using HttpSession.

Ans.

- HttpSession Interface(javax.servlet.http.**HttpSession**) provides a way to identify a user across more than one page request
- The container creates a session id for each user.
- The container uses this id to identify the particular user.
- An object of HttpSession can be used to perform two tasks:
    1. Bind objects
    2. View and manipulate information about a session, such as the session identifier, creation time, and last accessed time.
- **HttpSession** object is used to store entire session with a specific client.
- We can store, retrieve and remove attribute from **HttpSession** object.
- Any servlet can have access to **HttpSession** object throughout the getSession() method of the **HttpServletRequest** object.
- The servlet container uses this interface to create a session between an HTTP client and an HTTP server.
- In this technique create a session object at server side for each client.
- Session is available until the session time out, until the client log out.
- The default session time is 30 minutes and can configure explicit session time in web.xml file.

Working of HttpSession

1. On client's first request, the Web Container generates a unique **session ID** and gives it back to the client with response.
2. The client sends back the session ID with each request. Making it easier for the web container to identify where the request is coming from.
3. The Web Container uses this ID, finds the matching session with the ID and associates the session with the request.

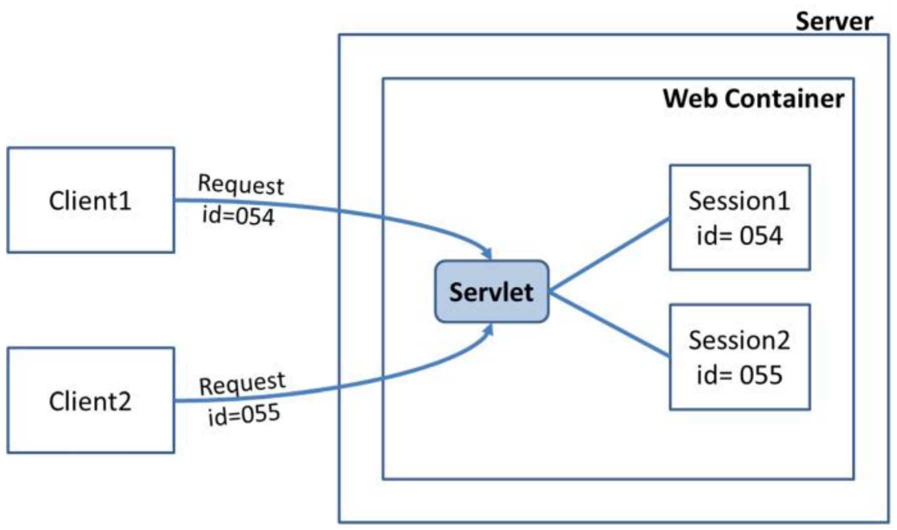


Figure: Working of HttpSession

Methods of HttpServletRequest interface:

HttpSession getSession()	Returns the current session associated with this request, or if the request does not have a session, creates one.
HttpSession getSession(boolean create)	Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.
String getId()	Returns a string containing the unique identifier value.
long getCreationTime()	Returns the time when this session was created, measured in milliseconds.
long getLastAccessedTime()	Returns the last time the client sent a request associated with this session, as the number of milliseconds.
void invalidate()	Invalidates this session then unbinds any objects bound to it.

How to create the Session?

```
HttpSession hs=request.getSession();
hs.setAttribute("s_id", "diet054");
```

How to retrieve a Session?

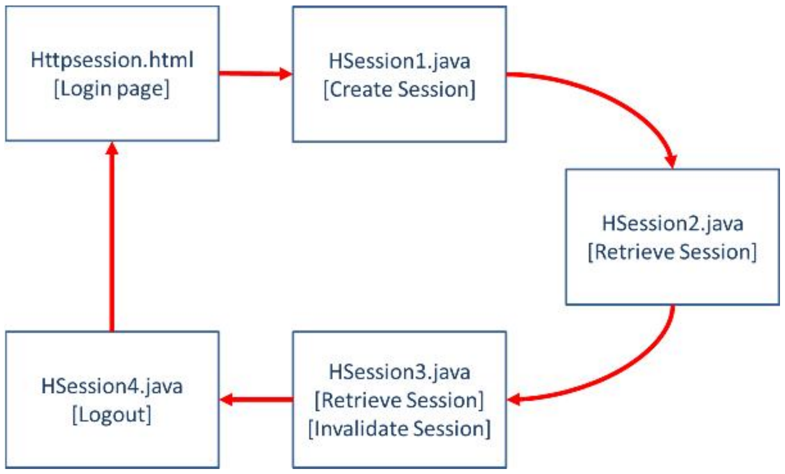
```
HttpSession hs=request.getSession(false);
String n=(String)hs.getAttribute("s_id");
```

How to invalidate a Session?

```
hs.invalidate();
```

Q21. Write a servlet program for Session Management using HttpSession.

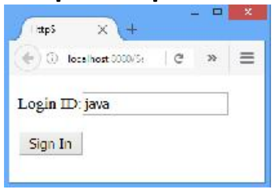
Ans.



Httpsession.html

```
<html>
<head>
<title>HttpSession</title>
</head>
<body>
<form action="/Session/HSession1" method="Get">
  <p>Login ID:<input type="text" name="login"></p>
  <p><input type="submit" value="Sign In"></p>
</form>
    </body>
    </html>
```

Output:Httpsession.html

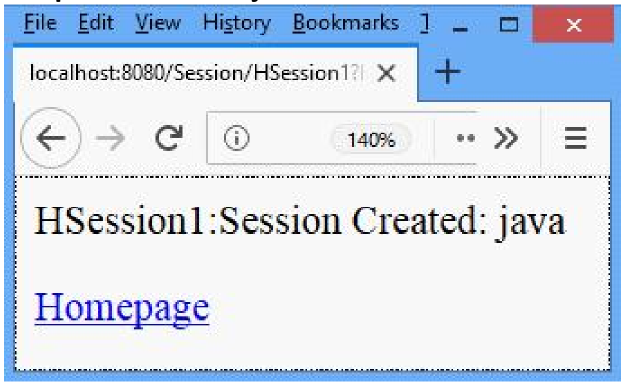


Login ID: java

HSession1.java

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class HSession1 extends HttpServlet
{   public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                           throws ServletException,IOException
{    response.setContentType("text/html");
     PrintWriter out=response.getWriter();
     RequestDispatcher rd;
     String login=request.getParameter("login");
     if(login.equals("java") )
     {    HttpSession hs=request.getSession();
          hs.setAttribute("s_id",login);
          out.println("<p> HSession1:Session Created:
                         "+hs.getAttribute("s_id")+"</p>");
          out.print("<a href='HSession2'>Homepage</a>");
     }
     else
     {   out.println("<p><h1>Incorrect Login Id/Password
                                         </h1></p>");
         rd=request.getRequestDispatcher("/httpsession.html");
         rd.include(request, response);
     }
     } }
```

Output:HSession1.java


File Edit View History Bookmarks
localhost:8080/Session/HSession1?l
140%

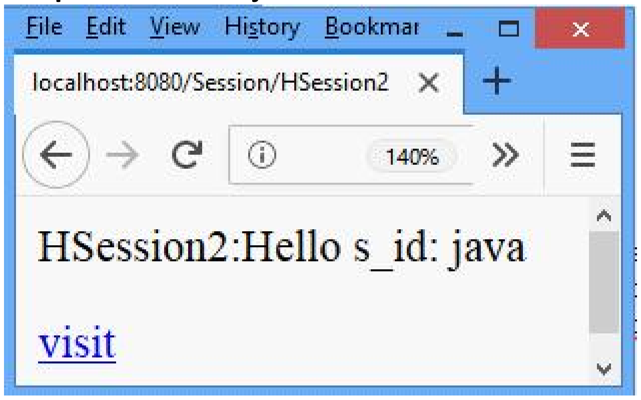
HSession1:Session Created: java

Homepage

HSession2.java

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class HSession2 extends HttpServlet
{   public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                           throws ServletException,IOException
{    response.setContentType("text/html");
     PrintWriter out=response.getWriter();
     HttpSession hs=request.getSession(false);
     String n=(String)hs.getAttribute("s_id");
     out.print("HSession2:Hello s_id: "+n);
     out.print("<p><a href='HSession3'>visit</a></p>");
     } }
```

Output:HSession2.java


File Edit View History Bookmar
localhost:8080/Session/HSession2
140%

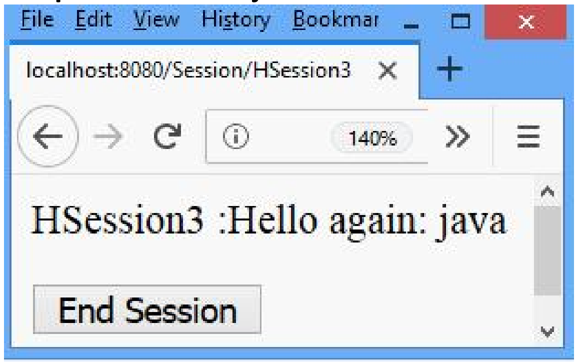
HSession2:Hello s_id: java

visit

HSession3.java

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class HSession3 extends HttpServlet
{   public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                          throws ServletException,IOException
{    response.setContentType("text/html");
     PrintWriter out=response.getWriter();
     HttpSession hs=request.getSession(false);
     String n=(String)hs.getAttribute("s_id");
     out.print("HSession3 :Hello again: "+n);
     out.println("<p><form action='/Session/HSession4'></p>");
     out.println("<p><input type='submit' value='End
                                    Session'></p></form>");
     hs.invalidate();//Session Invalidated
     }}
```

Output:HSession3.java

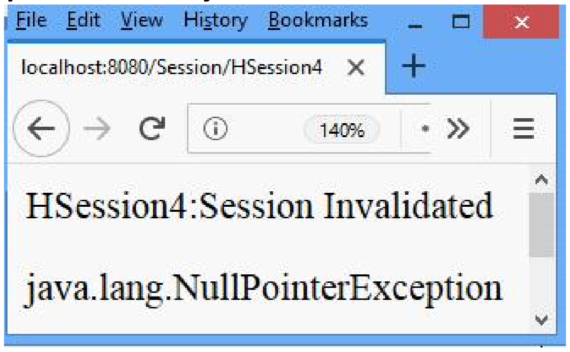

File Edit View History Bookmar
localhost:8080/Session/HSession3
140%
End Session

```
HSession3 :Hello again: java
```

HSession4.java

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class HSession4 extends HttpServlet
{   public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                           throws ServletException,IOException
{     response.setContentType("text/html");
      PrintWriter out=response.getWriter();
      HttpSession hs=request.getSession(false);
      try{
           String n=(String)hs.getAttribute("s_id");
      }
      catch(NullPointerException ne)
      {out.println("<p>HSession4:Session Invalidated
           </p>"+ne.toString());}
      out.println("<form action='/Session/httpsession.html'>");
      out.println("<p><input type='submit'
                            value='logout'></p></form>");
      }
}
```

Output:HSession4.java


File Edit View History Bookmarks
localhost:8080/Session/HSession4
140%

```
HSession4:Session Invalidated

java.lang.NullPointerException
```

Q22. What is filter? What is its use? List different filter interfaces with their important methods. List the applications of filter.

Ans. What is filter?

A filter is an object that is invoked at the preprocessing and post processing of a request. Java Servlet Filter is used to intercept request and do some pre-processing and can be used to intercept response and do post-processing before sending to client in web application.

What is its use?

- Recording all incoming requests
- Logs the IP addresses of the computers from which the requests originate
- Conversion
- Data compression
- Encryption and Decryption
- Input validation etc.

Filter interfaces

The javax.servlet package contains the three interfaces of Filter API.

1. **Filter**
   - For creating any filter, you must implement the Filter interface.
   - Filter interface provides the life cycle methods for a filter.

Method of Filter Interface

void **init**(FilterConfig config)	init() method is invoked only once. It is used to initialize the filter.
void **doFilter** (HttpServletRequest reuest, HttpServletResponse response, FilterChain chain)	doFilter() method is invoked every time when user request to any resource, to which the filter is mapped.It is used to perform filtering tasks.
void **destroy**()	This is invoked only once when filter is taken out of the service.

2. **FilterChain**

- The object of FilterChain is responsible to invoke the next filter or resource in the chain.
- This object is passed in the doFilter method of Filter interface.
- The FilterChain interface contains only one method:

void **doFilter** (HttpServletRequest request, HttpServletResponse response)	It passes the control to the next filter or resource.

3. FilterConfig

- FilterConfig is created by the web container.
- This object can be used to get the configuration information from the web.xml file.

Method

void init(FilterConfig config)	init() method is invoked only once it is used to initialize the filter.
String getInitParameter (String parameterName)	Returns the parameter value for the specified parameter name.

List the applications of filter

1. Authentication-Blocking requests based on user identity.
2. Logging and auditing-Tracking users of a web application.
3. Image conversion-Scaling maps, and so on.
4. Data compression-Making downloads smaller.
5. Localization-Targeting the request and response to a particular locale.

Advantage of Filter

- Filter is pluggable.
- One filter don't have dependency onto another resource.
- Less Maintenance Cost

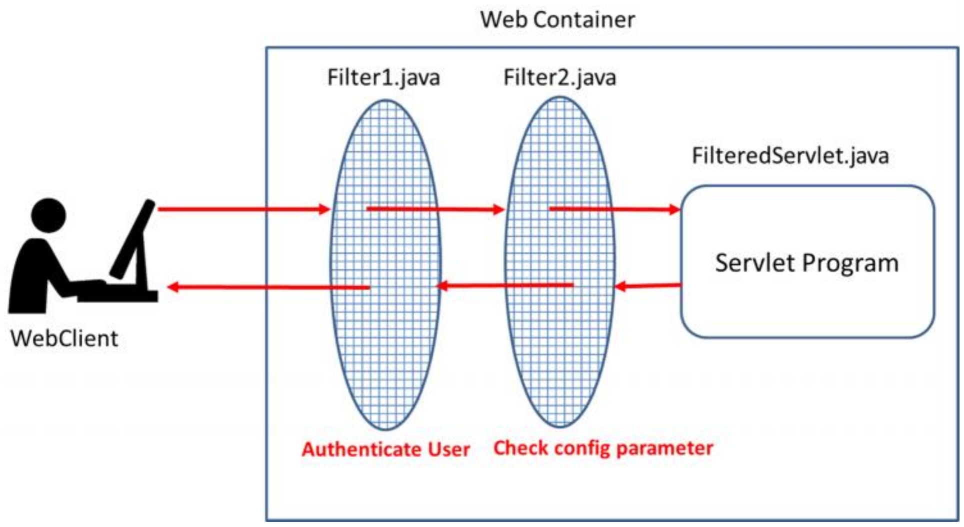
The **servlet filter is pluggable**, i.e. its entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet.
So maintenance cost will be less.

Q23. Write a Servlet Program that users two filters

i) Authentication Filter: Checks authentication value (userid/password)

ii) Config Param Filter: checks value of config param in web.xml

Ans.



Web.xml

```
<web-app>
<servlet>
   <servlet-name>FilteredServlet</servlet-name>
   <servlet-class>FilteredServlet</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>FilteredServlet</servlet-name>
   <url-pattern>/FilteredServlet</url-pattern>
</servlet-mapping>

     <filter>
             <filter-name>f1</filter-name>
             <filter-class>Filter1</filter-class>
     </filter>
     <filter-mapping>
             <filter-name>f1</filter-name>
             <url-pattern>/FilteredServlet</url-pattern>
     </filter-mapping>
```

```
    <filter>
            <filter-name>f2</filter-name>
            <filter-class>Filter2</filter-class>
            <init-param>
                <param-name>permit</param-name>
                <param-value>yes</param-value>
            </init-param>
    </filter>
    <filter-mapping>
            <filter-name>f2</filter-name>
            <url-pattern>/FilteredServlet</url-pattern>
      </filter-mapping>
    </web-app>
```

index.html

```
<html>
    <head>
        <title>filter</title>
    </head>
 <body>
  <form action="/Filter/FilteredServlet" >
     <p>Login ID:<input type="text"     name="login"></p>
     <p>Password:<input type="password" name="pwd"></p>
     <p><input type="submit" value="Sign In"></p>
  </form>
 </body>
</html>
```

Filter1.java

```
    import java.io.IOException;
    import java.io.PrintWriter;
    import javax.servlet.*;
    public class Filter1 implements Filter{
    public void init(FilterConfig config) {}
    public void doFilter(ServletRequest req,
                         ServletResponse resp,
                         FilterChain chain)
                         throws IOException, ServletException
    {
       PrintWriter out=resp.getWriter();
       out.print("<p>filter1 is invoked before</p>");
       if(req.getParameter("login").equals("java") &&
                   req.getParameter("pwd").equals("servlet"))
  {
    chain.doFilter(req, resp);//send request to next resource
  }//if
```

```
    else
      {out.print("<p>invalid login/password</p>");}//else

      out.print("<p>filter1 is invoked after</p>");
    }
    public void destroy() {}}
```

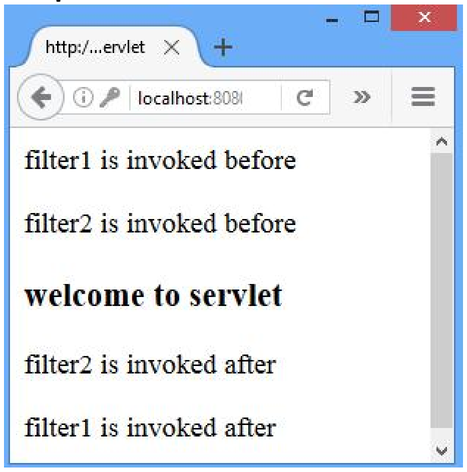
Filter2.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.*;
public class Filter2 implements Filter{
String permission;
public void init(FilterConfig config) throws ServletException
{           permission=config.getInitParameter("permit");
}
public void doFilter(ServletRequest req, ServletResponse resp,
                                                   FilterChain chain)
                           throws IOException, ServletException
{         PrintWriter out=resp.getWriter();
              out.print("<p>filter2 is invoked before</p>");
              if(permission.equals("yes"))
             { chain.doFilter(req, resp);}//if
          else
          { out.println("Permission Denied"); }//else
              out.print("<p>filter2 is invoked after</p>");
          }
          public void destroy() {}}
```

FilteredServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.*;
public class FilteredServlet extends HttpServlet
{
 public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
              throws ServletException, IOException
{
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<p><h3>welcome to servlet</h3></p>");
          }
    }
```

Output:



```
filter1 is invoked before

filter2 is invoked before

welcome to servlet

filter2 is invoked after

filter1 is invoked after
```

Q23. Explain Session Timeout

Ans. The session timeout in a web application can be configured in two ways

1. Timeout in the deployment descriptor (web.xml)
2. Timeout with setMaxInactiveInterval()

Timeout in the deployment descriptor (web.xml)

```
<web-app>
   <session-config>
     <session-timeout> 10 </session-timeout>
   </session-config>
</web-app>
```

Note that the value of the timeout is set in minutes, not in seconds.

Timeout with setMaxInactiveInterval()

The timeout of **the current session only** can be specified programmatically via the API of the *javax.servlet.http.HttpSession*

```
HttpSession session = request.getSession();
session.setMaxInactiveInterval(10*60);//time specified in seconds
```

Q24. State and explain Types of Servlet Events

Ans.

- Events are basically occurrence of something.
- Changing the state of an object is known as an event.
- There are many Event classes and Listener interfaces in the javax.servlet and javax.servlet.http packages.
- In web application world an event can be
  1. Initialization of application
  2. Destroying an application
  3. Request from client
  4. Creating/destroying a session
  5. Attribute modification in session etc.

Event classes

ServletRequestEvent	Events of this kind indicate lifecycle events for a ServletRequest. The source of the event is the ServletContext of this web application.
ServletContextEvent	This is the event class for notifications about changes to the servlet context of a web application.
ServletRequestAttributeEvent	This is the event class for notifications of changes to the attributes of the servlet request in an application.
ServletContextAttributeEvent	Event class for notifications about changes to the attributes of the ServletContext of a web application.
HttpSessionEvent	This is the class representing event notifications for changes to sessions within a web application.
HttpSessionBindingEvent	Send to an Object that implements HttpSessionBindingListener when bound into a session or unbound from a session.

Q25. Write a Servlet program which retrieves record from database

Ans.

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class JDBCServlet extends HttpServlet
{
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
                        throws ServletException,IOException
    {   response.setContentType("text/html");
        PrintWriter out=response.getWriter();
     try{
        Class.forName("com.mysql.jdbc.Driver");
        Connection con=DriverManager.getConnection
          ("jdbc:mysql://localhost:3306/ajava","root","DIET");
        Statement st=con.createStatement();
        ResultSet rs=st.executeQuery("select * from cxcy");
        while(rs.next())
        { out.println("<p>"+rs.getInt(1));
          out.println(rs.getString(2));
          out.println(rs.getString(3)+"</p>");
         }
     }catch(Exception e)
     {out.println("<p>inside exception"+e.toString()+"</p>");}
```

```
        }//doGet()
       }//Class
```

Servlet Interview Questions

Q1. Servlet is Java class. Then why there is no constructor in Servlet? Justify your answer.

Ans.

- Servlet object has a well-defined life cycle where creation and initialization steps are distinct.
- Servlets are not directly instantiated by Java code, instead container create there instance and keep them in pool.
- Servlet implementation classes can have constructor but they should be using init() method to initialize Servlet because of two reasons, first you cannot declare constructors on interface in Java, which means you cannot enforce this requirement to any class which implements Servlet interface and second, Servlet require ServletConfig object for initialization which is created by container

Q2. Can we write the constructor in Servlet?

Ans.

- Servlet is a java class, whose object will be created by servlet container, and then container will call 3 life cycle method on that object.
- Container creates our servlet class object using 0-args constructor, If u want to place your own constructor you can, but make sure 0-args constructor is always present.
- However, our parameterized constructor will not be used by servlet container, and will be of no use.
- Also If u declare your own 0-args constructor, make sure it is public, so that it could be easily accessible to container

Q3. What happens if you add a main method to servlet?

Ans.

- We can place but it never executes automatically, because it is not a life cycle method.
- We can see main() method acting as helper method(normal method) to life cycle methods. We can call this main() from any lifecycle method explicitly as a helper method.
- Main method is for standalone apps,not for servlets which are executed by life cycle methods.

Q4. Consider a scenario in which 4 users are accessing a servlet instance. Among which one user called destroy() method. What happens to the rest 3 users?

Ans. We know that by default servlet is multithreaded, for every client request a new thread will be created and assigned to that to perform the service. so if one thread initiates destroy() only itself will be terminated but other threads not terminates.

Q5. How does the JVM execute a servlet compared with a regular Java class?

Ans.

- Servlets are standard Java classes and are executed by the Java Virtual Machine in exactly the same way as any other. However, the environment or context in which Servlets are executed is different. A Servlet is not invoked directly through a main method, the class is loaded and run by a Servlet Container.
- A servlet is not invoked directly using a main() method like any other class.
- The servlet class is invoked and executed by a web container (Like Apache Tomcat).
- The container reads the configuration (like web.xml), identifies the servlet class, and uses java class loader system to load and run the servlets.

Q6. How to get the fully qualified name of the client that sent the request in servlet?

Ans. A servlet can use getRemoteAddr() and getRemoteHost() to retrieve the client's IP Address and client's host name from a http requisition.

Q7. Why GenericServlet is an abstract class ?

Ans.

- GenericServlet class is abstract because there is a method called service() which is public abstract void. service() must be override, service() method defines what type of protocol is used for request.
- Another thing is that according to Java specification those classes have abstract methods must declared abstract.
- Abstract methods have no body only prototype.

Q8. Who is responsible to create the object of servlet?

Ans. The web container or servlet container.

Q9. What is difference between Cookies and HttpSession?

Ans. Cookie works at client side whereas HttpSession works at server side.

Q10. Can filtering be done in an ordered way? If so then how to achieve it?

Ans. Yes. The order of filter-mapping elements in web.xml determines the order in which the web container applies the filter to the servlet. To reverse the order of the filter, you just need to reverse the filter-mapping elements in the web.xml file.

GTU Questions

1.	Write a Login servlet. Take input username and password from html file login.html and authenticate the user. Write the web.xml.[7]	Win’17
2.	List and Explain various stages of Servlet life cycle. Explain role of web container.[7]	Win’17 Sum’16
3.	What is Session? Explain various Session tracking mechanisms in servlet with example.[7]	Win’17 Sum’16 Win’16
4.	What is filter? What is its use? List different filter interfaces with their important methods[7]	Win’17 Win’16
5.	Explain use of ServletConfig and ServletContext object with example.[4]	Win’17 Sum’17
6.	Discuss the use of GET and POST with example.[4]	Win’17 Sum’16
7.	Write a servlet RegistrationServlet to get the values from registration.html html page and display the contents. Write the web.xml file.[7]	Win’17
8.	Design a form to input details of an employee and submit the data to a servlet. Write code for servlet that will save the entered details as a new record in database table Employee with fields (EmpId, EName, Email, Age).[7]	Sum’17
9.	Explain Request and Response object in Servlet.[7]	Win’16
10.	Write servlet which displayed following information of client.[7] I. Client Browser II. Client IP address III. Client Port No IV. Server Port No V. Local Port No VI. Method used by client for form submission VII. Query String name and values	Sum’16
11.	Write small web application which takes marks of three subject and pass to servlet. Servlet forward to model class having method getClass() and getPercentage(). Display class and percentage in .jsp page.[7]	Sum’16
12.	i. GenericServlet vs HttpServlet ii. doGet() vs doPost() iii. Servlets vs CGI iv. Session and Cookie [7]	Sum’16