

102010505

Advanced Java

# Unit-3 (Part-II)

## Servlet API and Overview



# Cookies and Session Management

# Session Management in Servlets

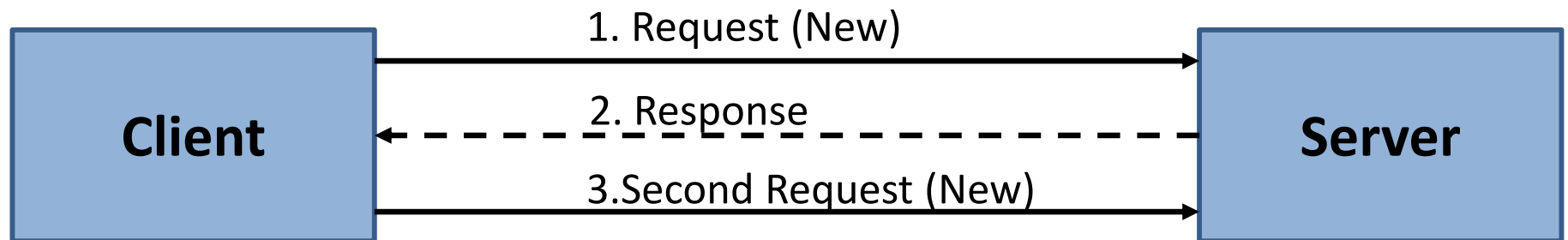
## What is Session?

*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.*

# Session Management in Servlets

## 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.



# Session Management

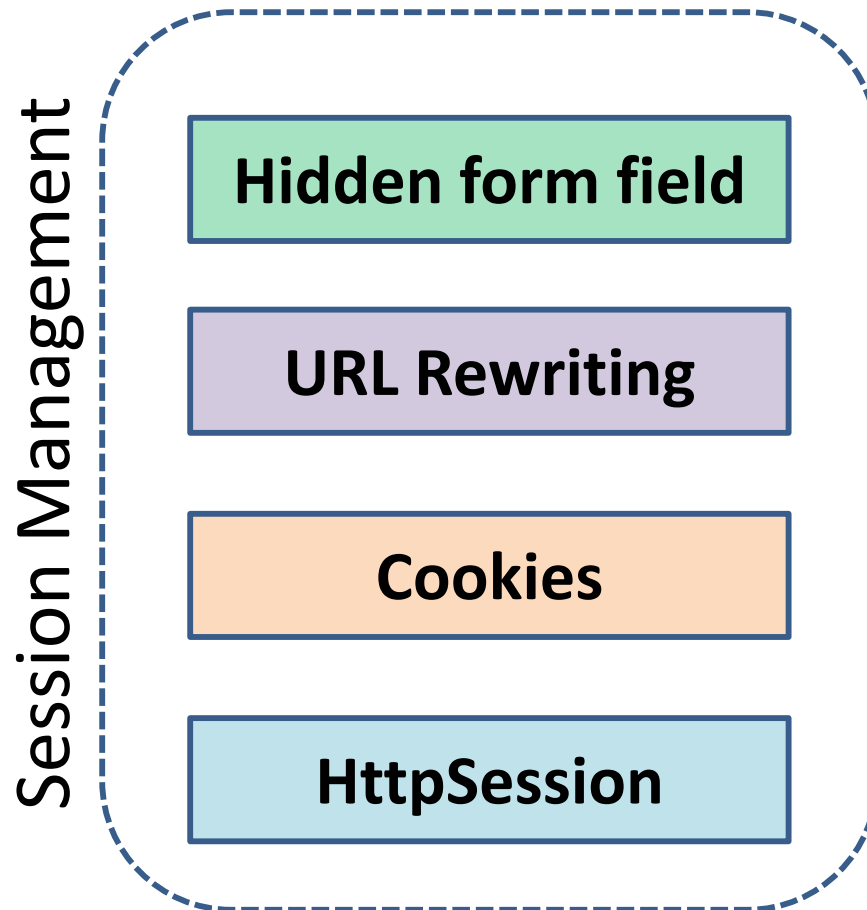
## Example: Application of Session

When a User logs into your website, no matter on which web page he visits after ***logging in***, his credentials will be with the server, until user ***logs out***.

So this is managed by creating a session.

# Session Management

- **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.



# Session Management: Hidden form field

- 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.

Example

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

# Session Management: Hidden form field

login.html

Name:

Password:

Session\_ID:

Valid.java

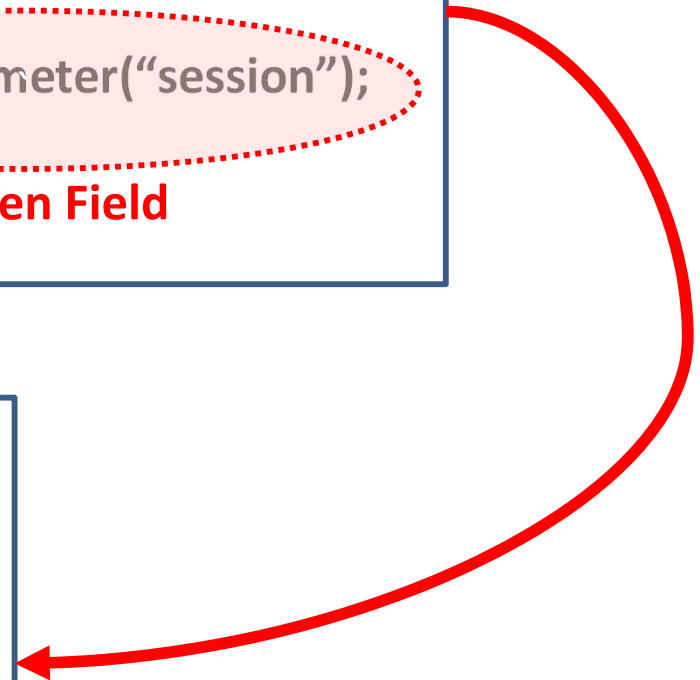
```
request.getParameter("name");  
request.getParameter("password");
```

```
request.getParameter("session");
```

**Hidden Field**

Welcome.java

```
request.getParameter("session");
```

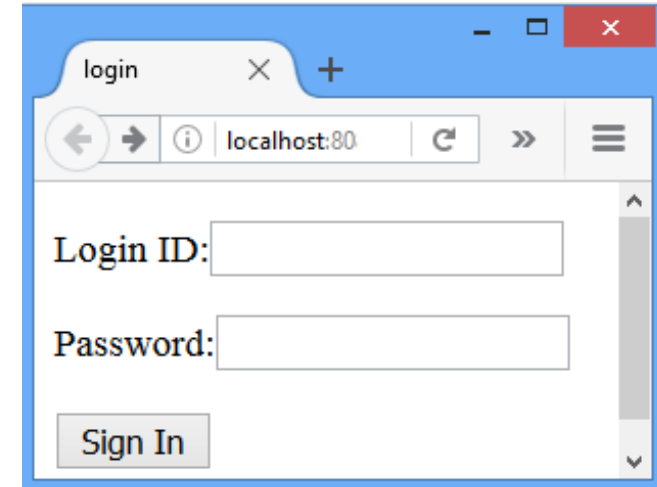




# Session Management: Hidden form field

*login.html*

```
1. <html>
2.     <head>
3.         <title>login</title>
4.     </head>
5. <body>
6.     <form action="/Session/Valid" method="POST">
7.         <p>Login ID:<input type="text" name="login"></p>
8.         <p>Password:<input type="text" name="pwd"></p>
9.         <p><input type="hidden" name="session_id"
10.             value="054"></p>
11.         <p><input type="submit" value="Sign In"></p>
12.     </form>
13.</body>
14.</html>
```



# Session Management: Hidden form field

Valid.java

```
1. public class Valid extends HttpServlet
2. {    public void doPost(HttpServletRequest request,
           HttpServletResponse response)
3.           throws ServletException, IOException
4. {
5.     response.setContentType("text/html");
6.     PrintWriter out=response.getWriter();
7.     RequestDispatcher rd;
8.     String login=request.getParameter("login");
9.     String pwd=request.getParameter("pwd");
10.    String session=request.getParameter("session_id");
```

Hidden  
Field

# Session Management: Hidden form field

Valid.java

```
11. if (login.equals("java") && pwd.equals("servlet"))
12. {
13.     rd=request.getRequestDispatcher("Welcome");
14.     rd.forward(request, response);
15. } //if
16. else
17. {
18.     out.println("<p><h1>Incorrect LoginId/Password
                                </h1></p>");
19.     rd=request.getRequestDispatcher("/login.html");
20.     rd.include(request, response);
21. } //else
22. } }
```

# Session Management: Hidden form field

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>");
    }
}
```

# Session Management: Hidden form field

## 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.

# Session Management: Hidden form field

## **Advantage of Hidden Form Field**

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

# Session Management: Hidden form field

## 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

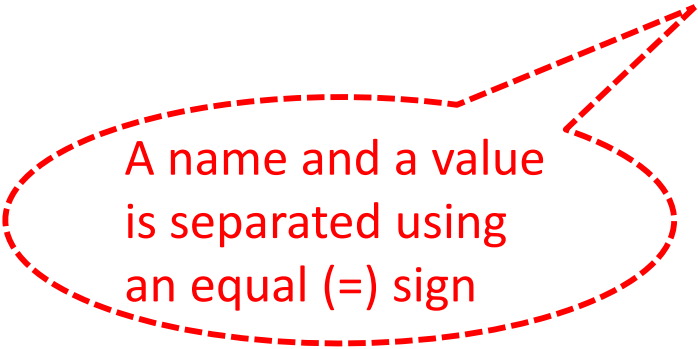
# URL Rewriting



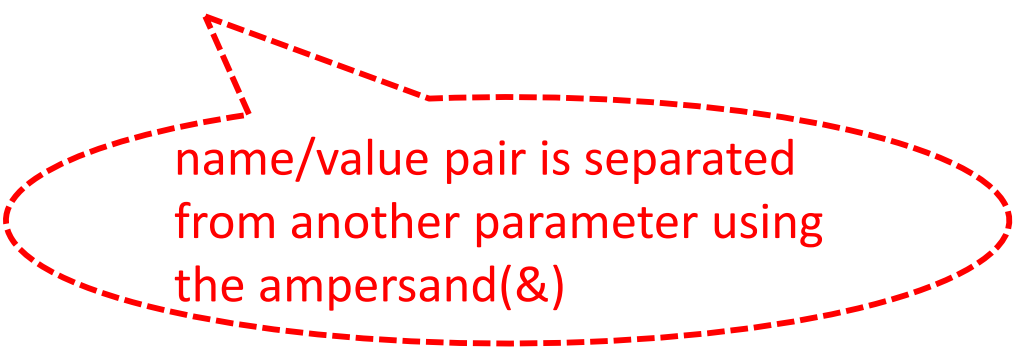
# Session Management: URL Rewriting

- 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 &...**



A name and a value  
is separated using  
an equal (=) sign



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.

# Session Management: URL Rewriting

Url1.java

```
1. import javax.servlet.*;
2. import javax.servlet.http.*;
3. import java.io.*;
4. public class Url1 extends HttpServlet
5. {    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
6.        throws ServletException, IOException
7. {    String url;
8.        response.setContentType("text/html");
9.        PrintWriter out=response.getWriter();
10.        //for URL rewriting
11.        url= "http://localhost:8080/Session  

               /Url2?s_id1=054&s_id2=055";
12.        out.println("<a href="+url+">next page</a>");
13. } }
```



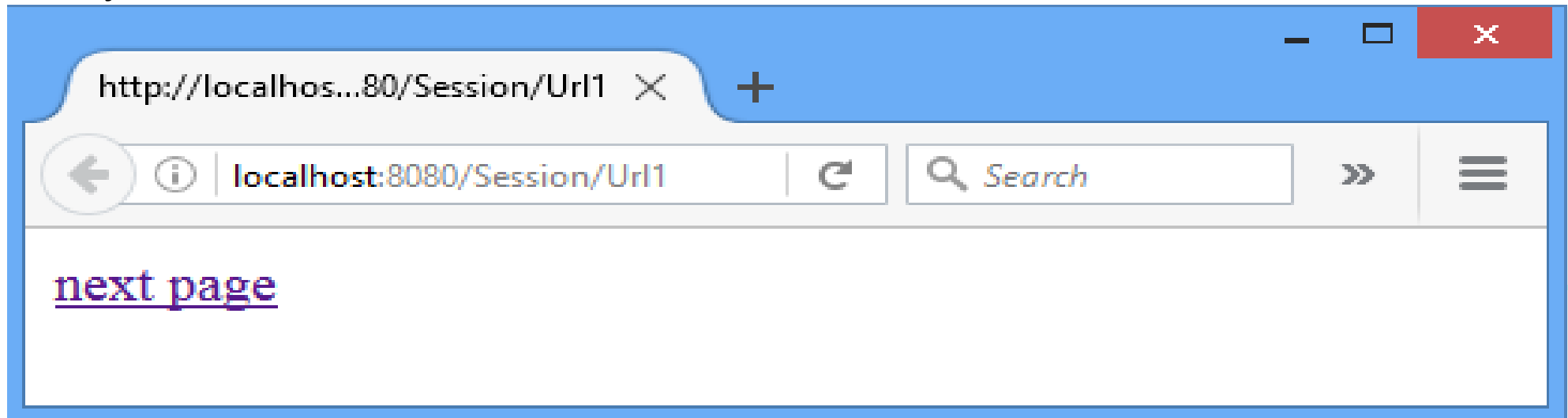
# Session Management: URL Rewriting

Url2.java

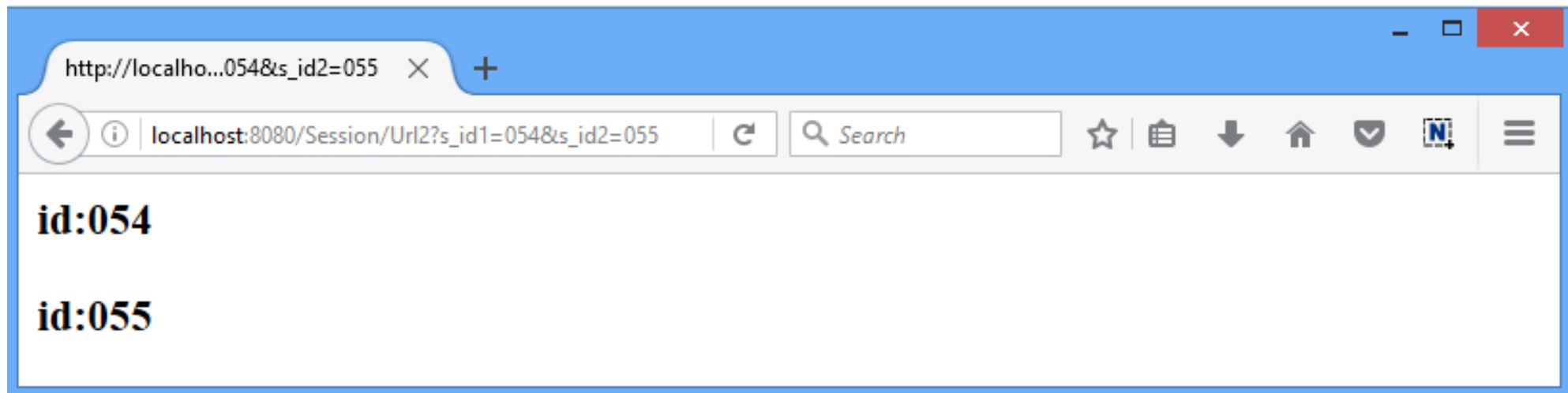
```
1. import javax.servlet.*;
2. import javax.servlet.http.*;
3. import java.io.*;
4. public class Url2 extends HttpServlet
5. {    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
6.        throws ServletException, IOException
7.    {    response.setContentType("text/html");
8.        PrintWriter out=response.getWriter();
9.        String session1=request.getParameter("s_id1");
10.        String session2=request.getParameter("s_id2");
11.        out.println("<h3>"+ "id: "+session1+"</h3>");
12.        out.println("<h3>"+ "id: "+session2+"</h3>");
13.    }
14. }
```

# Session Management: URL Rewriting

Url1.java



Url2.java



# Session Management: URL Rewriting

## **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 '?'.

# Cookies

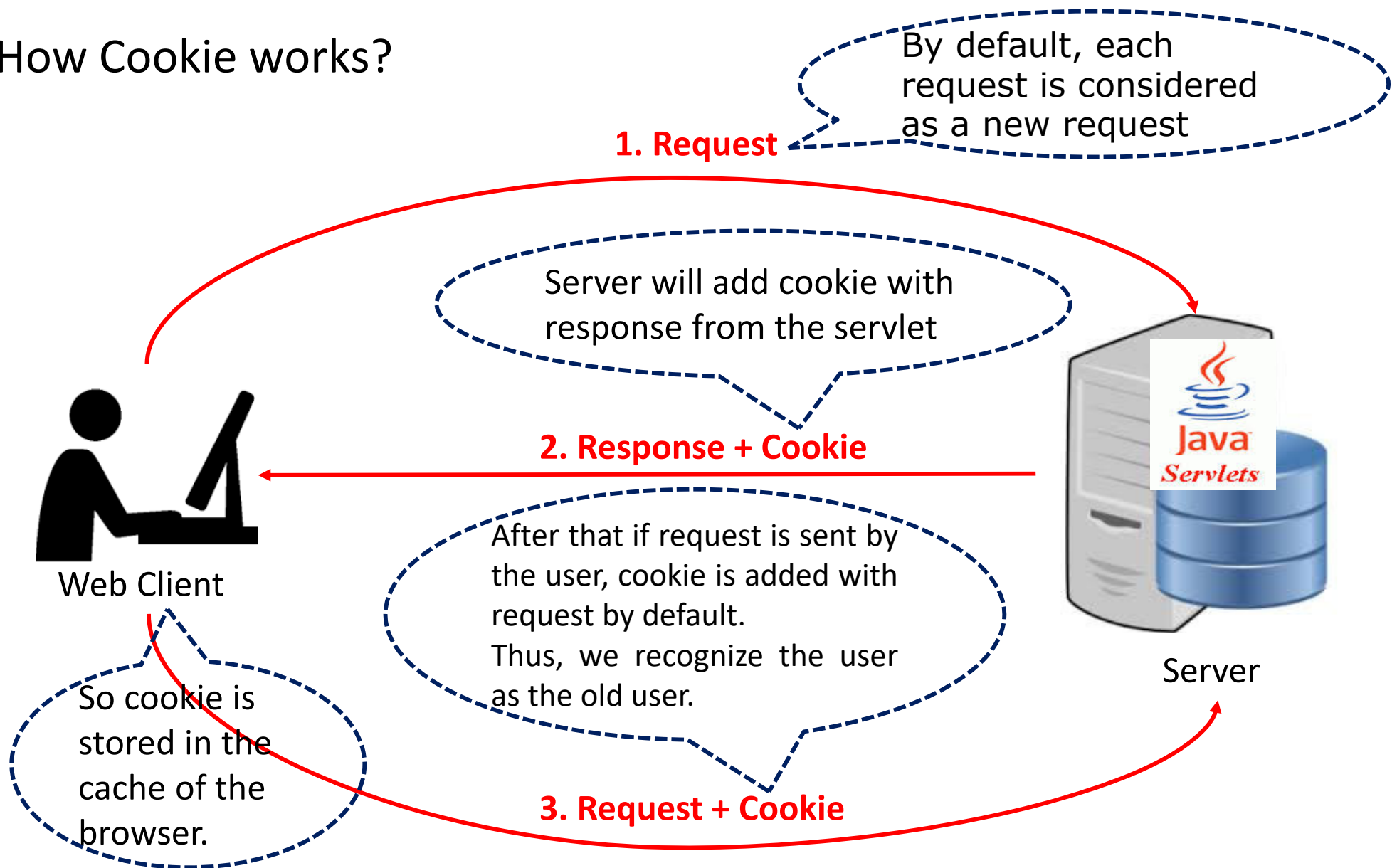
`javax.servlet.http.Cookie`

# Session Management: Cookies

- 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

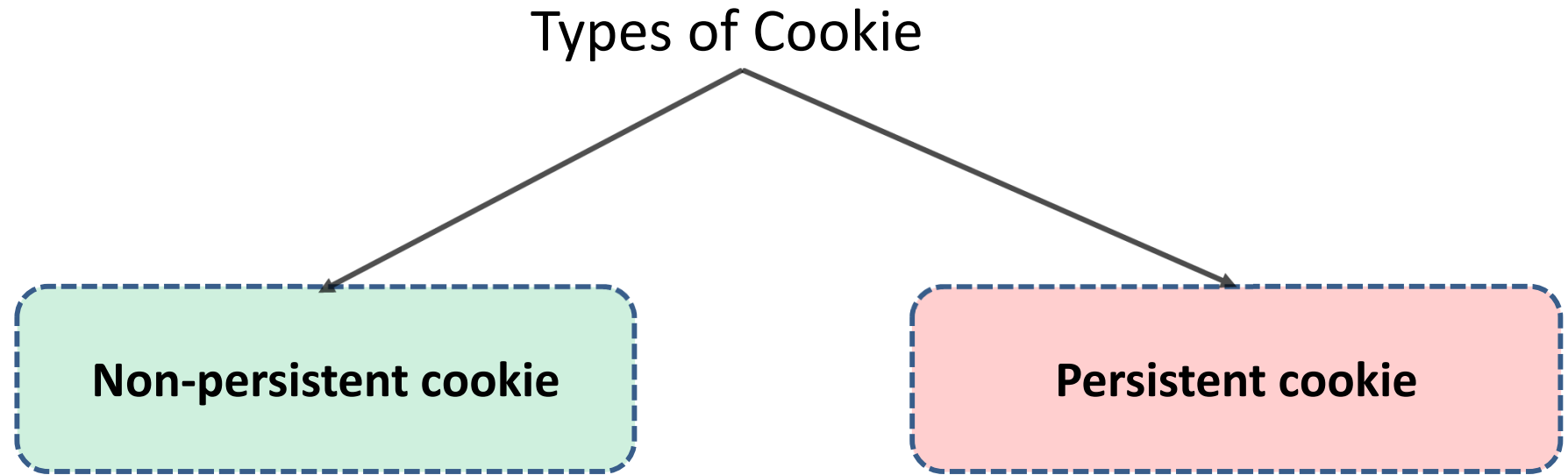
# Session Management: Cookies

## How Cookie works?





# Session Management: Cookies



- It is **valid for single session** only.
- It is removed each time when user closes the browser.

- 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.

# Session Management: Cookies

## Cookie class

- **javax.servlet.http.Cookie**

This class provides the functionality of using cookies.

It provides a lots of useful methods for cookies.

## ***Constructor***

|   |  |
|---|--|
| <b>Cookie</b> (String name, String value) | constructs a cookie with a specified name and value. |
|---|--|

## ***Example***

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

# Session Management: Cookies

## Methods of Cookie class

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

# Session Management: Cookies

## Other Methods of HttpServletRequest & HttpServletResponse

|                                       |  |
|---------------------------------------|--|
| void <b>addCookie</b> (Cookie cookie) | Method of HttpServletResponse interface is used to add cookie in response object.  |
| Cookie[] <b>getCookies</b> ()         | Returns an array containing all of the Cookie objects the client sent with this request. This method returns null if no cookies were sent. |

# Session Management: Cookies

How to create Cookie?

*Example*

//creating cookie object

```
Cookie c= new Cookie("session_id", "054");
```

//adding cookie in the response

```
response.addCookie(c);
```

# Session Management: Cookies

## 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  
}
```

# Session Management: Cookies

## 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.

# Session Management: Cookies

## How to delete Cookie?

```
//deleting value of cookie
```

```
Cookie c = new Cookie("user", "");
```

```
//changing the maximum age to 0 seconds
```

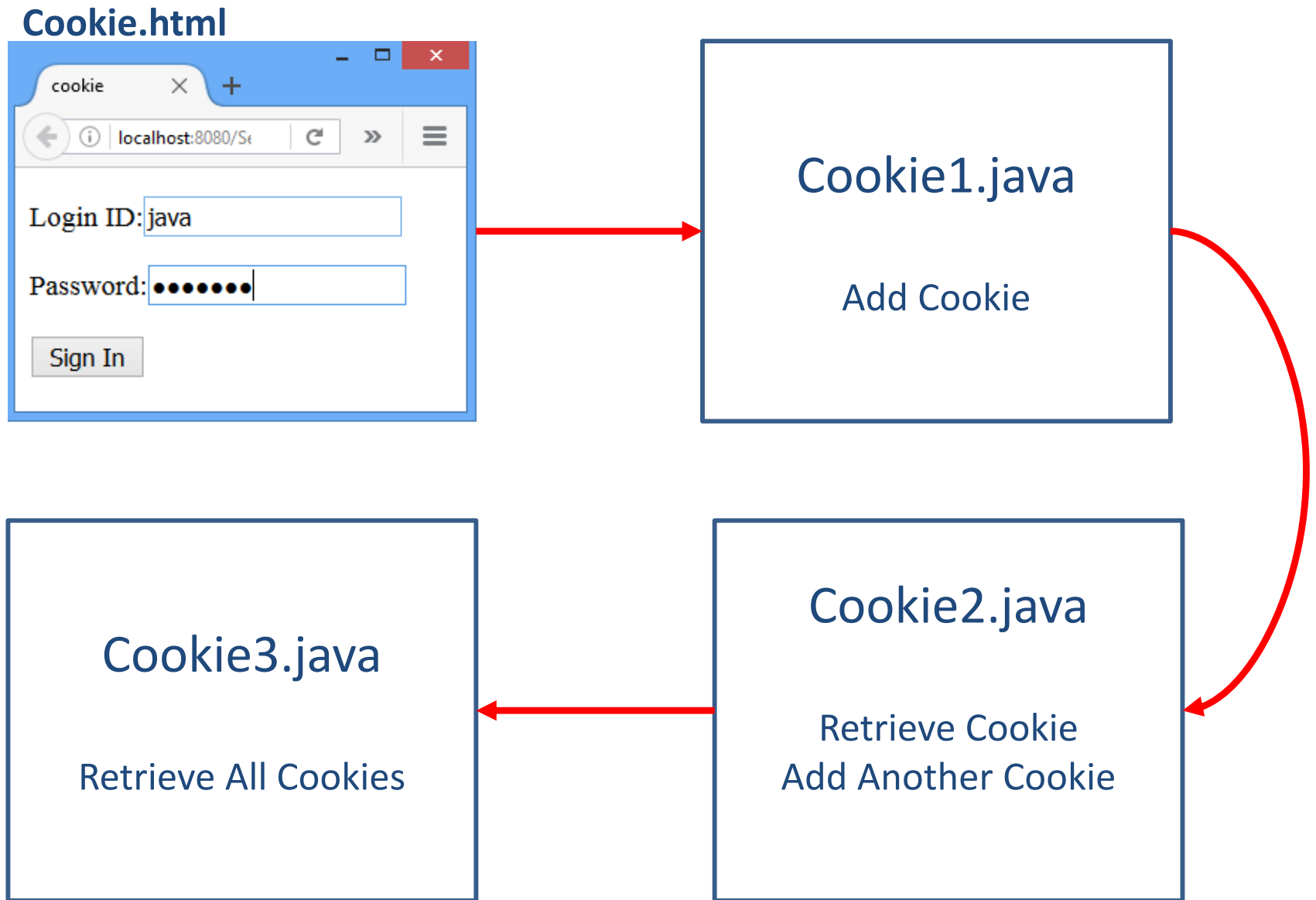
```
c.setMaxAge(0);
```

```
//adding cookie in the response
```

```
response.addCookie(c);
```



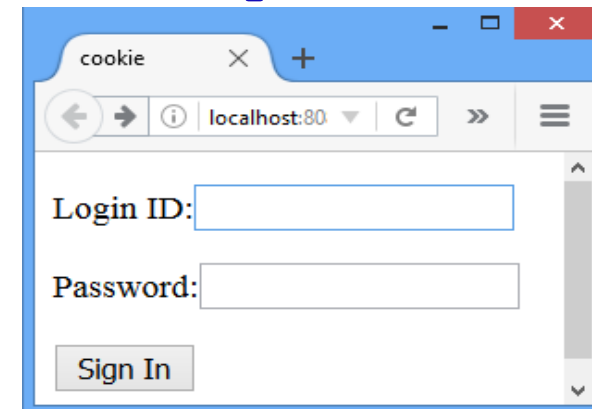
# Session Management: Cookies



# Session Management: Cookies

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>
```



cookie

localhost:80

Login ID:

Password:

Sign In

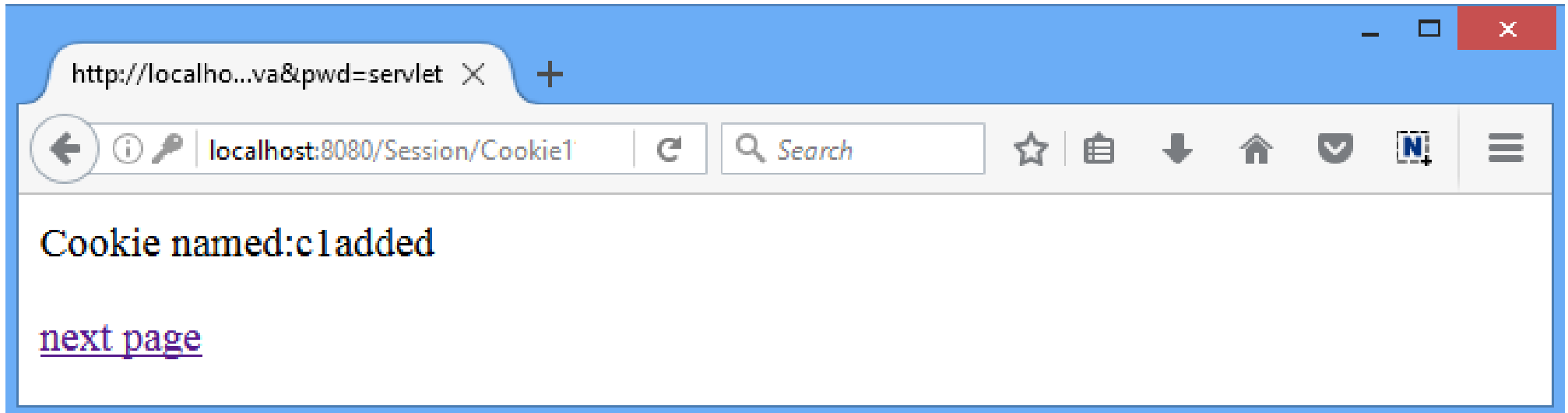
# Session Management: Cookies

Cookie1.java

```
1.  public class Cookie1 extends HttpServlet
2.  {    public void doGet(HttpServletRequest request,
           HttpServletResponse response)
3.           throws ServletException, IOException
4.  {    response.setContentType("text/html");
5.      PrintWriter out=response.getWriter();
6.      String login=request.getParameter("login");
7.      String pwd=request.getParameter("pwd");
8.      if(login.equals("java") && pwd.equals("servlet"))
9.      {    Cookie c = new Cookie("c1",login); //create cookie
10.         response.addCookie(c); //adds cookie with response
11.         out.println("Cookie named:"+c.getName()+" added");
12.         String path="/Session/Cookie2";
13.         out.println("<p><a href="+path+">next page</a></p>");
14.     }
15.     else {    //Redirect page to cookie.html}
16. } }
```

# Session Management: Cookies

Output: Cookie1.java [add Cookie]



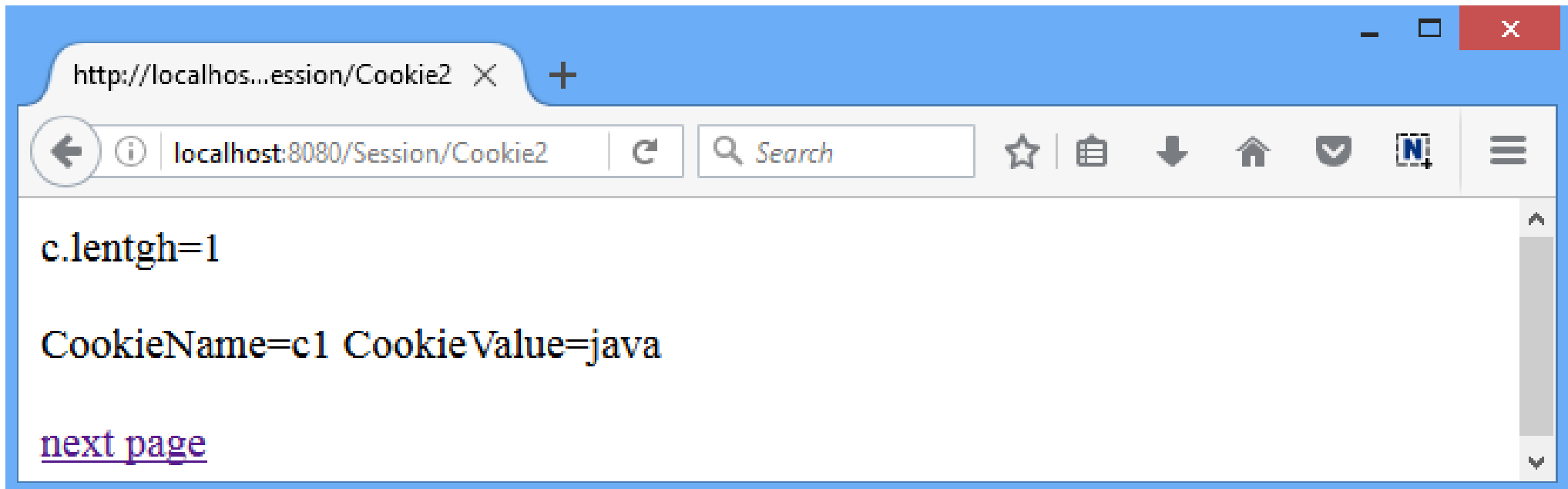
# Session Management: Cookies

Cookie2.java

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

# Session Management: Cookies

**Output: Cookie1.java** [Retrive Cookie and add one more cookie]



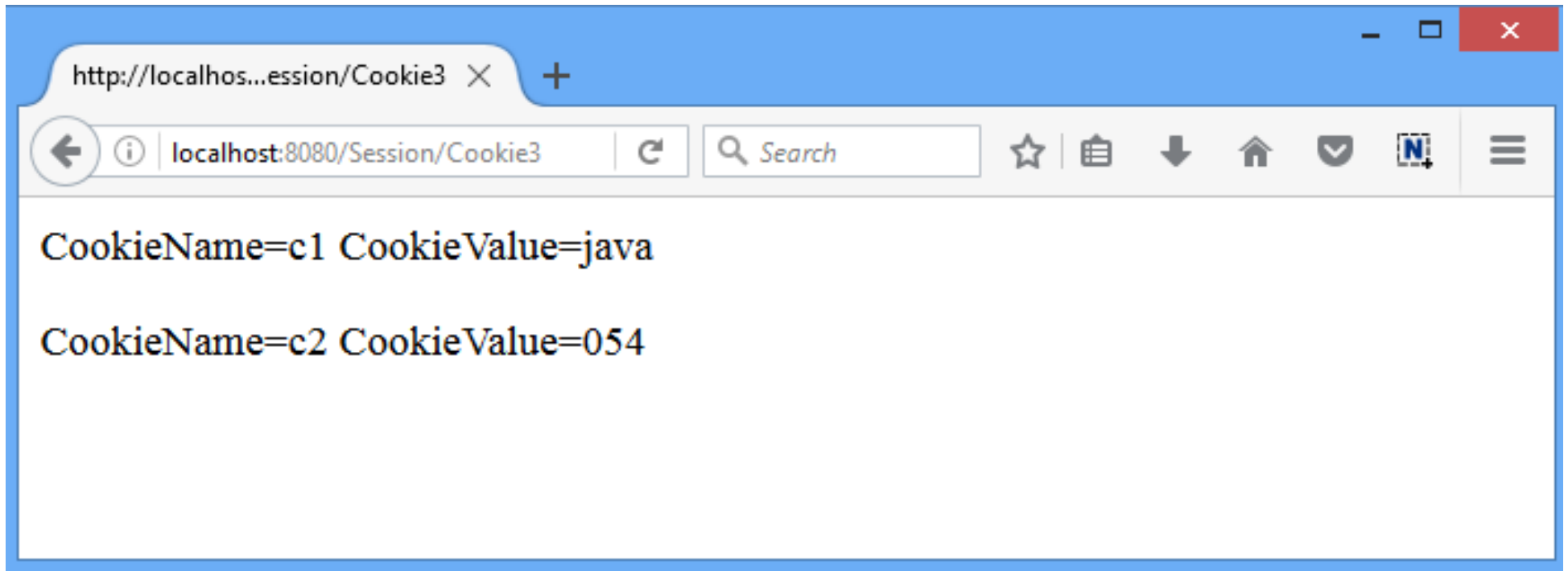
# Session Management: Cookies

Cookie3.java

```
1. public class Cookie3 extends HttpServlet
2. {    public void doGet(HttpServletRequest request,
           HttpServletResponse response)
3.           throws ServletException, IOException
4.     {    response.setContentType("text/html");
5.         PrintWriter out=response.getWriter();
6.         Cookie c[]=request.getCookies();
7.         for(int i=0;i<c.length;i++)
8.         {    out.println("<p>");
9.             out.println("CookieName="+c[i].getName()+
10.                          "CookieValue="+c[i].getValue());
11.             out.println("</p>");
12.         }
13.     }
14. }
```

# Session Management: Cookies

**Output: Cookie1.java** [Retrive all the Cookies]





# Session Management: Cookies

## **Advantage of Cookies**

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

## **Disadvantage of Cookies**

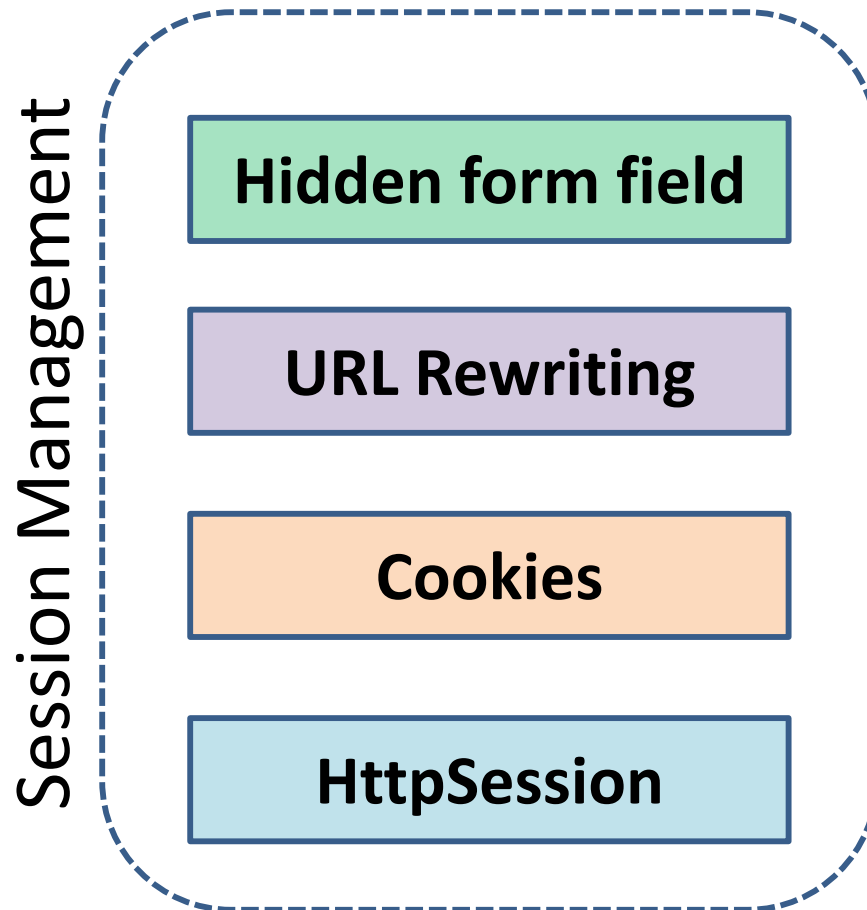
- It will not work if cookie is disabled from the browser.
- Only textual information can be set in Cookie object.

# Interview Questions: Cookies

|    |   |
|----|---|
| 1. | What Are Cookies?                                   |
| 2. | Can I see/view the cookies I have on my computer?   |
| 3. | What's in a Cookie? OR What does a Cookie contains? |
| 4. | Why are Cookies Used?                               |
| 5. | How Long Does a Cookie Last?                        |
| 6. | How Secure are Cookies?                             |
| 7. | What are Tracking Cookies?                          |
| 8. | What do the Cookies Do?                             |

# Session Management

- **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.



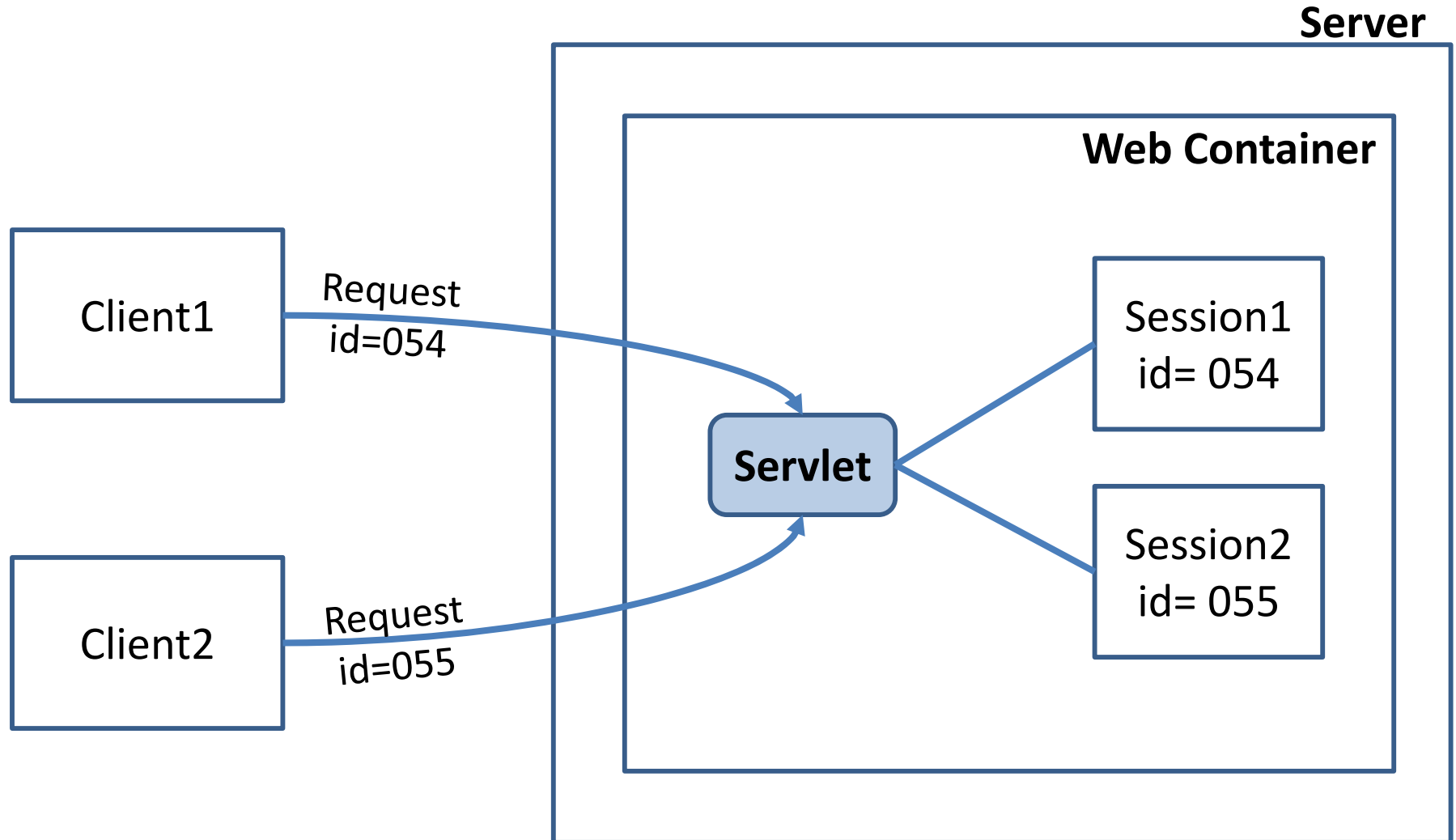
# HttpSession

`javax.servlet.http.HttpSession`

# Session Management: HttpSession


- Apart from the above mentioned three ways, servlet provides HttpSession Interface which 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.

# Session Management : HttpSession



**Working of HttpSession**

# Session Management :HttpSession

- Package: javax.servlet.http.**HttpSession**  
**Interface**
- 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.

# Session Management : HttpSession

The HttpServletRequest interface provides two methods to get the object of HttpSession

|  |  |
|--|--|
| HttpSession <b>getSession()</b>                  | Returns the current session associated with this request, or if the request does not have a session, creates one.                          |
| HttpSession<br><b>getSession(boolean create)</b> | Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session. |



# Session Management : HttpSession

## Methods of HttpSession interface

|                                   |  |
|-----------------------------------|--|
| String <b>getId()</b>             | Returns a string containing the unique identifier value.   |
| long <b>getCreationTime()</b>     | Returns the time when this session was created, measured in milliseconds.                                    |
| long <b>getLastAccessedTime()</b> | Returns the last time the client sent a request associated with this session, as the number of milliseconds. |
| void <b>invalidate()</b>          | Invalidates this session then unbinds any objects bound to it.   |

# Session Management : HttpSession

*How to create the session?*

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

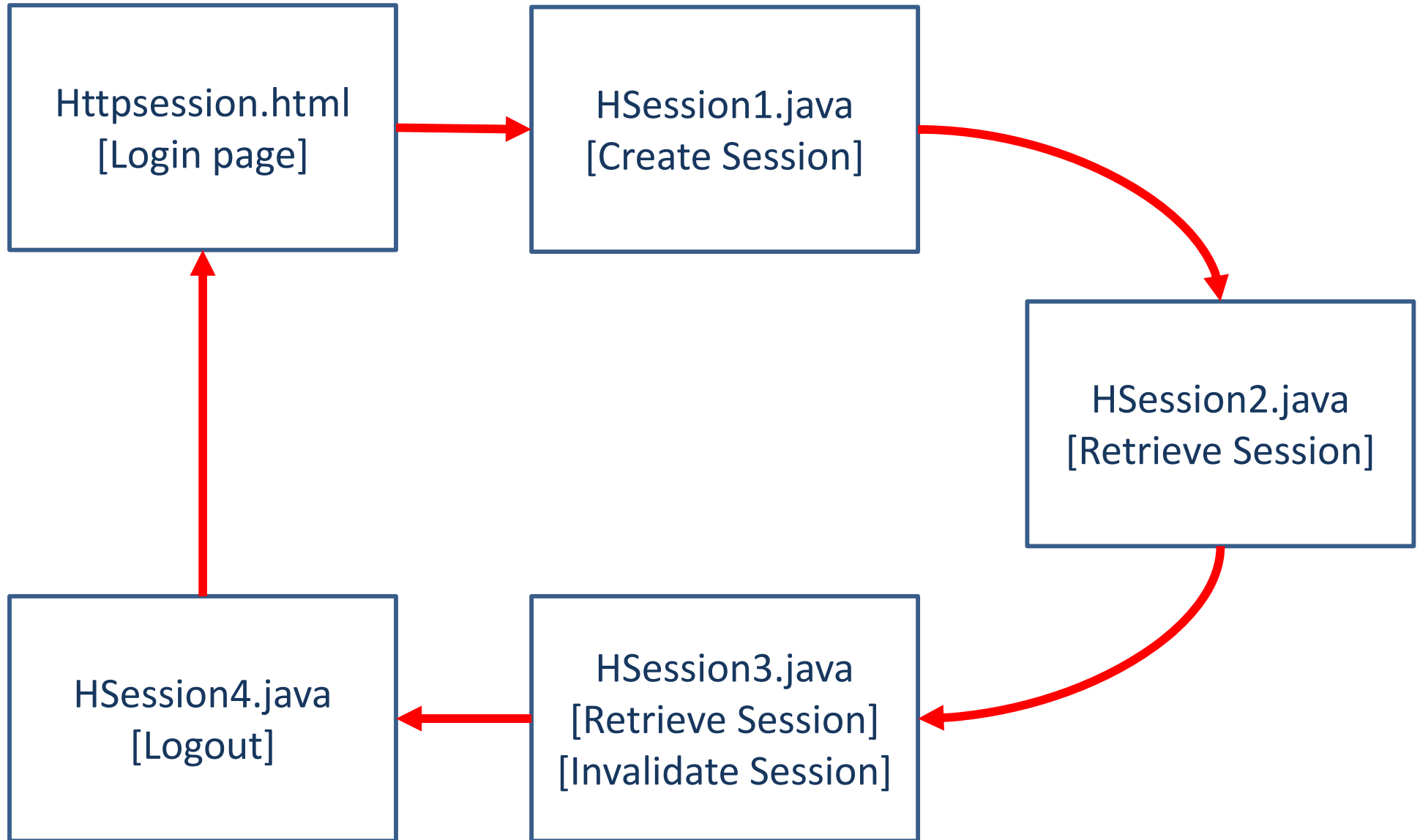
*How to retrieve a session?*

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

*How to invalidate a session?*

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

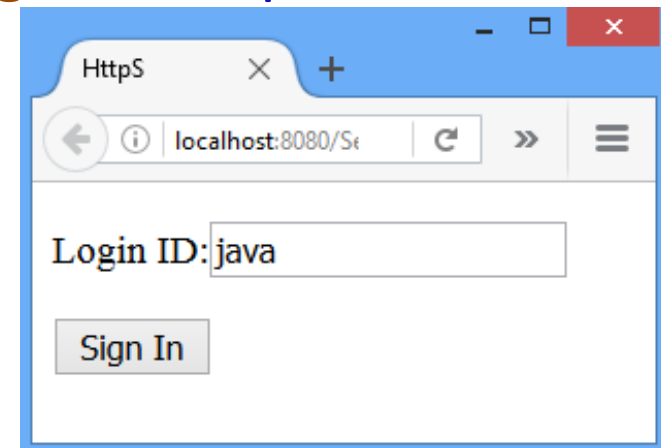
# Session Management : HttpSession



# Session Management : HttpSession

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>
```



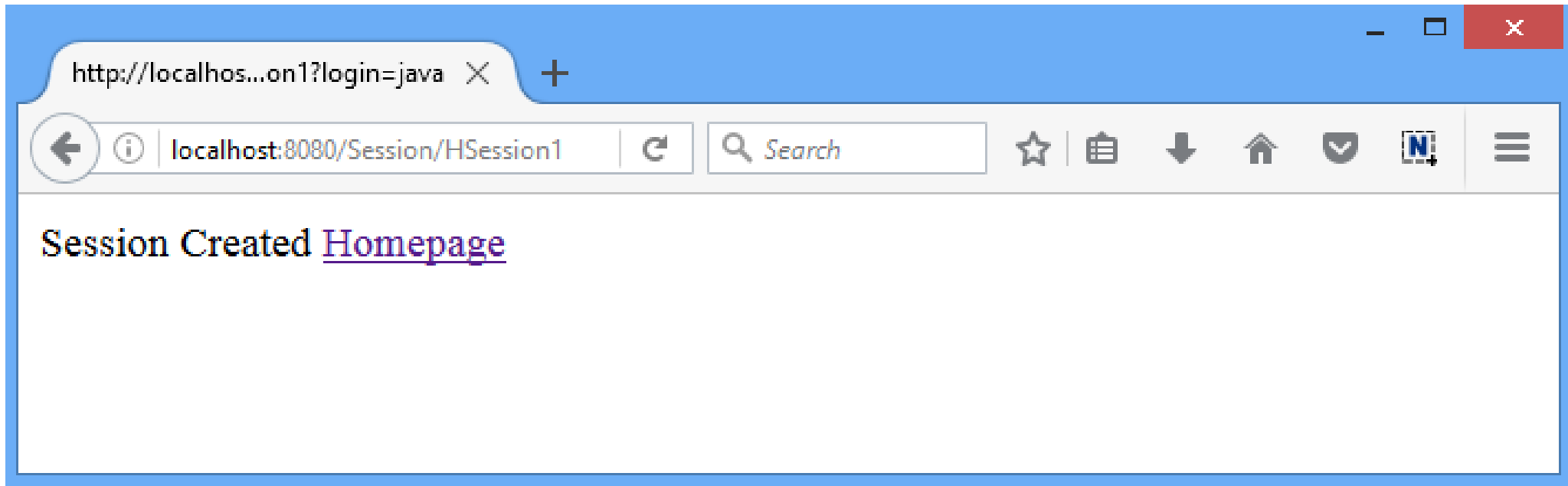
# Session Management : HttpSession

HSession1.java

```
1. response.setContentType("text/html");
2. PrintWriter out=response.getWriter();
3. RequestDispatcher rd;
4. String login=request.getParameter("login");
5. if(login.equals("java") )
6. {    HttpSession hs=request.getSession();
7.      hs.setAttribute("s_id",login);//set HttpSession
8.      out.println("Session Created");
9.      out.print("<a href='HSession2'>Homepage</a>");
10. }
11. else
12. {    out.println("<p><h1>Incorrect Login Id/Password
                                     </h1></p>");
13.      rd=request.getRequestDispatcher("/httpsession.html");
14.      rd.include(request, response);}
```

# Session Management : HttpSession

Output: HttpSession1.java

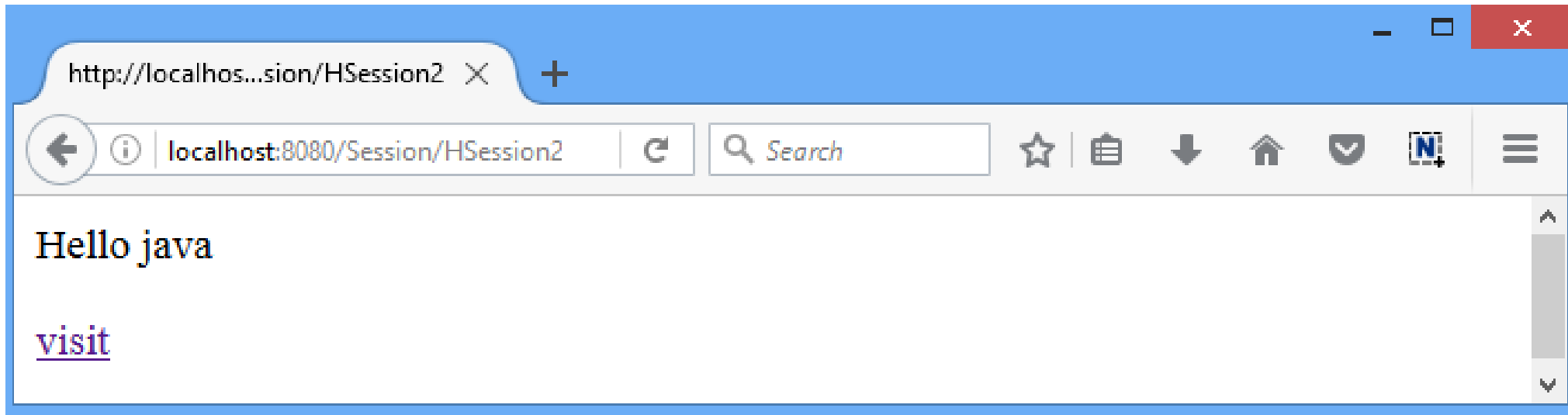


# Session Management : HttpSession

```
1. public class HSession2 extends HttpServlet HSession2.java
2. {    public void doGet(HttpServletRequest request,
           HttpServletResponse response)
3.           throws ServletException, IOException
4.     {response.setContentType("text/html");
5.       PrintWriter out=response.getWriter();
6.       HttpSession hs=request.getSession(false);
7.       String n=(String)hs.getAttribute("s_id");
8.       out.print("Hello "+n);
9.       out.print("<p><a href='HSession3'>visit</a></p>");
10. } }
```

# Session Management : HttpSession

Output: HttpSession2.java



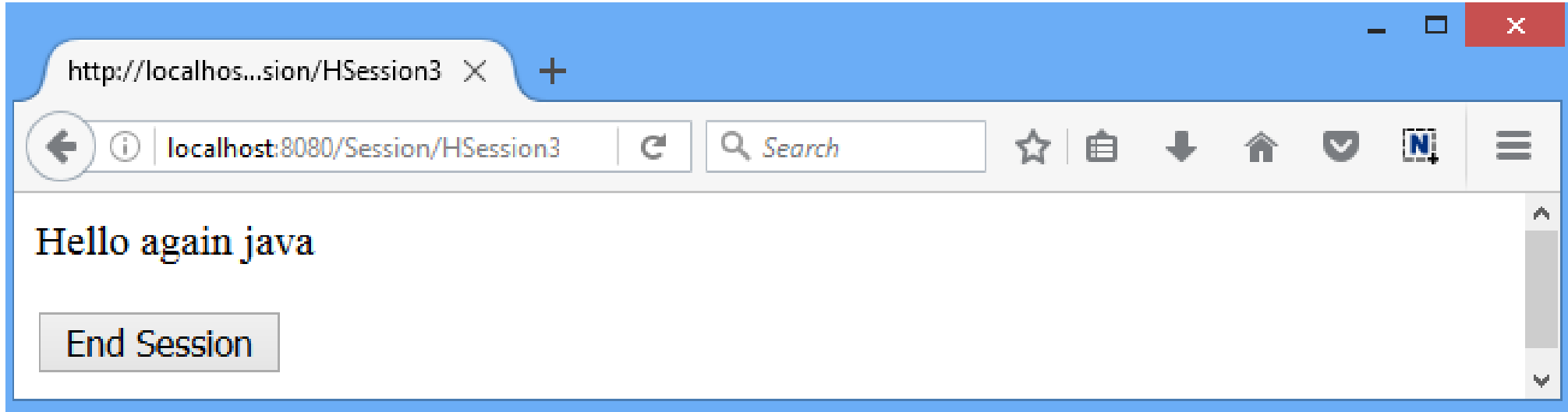


# Session Management : HttpSession

```
1. public class HSession3 extends HttpServlet HSession3.java
2. {     public void doGet(HttpServletRequest request,
               HttpServletResponse response)
3.         throws ServletException, IOException
4.     {response.setContentType("text/html");
5.       PrintWriter out=response.getWriter();
6.       HttpSession hs=request.getSession(false);
7.       String n=(String)hs.getAttribute("s_id");
8.       out.print("Hello again "+n);
9.       out.println("<form action='/Session/HSession4'>");
10.      out.println("<p><input type='submit'
11.                  value='End Session'></p></form>");
12.      hs.invalidate();//Session Invalidated
13.  }
14. }
```

# Session Management : HttpSession

Output: HttpSession3.java



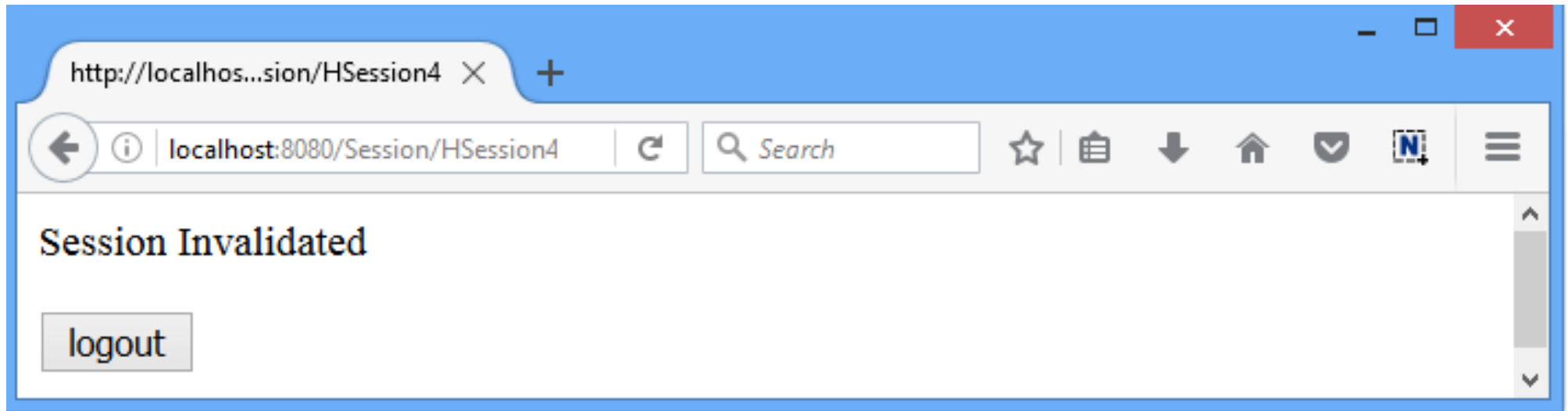
# Session Management : HttpSession

HSession4.java

```
1. public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
2.             throws ServletException, IOException
3. {   response.setContentType("text/html");
4.     PrintWriter out=response.getWriter();
5.     HttpSession hs=request.getSession(false);
6.     try
7.     {   String n=(String)hs.getAttribute("s_id");
8.     } catch (NullPointerException ne)
9.     {out.println("Session Invalidated");}
10.    out.println("<form action='/Session/httpsession.html'>");
11.    out.println("<p><input type='submit'
                    value='logout'></p></form>");
12. } //doGet
```

# Session Management : HttpSession

Output: HttpSession4.java



# Session Timeout

# Session Timeout


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\(\)](#)

# Session Timeout

## 1. Timeout in the deployment descriptor (web.xml)

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



Here specified  
time is in  
**minutes**

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

# Session Timeout

## 2. 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);
```



Here specified  
time is in  
**seconds**

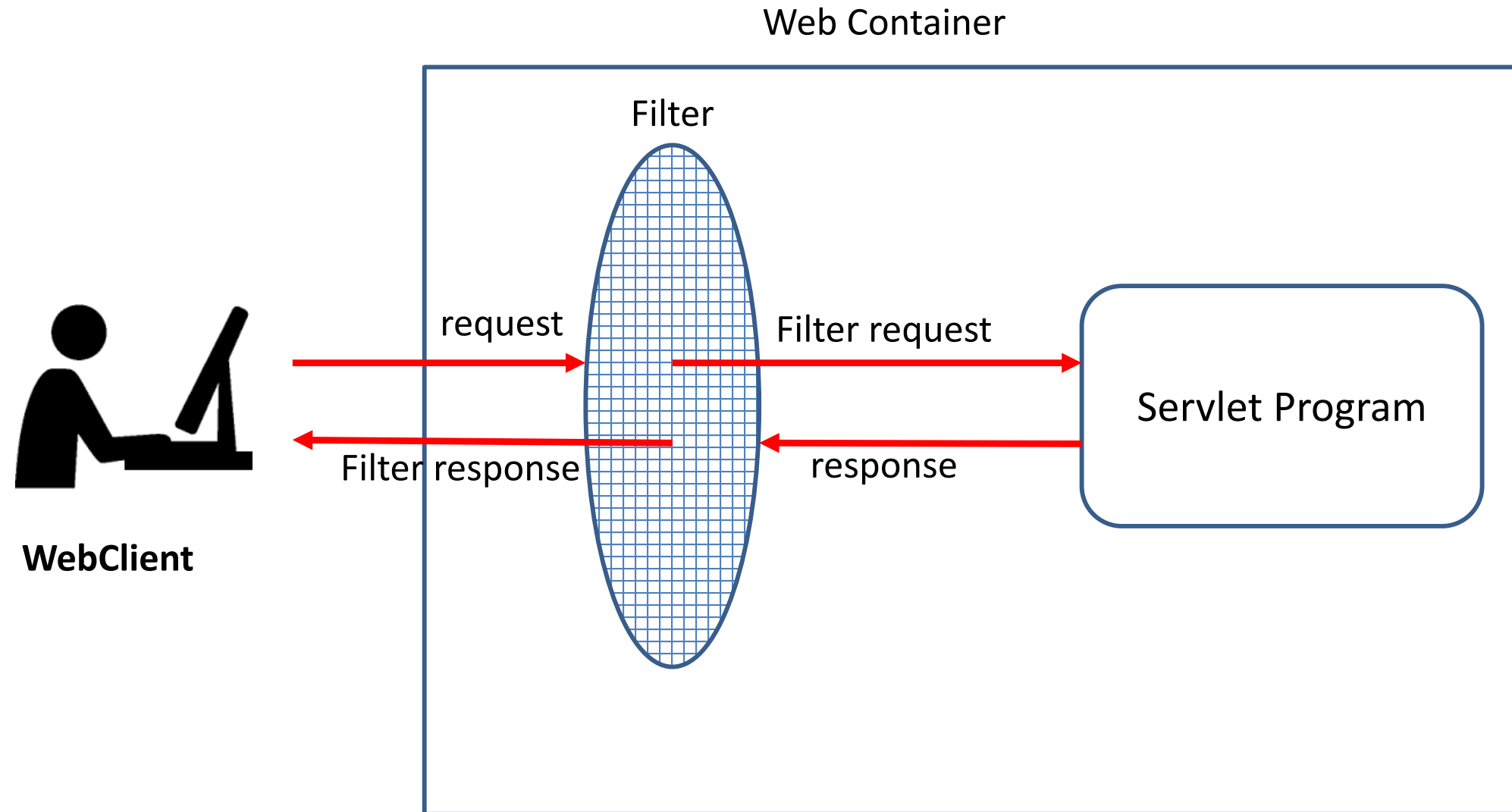


# GTU Questions

1. Write Servlet program to create cookie. Also write code to display contents of cookie on page. [7]
2. What is session? Explain session management using HttpSession. [7]
3. List the different ways to manage the session. [4]

# Filter API

# Filter API



# Filter

- Filter is used for pre-processing of requests and post-processing of responses.
- Filters are configured in the deployment descriptor of a web application.

# Filter

## Usage of Filter

- 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 API

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

1. Filter
2. FilterChain
3. FilterConfig

# Filter Interface

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

## *Method*

|   |  |
|---|--|
| void <b>init</b> (FilterConfig config)  | init() method is invoked only once. It is used to initialize the filter.   |
| void <b>doFilter</b><br>(HttpServletRequest request,<br>HttpServletResponse response,<br>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 <b>destroy</b> ()  | This is invoked only once when filter is taken out of the service.   |

# Filter Interface

## *Example*

```
public void init(FilterConfig config)
    throws ServletException {...}

public void doFilter(ServletRequest req,
    ServletResponse resp,
    FilterChain chain)
    throws IOException, ServletException
    { //filter logic... }

public void destroy() {...}
```



# FilterChain interface

- 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:

|  |  |
|--|--|
| <code>void <b>doFilter</b><br/>(HttpServletRequest request,<br/>HttpServletResponse response)</code> | <code>It passes the control to the next filter or<br/>resource.</code> |
|--|--|

## *Example*

```
FilterChain chain;
```

```
chain.doFilter(req, resp) ; //send request to next resource
```

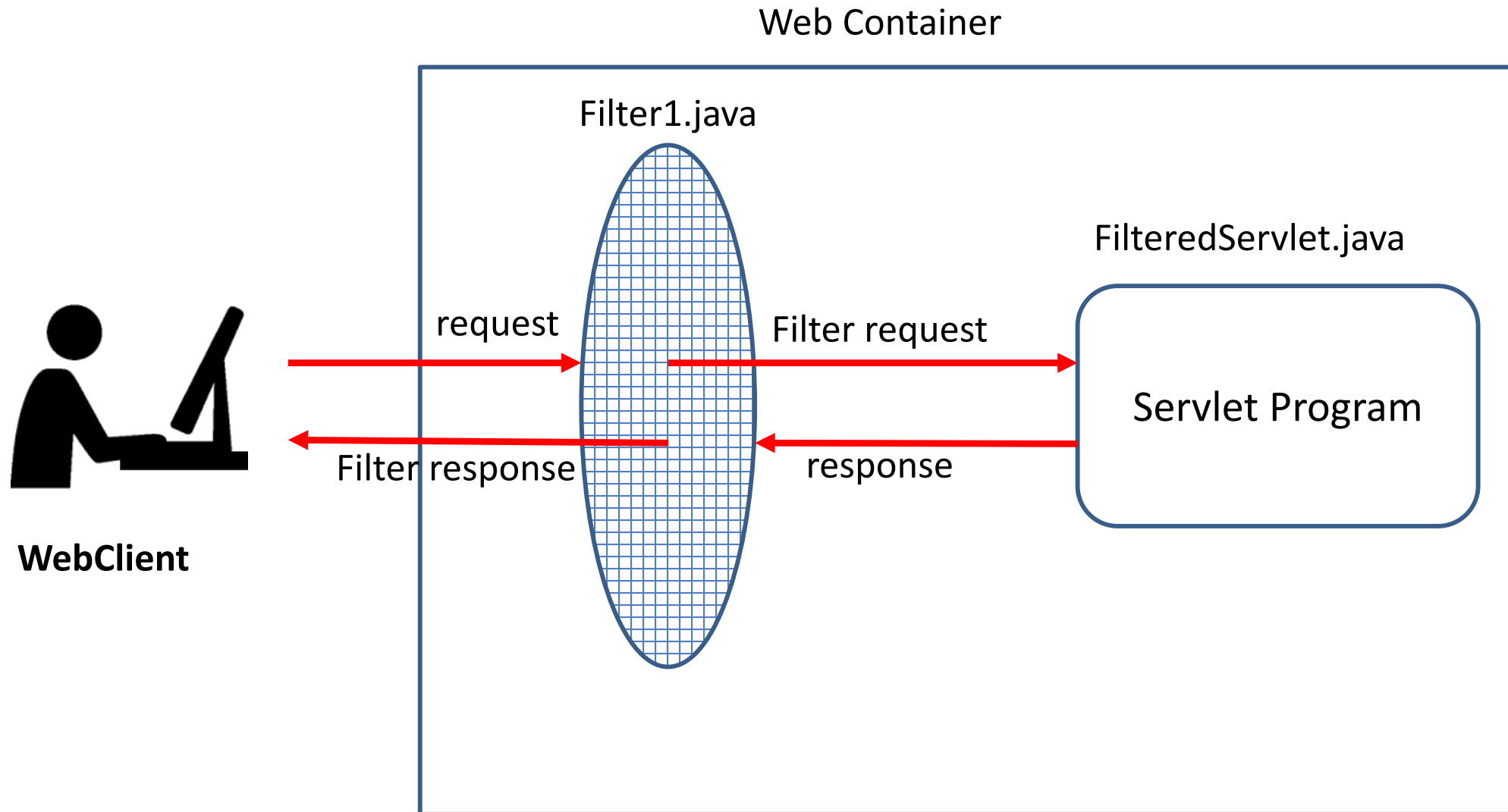
# Filter Config

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

## *Method*

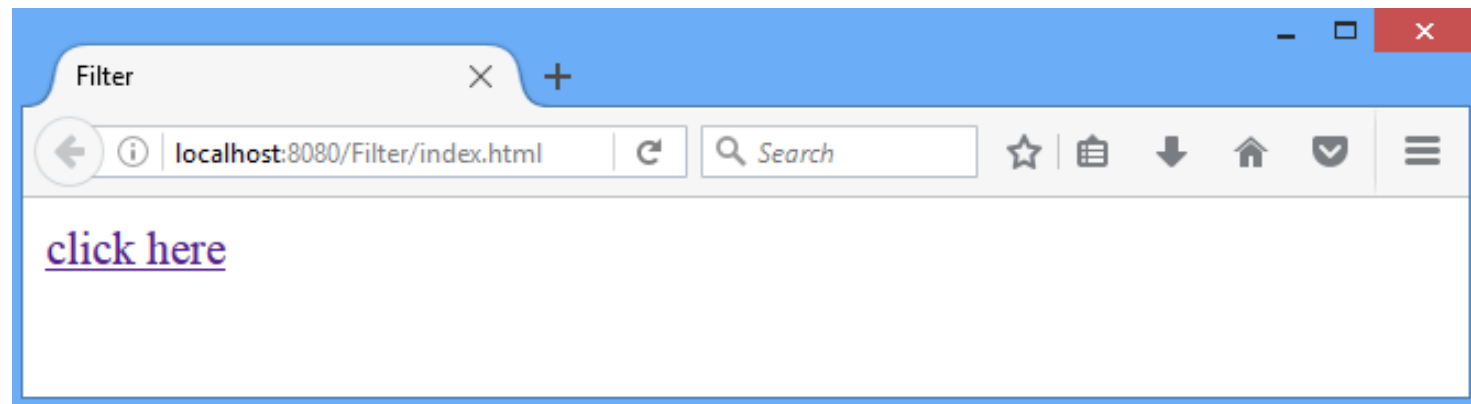
|  |   |
|--|---|
| void <b>init</b> (FilterConfig config)                   | init() method is invoked only once it is used to initialize the filter. |
| String <b>getInitParameter</b><br>(String parameterName) | Returns the parameter value for the specified parameter name.           |

# Filter Example



# Filter Example: index.html

```
1. <html>
2.     <head>
3.         <title>Filter</title>
4.     </head>
5.     <body>
6.         <a href="FilteredServlet">click here</a>
7.     </body>
8. </html>
```



# Filter Example1

web.xml

```
1. <web-app>
2. <servlet>
3.     <servlet-name>FilteredServlet</servlet-name>
4.     <servlet-class>FilteredServlet</servlet-class>
5. </servlet>
6. <servlet-mapping>
7.     <servlet-name>FilteredServlet</servlet-name>
8.     <url-pattern>/FilteredServlet</url-pattern>
9. </servlet-mapping>
```

# Filter Example1

web.xml

10.<filter>

11.           <filter-name>f1</filter-name>

12.           <filter-class>Filter1</filter-class>

13.</filter>

14.<filter-mapping>

15.           <filter-name>f1</filter-name>

16.           <url-pattern>/FilteredServlet</url-pattern>

17.</filter-mapping>

# Filter Example1: Filter1.java

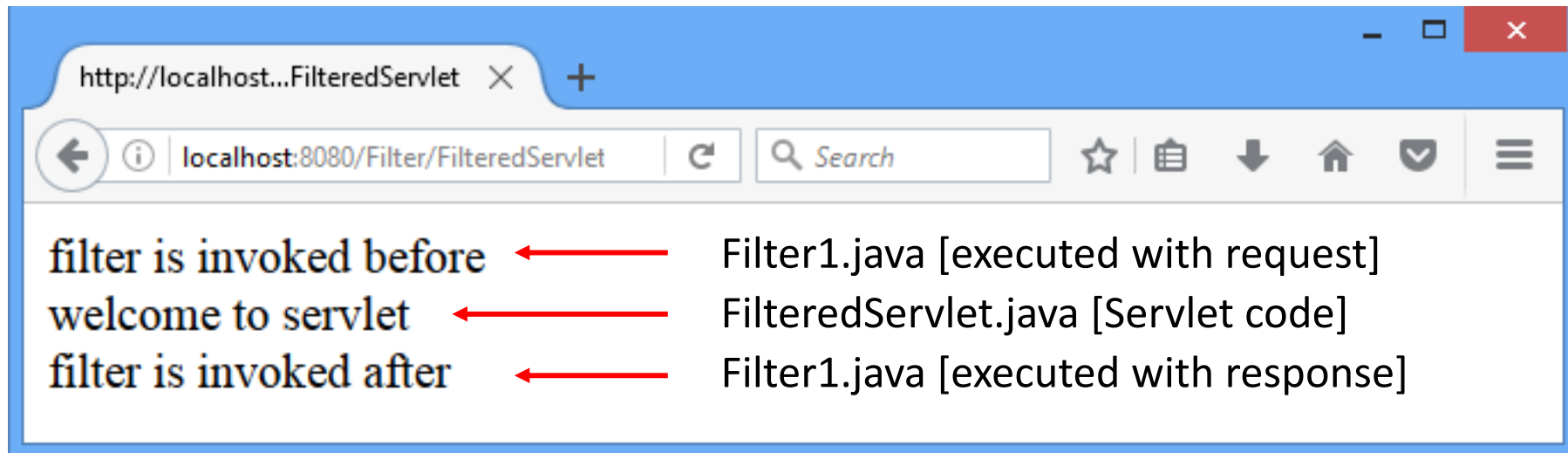
```
1. public class Filter1 implements Filter
2. {public void init(FilterConfig arg0) throws
    ServletException {//overridden init() method}
3. public void doFilter(ServletRequest req,
4.                     ServletResponse resp,FilterChain chain)
    throws IOException, ServletException
5. { PrintWriter out=resp.getWriter();
6.   out.print("filter is invoked before");//exe. with request
7.   chain.doFilter(req, resp);//send request to nextresource
8.   out.print("filter is invoked after");//exe. with response
9. }
10. public void destroy() {//overridden destroy() method}
11. }
```

# Filter Example1: FilteredServlet.java

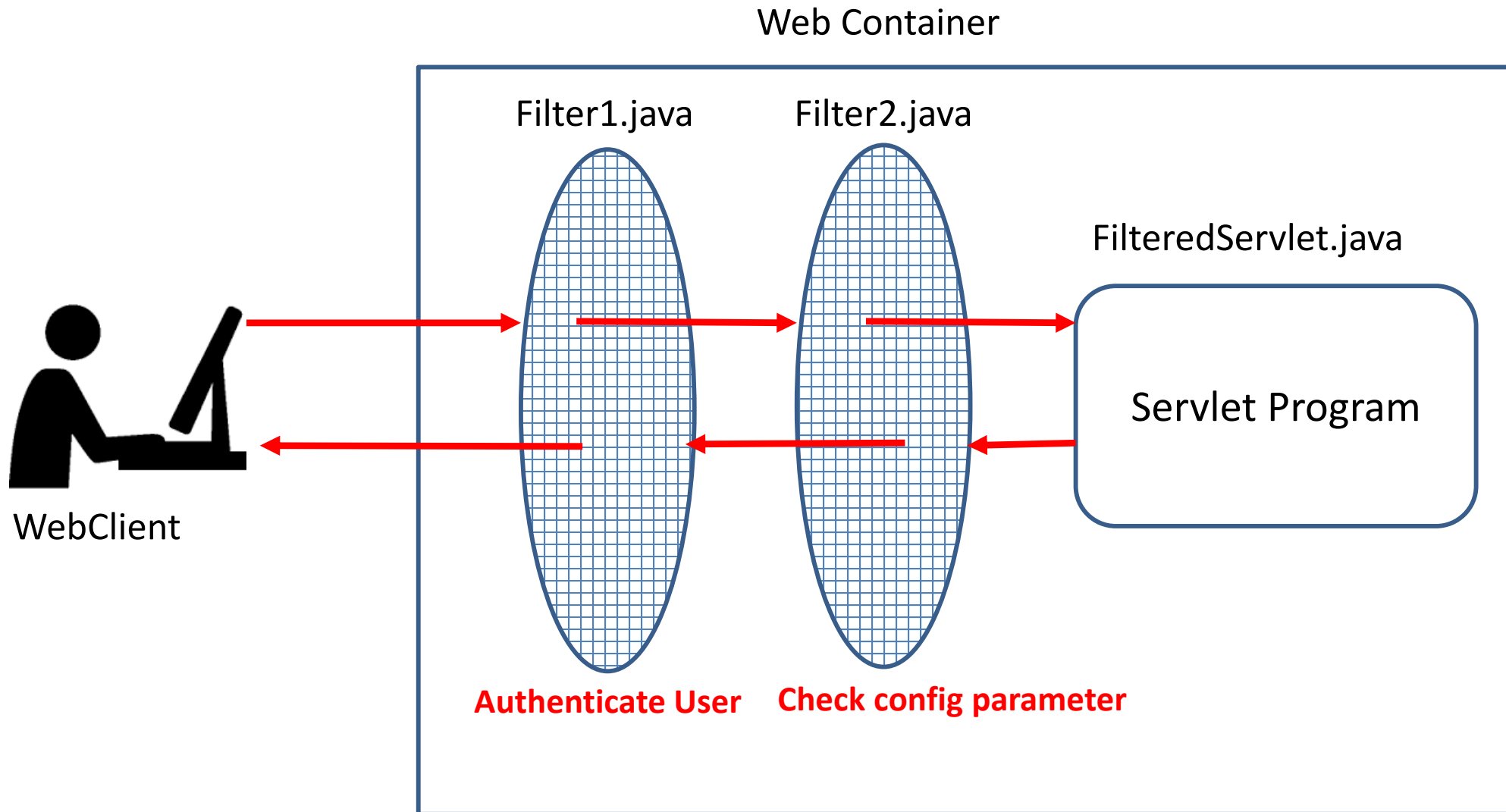
```
1. import java.io.IOException;
2. import java.io.PrintWriter;
3. import javax.servlet.*;
4. import javax.servlet.http.*;
5. public class FilteredServlet extends HttpServlet
6. {   public void doGet(HttpServletRequest request,
7.                           HttpServletResponse response)
8.     throws ServletException, IOException
9.     {
10.         response.setContentType("text/html");
11.         PrintWriter out = response.getWriter();
12.         out.println("<br>welcome to servlet<br>");
13.     }
```



# Filter Example1: Output



# Filter Example2



# Filter Example2

index.html

```
1. <html>
2.     <head>
3.         <title>filter</title>
4.     </head>
5. <body>
6.     <form action="/Filter/FilteredServlet" >
7.         <p>Login ID:<input type="text" name="login"></p>
8.         <p>Password:<input type="password" name="pwd"></p>
9.         <p><input type="submit" value="Sign In"></p>
10.    </form>
11. </body>
12.</html>
```

# Filter Example2

web.xml

```
1. <web-app>
2. <servlet>
3.     <servlet-name>FilteredServlet</servlet-name>
4.     <servlet-class>FilteredServlet</servlet-class>
5. </servlet>
6. <servlet-mapping>
7.     <servlet-name>FilteredServlet</servlet-name>
8.     <url-pattern>/FilteredServlet</url-pattern>
9. </servlet-mapping>
```

# Filter Example2

web.xml

10.<filter>

11.           <filter-name>f1</filter-name>

12.           <filter-class>Filter1</filter-class>

13.</filter>

14.<filter-mapping>

15.           <filter-name>f1</filter-name>

16.           <url-pattern>/FilteredServlet</url-pattern>

17.</filter-mapping>

# Filter Example2

web.xml

```
18.<filter>
19.    <filter-name>f2</filter-name>
20.    <filter-class>Filter2</filter-class>
21.    <init-param>
22.        <param-name>permit</param-name>
23.        <param-value>yes</param-value>
24.    </init-param>
25.</filter>
26.<filter-mapping>
27.    <filter-name>f2</filter-name>
28.    <url-pattern>/FilteredServlet</url-pattern>
29.</filter-mapping>
30.</web-app>
```

# Filter Example2

Filter1.java

```
1. public class Filter1 implements Filter{
2.     public void init(FilterConfig config) {}
3.     public void doFilter(ServletRequest req,
4.                           ServletResponse resp, FilterChain chain)
5.         throws IOException, ServletException
6.     {   PrintWriter out=resp.getWriter();
7.         out.print("<p>filter1 is invoked before</p>");
8.         if(req.getParameter("login").equals("java") &&
9.            req.getParameter("pwd").equals("servlet"))
10.        { chain.doFilter(req, resp); //send request to next resource
11.        } //if
12.        else
13.        {out.print("<p>invalid login/password</p>");} //else
14.
15.        out.print("<p>filter1 is invoked after</p>");
16.    }
17. public void destroy() {} }
```

# Filter Example2

Filter2.java

```
1. public class Filter2 implements Filter{
2.     String permission;
3.     public void init(FilterConfig config) throws ServletException
4.     {
5.         permission=config.getInitParameter("permit");
6.     }
7.     public void doFilter(ServletRequest req, ServletResponse resp,
8.         FilterChain chain) throws IOException, ServletException
9.     {
10.         PrintWriter out=resp.getWriter();
11.         out.print("<p>filter2 is invoked before</p>");
12.
13.         if(permission.equals("yes"))
14.             { chain.doFilter(req, resp);}//if
15.         else
16.             { out.println("Permission Denied"); }//else
17.
18.         out.print("<p>filter2 is invoked after</p>");
19.     }
20.     public void destroy() {}}
```

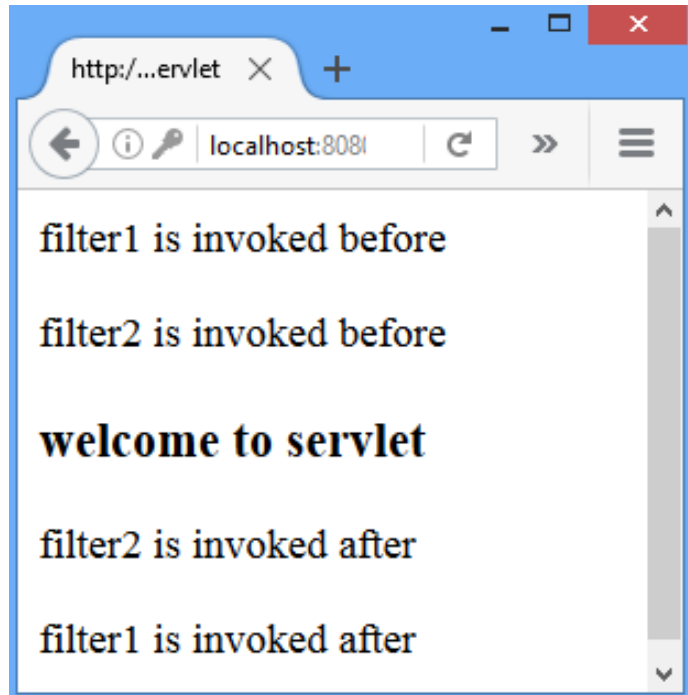


# Filter Example2

FilteredServlet.java

```
1. public class FilteredServlet extends HttpServlet {
2.     public void doGet(HttpServletRequest request,
3.                         HttpServletResponse response)
4.         throws ServletException, IOException
5.     {
6.         response.setContentType("text/html");
7.         PrintWriter out = response.getWriter();
8.         out.println("<p><h3>welcome to servlet</h3></p>");
9.     }
```

# Filter Example2:output



# Filter

## 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.

# GTU Questions:Filter

|    |   |                  |
|----|---|------------------|
| 1. | What is Filter? List the applications of filter. [3]                | Win'17<br>Win'18 |
| 2. | Explain the configuration of filter using deployment descriptor.[4] | Sum'18           |

# Servlet with JDBC

# Servlet with JDBC

```
1. import java.io.*;
2. import java.sql.*;
3. import javax.servlet.*;
4. import javax.servlet.http.*;
5. public class JDBCServlet extends HttpServlet
6. {
7.     public void doGet(HttpServletRequest request,
8.                         HttpServletResponse response)
9.                         throws ServletException, IOException
10.    {    response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        //Program continued in next slide...
        ...
```

# Servlet with JDBC

```
11. try{
12.     Class.forName("com.mysql.jdbc.Driver");
13.     Connection con=DriverManager.getConnection
        ("jdbc:mysql://localhost:3306/ajava","root","");
14.     Statement st=con.createStatement();
15.     ResultSet rs=st.executeQuery("select * from cxcy");
16.     while(rs.next())
17.     { out.println("<p>" + rs.getInt(1));
18.         out.println(rs.getString(2));
19.         out.println(rs.getString(3) + "</p>");
20.     }
21. }catch (Exception e)
22. {out.println("<p>inside exception" + e.toString() + "</p>");}
23. }//doGet()
24. }//Class
```

# Types of Servlet Events

- 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
  - Initialization of application
  - Destroying an application
  - Request from client
  - Creating/destroying a session
  - Attribute modification in session etc.



# Types of Servlet Events

- 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.                          |

# Servlet Interview Questions

|     |  |
|-----|--|
| 1.  | Who is responsible to create the object of servlet?  |
| 2.  | What is difference between Get and Post method?  |
| 3.  | When servlet object is created?  |
| 4.  | What is difference between PrintWriter and ServletOutputStream?  |
| 5.  | What is difference between GenericServlet and HttpServlet?   |
| 6.  | Can you call a jsp from the servlet?   |
| 7.  | Difference between forward() method and sendRedirect() method ?  |
| 8.  | What is difference between ServletConfig and ServletContext?   |
| 9.  | What happens if you add a main method to servlet?  |
| 10. | What is MIME Type?   |
| 11. | Why main() is not written in servlets programs?  |
| 12. | How does the JVM execute a servlet compared with a regular Java class?   |
| 13. | 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? |
| 14. | What is Connection Pooling?  |

# Servlet Interview Questions

|     |   |
|-----|---|
| 15. | Servlet is Java class. Then why there is no constructor in Servlet? Can we write the constructor in Servlet? Justify your answer. |
|-----|---|