Java Servlet

V.Mareeswari
Assistant Professor
School of Information Technology & Engineering
VIT University, Vellore

Server-Side Technologies

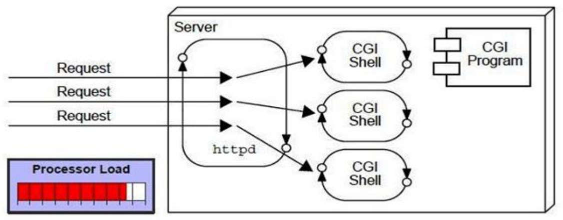
- There are many server-side technologies available: Java-based (servlet, JSP, JSF, Struts, Spring, Hibernate), ASP, PHP, CGI Script, and many others.
- Java servlet is the *foundation* of the Java server-side technology, JSP (JavaServer Pages), JSF (JavaServer Faces), Struts, Spring, Hibernate, and others, are extensions of the servlet technology.

Introduction

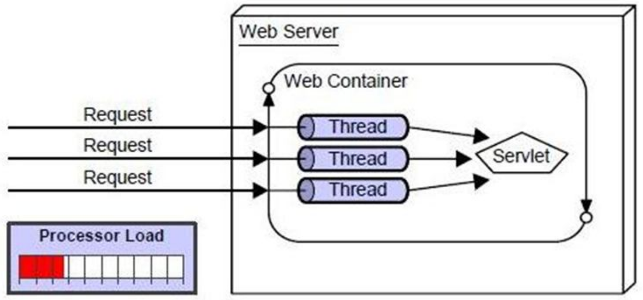
- **Servlet** technology is used to create web application (resides at server side and generates dynamic web page).
- **Servlet** technology is robust and scalable as it uses the java language. Before Servlet, CGI (Common Gateway Interface) scripting language was used as a server-side programming language.
- There are many interfaces and classes in the servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse etc.

Disadvantages of CGI (Common Gateway Interface)

- If number of clients increases, it takes more time for sending response.
- For each request, it starts a process and Web server is limited to start processes.
- It uses platform dependent language e.g. C, C++, perl.



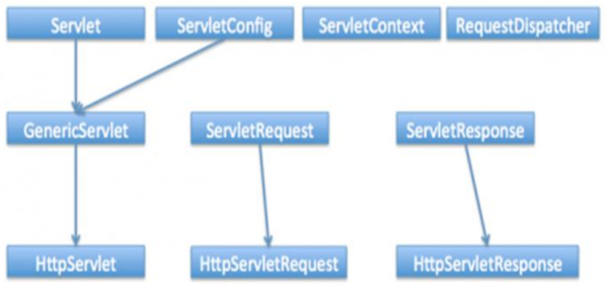
The web container creates threads for handling the multiple requests to the servlet.



Advantages of Servlet over CGI

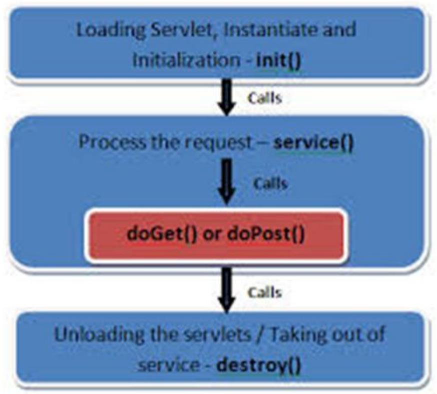
- The web container creates threads for handling the multiple requests to the servlet. Threads have a lot of benefits over the Processes such as they share a common memory area, lighweight, cost of communication between the threads are low. The basic benefits of servlet are as follows:
- **better performance:** because it creates a thread for each request not process.
- **Portability:** because it uses java language. [any standard web container such as Tomcat, JBoss, Glassfish servers and on operating systems such as Windows, Linux, Unix, Solaris, Mac etc.]
- **Robust:** Servlets are managed by JVM so no need to worry about memory leak, garbage collection etc.
- **Secure:** because it uses java language..

Servlet API Hierarchy



- javax.servlet.Servlet is the base interface of Servlet API.
- javax.servlet.ServletConfig is used to pass configuration information to Servlet.
- javax.servlet.ServletContext interface provides access to web application variables to the servlet.
- ServletRequest interface is used to provide client request information to the servlet.
- ServletResponse interface is used by servlet in sending response to the client.
- RequestDispatcher interface is used to forward the request to another resource that can be HTML, JSP or another servlet in the same context.
- HTTPServlet is an abstract class that extends GenericServlet and provides base for creating HTTP based web applications

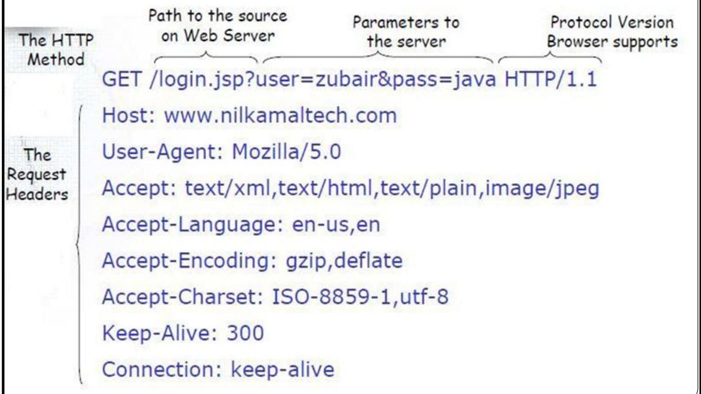
Life Cycle of Servlet


Loading Servlet, Instantiate and Initialization - init()
Calls
Process the request – service()
Calls
doGet() or doPost()
Calls
Unloading the servlets / Taking out of service - destroy()

Difference between Get and Post

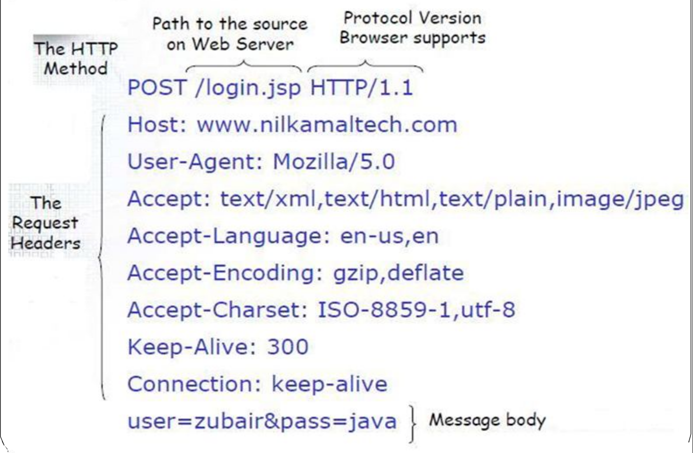
GET	POST
1) In case of Get request, only **limited amount of data** can be sent because data is sent in header.	In case of post request, **large amount of data** can be sent because data is sent in body.
2) Get request is **not secured** because data is exposed in URL bar.	Post request is **secured** because data is not exposed in URL bar.
3) Get request **can be bookmarked**	Post request **cannot be** bookmarked
4) Get request is **idempotent**. It means second request will be ignored until response of first request is delivered.	Post request is **non-idempotent**
5) Get request is **more efficient** and used more than Post	Post request is **less efficient** and used less than get.

Anatomy of Get Request


The HTTP Method
Path to the source on Web Server
Parameters to the server
Protocol Version Browser supports
The Request Headers

```
GET /login.jsp?user=zubair&pass=java HTTP/1.1
Host: www.nilkamaltech.com
User-Agent: Mozilla/5.0
Accept: text/xml,text/html,text/plain,image/jpeg
Accept-Language: en-us,en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8
Keep-Alive: 300
Connection: keep-alive
```

Anatomy of Post Request


The HTTP Method
Path to the source on Web Server
Protocol Version Browser supports
The Request Headers
Message body

```
POST /login.jsp HTTP/1.1
Host: www.nilkamaltech.com
User-Agent: Mozilla/5.0
Accept: text/xml,text/html,text/plain,image/jpeg
Accept-Language: en-us,en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8
Keep-Alive: 300
Connection: keep-alive
user=zubair&pass=java
```

- public void init(ServletConfig config) throws ServletException
- public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException
- public void destroy()
- void doGet(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException
- void doPost(HttpServletRequest *req*, HttpServletResponse *res*) throws IOException, ServletException

In NetBeans

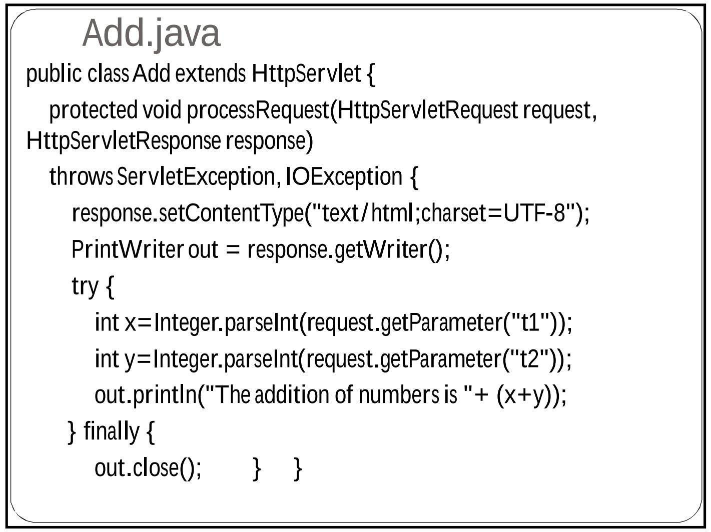
- File → New project → Java Web → Web Application
- Project Name: First Project Location: C:\Temp → Next
- Server : Apache Tomcat 6.0.18 → Next → Finish
- File → New File → Web → Servlet → Next
- Class Name : Welcome → Next → Finish
- Do changes in try block
- Run → Build Project (First)
- See the message "Build Successful " in bottom of the running window
- Run → Run File or Source Packages → Welcome.java → (right click) Run File

V.MAREESWARI / AP / SITE / VITU

```
protected void doGet(HttpServletRequest request,
  HttpServletResponse response)   throws ServletException,
  IOException {
     response.setContentType("text/html");
     PrintWriter out = response.getWriter();
       Date d=new Date();
       out.println("<html>");         out.println("<head>");
       out.println("<title>Servlet secondservlet</title>");
       out.println("</head>");         out.println("<body>");
       out.println("<h1>" + d + "</h1>");
       out.println("</body>");         out.println("</html>");
       out.close();       }    }
```

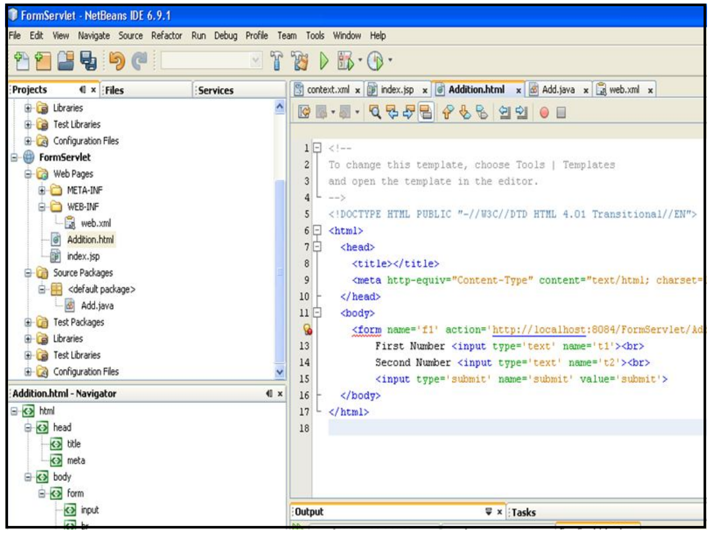
Addition.html

```
<form name='f1' action='http://localhost:8084/FormServlet/Add'
method='post'>
First Number <input type='text' name='t1'><br>
Second Number <input type='text' name='t2'><br>
<input type='submit' name='submit' value='submit'></form>
```

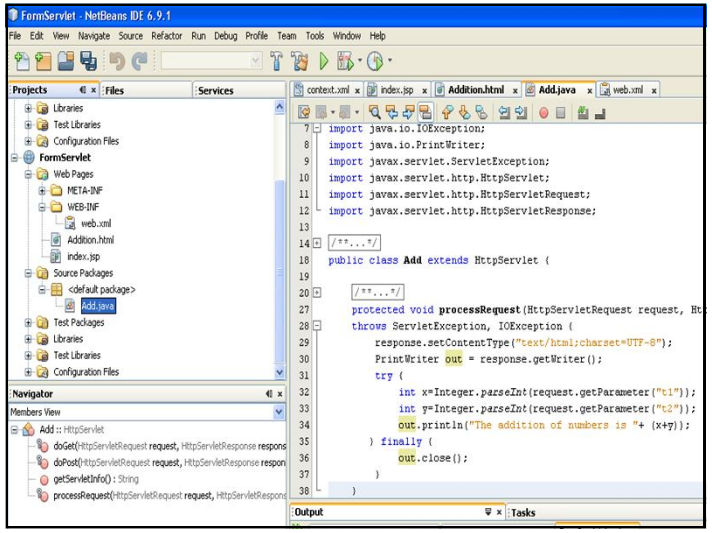


Add.java

```
public class Add extends HttpServlet {
    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            int x=Integer.parseInt(request.getParameter("t1"));
            int y=Integer.parseInt(request.getParameter("t2"));
            out.println("The addition of numbers is "+ (x+y));
        } finally {
            out.close();        }     }
```



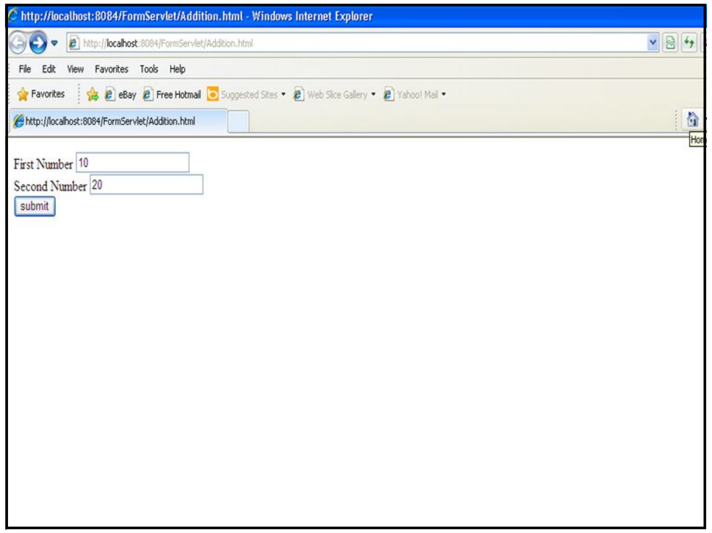
```
<!--
To change this template, choose Tools | Templates
and open the template in the editor.
-->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title></title>
    <meta http-equiv="Content-Type" content="text/html; charset=
  </head>
  <body>
    <form name='f1' action='http://localhost:8084/FormServlet/Ad
        First Number <input type='text' name='t1'><br>
        Second Number <input type='text' name='t2'><br>
        <input type='submit' name='submit' value='submit'>
  </body>
</html>
```



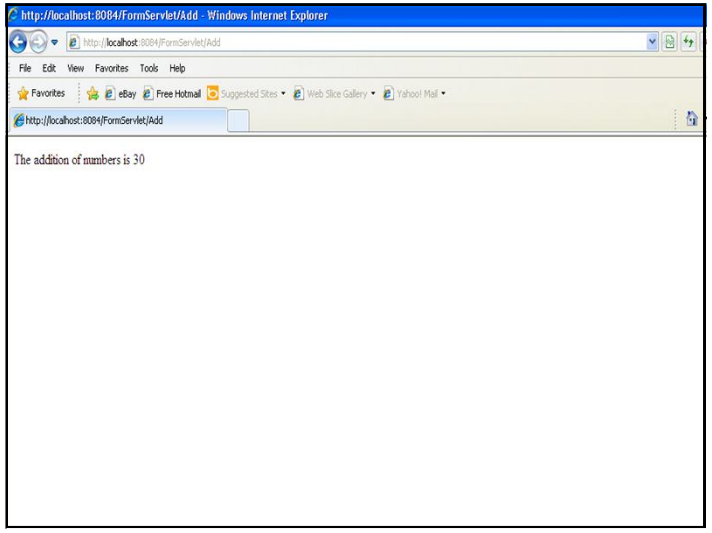
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**...*/
public class Add extends HttpServlet {

    /**...*/
    protected void processRequest(HttpServletRequest request, Ht
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            int x=Integer.parseInt(request.getParameter("t1"));
            int y=Integer.parseInt(request.getParameter("t2"));
            out.println("The addition of numbers is "+ (x+y));
        } finally {
            out.close();
        }
    }
```


http://localhost:8084/FormServlet/Addition.html - Windows Internet Explorer

First Number 10
Second Number 20
submit


http://localhost:8084/FormServlet/Add - Windows Internet Explorer
File Edit View Favorites Tools Help
Favorites
eBay
Free Hotmail
Suggested Sites
Web Slice Gallery
Yahoo! Mail

```
The addition of numbers is 30
```

Read all values of form element

```
Enumeration paramNames = request.getParameterNames();
while(paramNames.hasMoreElements()) {
      String paramName = (String)paramNames.nextElement();
      out.print(paramName);
      String[] paramValues =request.getParameterValues(paramName);
      // Read single valued data
      if (paramValues.length == 1) {
          String paramValue = paramValues[0];
          if (paramValue.length() == 0)
               out.println("<i>No Value</i>");
           else
               out.println(paramValue);
      } else {
         // Read multiple valued data
         out.println("<ul>");
         for(int i=0; i < paramValues.length; i++) {
           out.println("<li>" + paramValues[i]); }
         out.println("</ul>");        }     }
```

sendRedirect() in Servlet

- The **sendRedirect()** method of **HttpServletResponse** interface can be used to redirect response to another resource, it may be servlet, jsp or html file.
- It works at client side because it uses the url bar of the browser to make another request. So, it can work inside and outside the server.
- Example: response.sendRedirect("servlet2");

Creating custom google search using sendRedirect

```
<html>
  <head>
    <title>sendRedirect example</title>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
  </head>
  <body>
<form action="http://localhost:8084/Project/MySearch">
<input type="text" name="name">
<input type="submit" value="Google Search">
</form>
  </body>
</html>
```

Search.html

MySearch.java → Servlet file

```
public class MySearch extends HttpServlet {
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
  throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
String name=request.getParameter("name");
response.sendRedirect("https://www.google.co.in/#q="+name);
    } finally {
      out.close();    }   } }
```

RequestDispatcher in Servlet

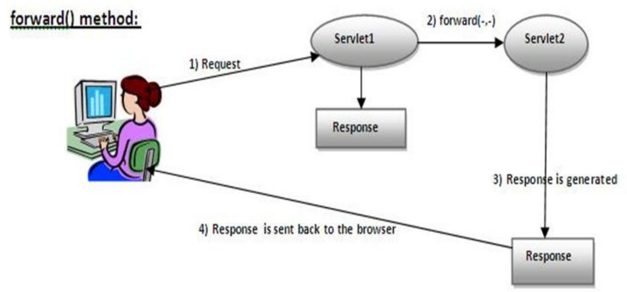
- The RequestDispacher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp.It is one of the way of servlet collaboration.
- There are two methods defined in the RequestDispatcher interface.

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

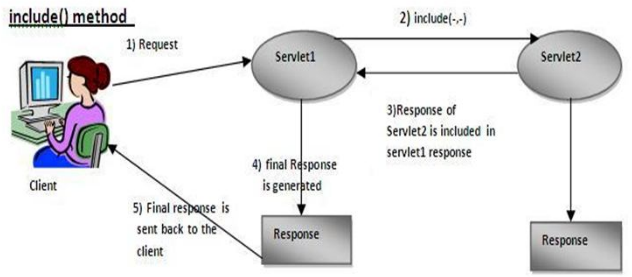
Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException

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

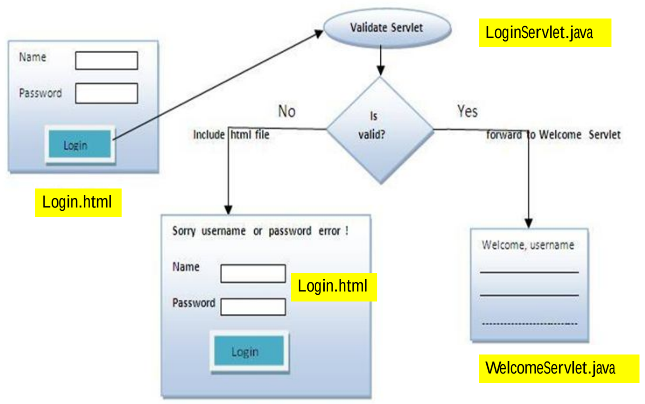


Response of second servlet is sent to the client. Response of the first servlet is not displayed to the user.


include() method
1) Request
Servlet1
2) include(-,-)
Servlet2
3)Response of Servlet2 is included in servlet1 response
4) final Response is generated
Client
5) Final response is sent back to the client
Response
Response

Response of second servlet is included in the response of the first servlet that is being sent to the client.

Example of RequestDispatcher interface



Login.html

```
<form action="servlet1" method="post">
Name:<input type="text" name="userName"/><br/>
Password:<input type="password" name="userPass"/><br/>
<input type="submit" value="login"/>
</form>
```

LoginServlet.java

```
String n=request.getParameter("userName");
   String p=request.getParameter("userPass");
   if(p.equals("VIT")){
      RequestDispatcher
rd=request.getRequestDispatcher("/WelcomeServlet");
      rd.forward(request, response);
   }
   else{
      out.print("Sorry UserName or Password Error!");
      RequestDispatcher rd=request.getRequestDispatcher("/Login.html");
      rd.include(request, response);
      }
```

WelcomeServlet.java

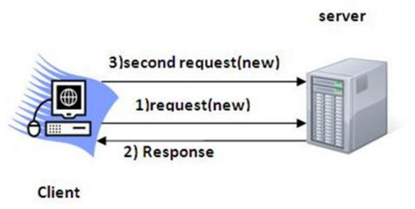
```
----------
--------
String n=request.getParameter("userName");
out.print("Welcome "+n);
-------------
-------
```

web.xml

```
<web-app>
 <servlet>
   <servlet-name>LoginServlet</servlet-name>
   <servlet-class>LoginServlet</servlet-class>
  </servlet>   ………..
  <servlet-mapping>
   <servlet-name>LoginServlet</servlet-name>
   <url-pattern> /LoginServlet </url-pattern>  → /servlet1 → in URL
  </servlet-mapping>  ……..
    <welcome-file-list>
  <welcome-file>Login.html</welcome-file>
  </welcome-file-list>
</web-app>
```

Session Tracking in Servlets

- **Session** simply means a particular interval of time.
- **Session Tracking** is a way to maintain state (data) of an user. It is also known as **session management** in servlet.
- Http protocol is a stateless so we need to maintain state using session tracking techniques. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.



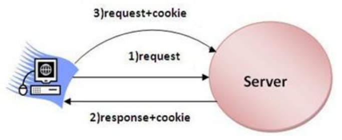
Session Tracking Techniques

There are four techniques used in Session tracking:

1. **Cookies**
2. **Hidden Form Field**
3. **URL Rewriting**
4. **HttpSession**

Cookies in Servlet

- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.
- By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



Types of Cookie

There are 2 types of cookies in servlets.

- Non-persistent cookie
- Persistent cookie

Non-persistent cookie

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

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.

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.

Note: Gmail uses cookie technique for login. If you disable the cookie, gmail won't work.

How to create Cookie?

```
Cookie ck=new Cookie("user",“marees");//creating cookie object
response.addCookie(ck);//adding cookie in the response
```

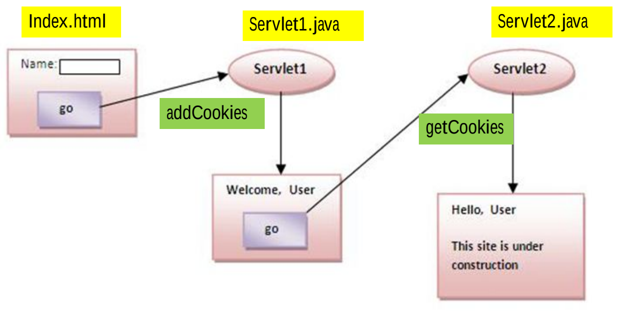
How to delete Cookie?

```
//It is mainly used to logout or signout the user.
Cookie ck=new Cookie("user","");//deleting value of cookie
ck.setMaxAge(0);//changing the maximum age to 0 seconds
response.addCookie(ck);//adding cookie in the response
```

How to get Cookies?

```
Cookie ck[]=request.getCookies();
for(int i=0;i<ck.length;i++){
out.print("<br>"+ck[i].getName()+" "+ck[i].getValue()); }
//printing name and value of cookie
```

Simple Example



Hidden Form Field

- In case of 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. This approach is better if we have to submit form in all the pages and we don't want to depend on the browser.

```
<input type="hidden" name="uname" value="marees">
```

Real application of hidden form field

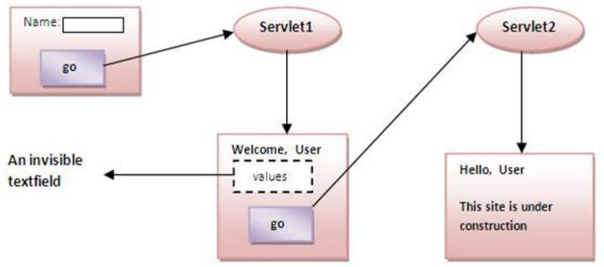
- It is widely used in comment form of a website. In such case, we store page id or page name in the hidden field so that each page can be uniquely identified.

Advantage of Hidden Form Field

- 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.



```
out.print("<input type='hidden' name='uname' value='"+n+"'>");
```

URL Rewriting

- In URL rewriting, we append a token or identifier 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&??

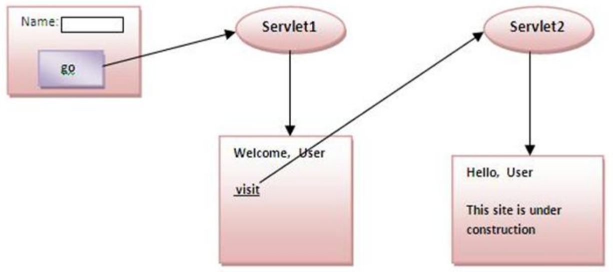
- A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&). When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can use getParameter() method to obtain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

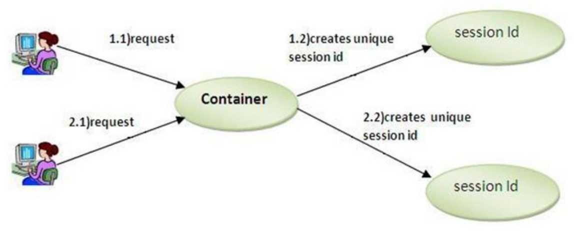
- It will work only with links.
- It can send only textual information.



```
out.print("<a href='servlet2?uname="+n+"'>visit</a>");
```

HttpSession interface

- In such case, 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:
  - bind objects
  - view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.



index.html

```
<form action="/servlet1">
Name:<input type="text" name="userName"/><br/>
<input type="submit" value="go"/>
</form>
```

Servlet1.java

```
String n=request.getParameter("userName");
out.print("Welcome "+n);
HttpSession session=request.getSession();
session.setAttribute("uname",n);
out.print("<a href='servlet2'>visit</a>");
```

Servlet2.java

```
HttpSession session=request.getSession(false);
String n=(String)session.getAttribute("uname");
out.print("Hello "+n);
```

1. **public String getId()**
2. **public long getCreationTime()**
3. **public long getLastAccessedTime()**
4. **public void invalidate()**

Multi-tier Applications: Using JDBC from a Servlet

- Many of today's applications are **three-tier distributed applications**, consisting of a
  - **User interface** (**HTML , XHTML**, Dynamic HTML or applets)
  - **Business logic (Web servers)**
  - **Database access.**
- Using the networking provided automatically by the browser, the user interface can communicate with the middle-tier business logic. The middle tier can then access the database to manipulate the data. The three tiers can reside on separate computers that are connected to a network.

V.MAREESWARI / AP / SITE / VITU

- In multi-tier architectures, Web servers often represent the middle tier. They provide the business logic that manipulates data from databases and that communicates with client Web browsers.
- Servlets, through JDBC, can interact with popular database systems. Developers do not need to be familiar with the specifics of each database system. Rather, developers use SQL-based queries and the JDBC driver handles the specifics of interacting with each database system.
- Three-tier architecture was developed by John J. Donovan in Open Environment Corporation (OEC), a tools company he founded in Cambridge, Massachusetts.

V.MAREESWARI / AP / SITE / VITU

Database Connectivity

You can use the following steps for creating DSN connection:

1. Open Data Sources (Start->Control Panel->Administrative Tool->Data Sources(ODBC)
2. Open User DSN tab
3. Add a user DSN
4. Select Microsoft Access Driver(*.mdb)
5. Select database name and Create the DSN name (e.g university.mdb)
6. Click "Ok" and then try the following Servlet code where 'DSN_Account' is our DSN:

Login.html

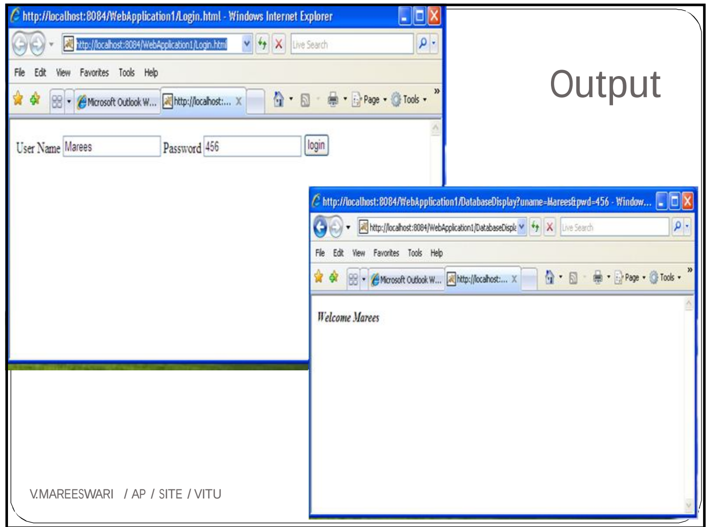
```
<html><body>
<form action="http://localhost:8084/Project/DatabaseDisplay"
   method="get">
      User Name <input type="text" name="uname">
      Password <input type="text" name="pwd">
      <input type="submit" value="login">
   </form>  </body></html>
```

```
import javax.servlet.*; import javax.servlet.http.*;
import java.sql.*; import java.io.*;
public class DatabaseDisplay extends HttpServlet {
   protected void processRequest(HttpServletRequest request,
   HttpServletResponse response)
   throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      int flag=0;
      String dbname=null;
      String dbpassword=null;
```

```
try {
out.println("<html><body>");
String name=request.getParameter("uname");
String password=request.getParameter("pwd");
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con =
   DriverManager.getConnection("jdbc:odbc:DSN_Account");
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("Select * from students");
while(rs.next()){
dbname=rs.getString("Name");
dbpassword=rs.getString("Password");
if(name.equals(dbname) && password.equals(dbpassword))
   flag=1;
 }//end of while
 if(flag==1)
          out.println("<b><i>Welcome "+name+"</i></b>");
else
       out.println("<b>Invalid User</b>");
out.println("</body></html>");
}//end of try
catch (Exception e)
{ out.println(e); }
finally {
       out.close();       }    }  }
```

V.MAREESWARI / AP / SITE / VITU

Output


http://localhost:8084/WebApplication1/Login.html - Windows Internet Explorer

User Name Marees Password 456 login

http://localhost:8084/WebApplication1/DatabaseDisplay?uname=Marees&pwd=456 - Window...

Welcome Marees

V.MAREESWARI / AP / SITE / VITU

MySQL

- *Class.forName("com.mysql.jdbc.Driver");*
- Connection connection =
- DriverManager.getConnection("jdbc:mysql://oopsla.snu.ac.kr/mydb",“userid",“password");
- **JDBC URL →** Vendor of database, Location of database server and name of database
- *Statement statement = connection.createStatement();*
- **executeQuery()** for QUERY statements
- Returns a ResultSet which contains the query results
- **executeUpdate()** for INSERT, UPDATE, DELETE, or DDL
  statements
- Returns an integer, the number of affected rows from the SQL
- **execute()** for either type of statement

- Statement stmt = conn.createStatement();
- ResultSet rset = stmt.executeQuery

("select RENTAL_ID, STATUS from ACME_RENTALS");
- Statement stmt = conn.createStatement();
- int rowcount = stmt.executeUpdate

("delete from ACME_RENTAL_ITEMS where rental_id = 1011");
- While (rs.next()){
- int id = rs.getInt(“ID”);
- String name = rs.getString(“name”);
- float score = rs.getFloat(“score”);
- System.out.println(“ID=” + id + “ ” + name + “ ” + score);}