Module 4 : Servlet

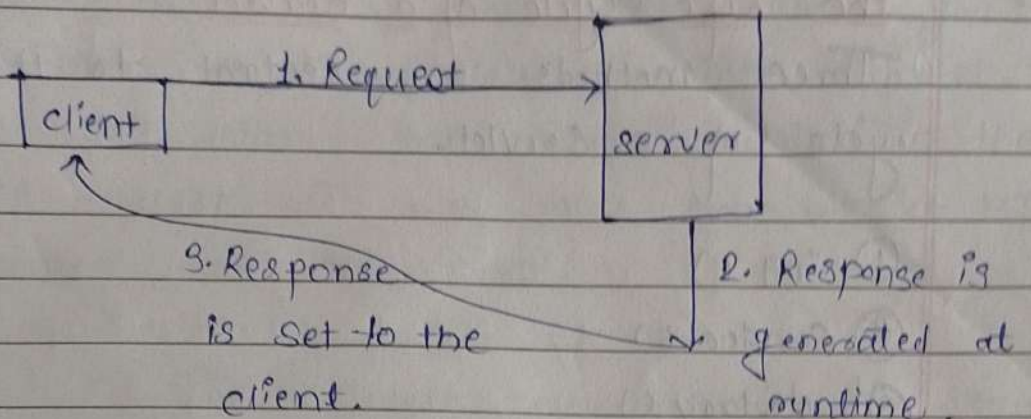
* Background :

Servlet Tutorial :

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

What is a Servlet ?

1. Servlet is a technology i.e. used to create web application.
2. Servlet is an API that provides many interfaces & classes including documentations.
3. Servlet is an interface that must be implemented for creating any servlet.
4. Servlet is a class that extends the capabilities of the servers & responds to the incoming requests. It can responds to any type of requests.
5. Servlet is a web component that is deployed on the server to create dynamic web page.



Advantages of Servlet

(1) Performance is significantly better. Servlets executes within the address space of a web server. It is not necessary to
(2) create a separate process to handle each client request.
(3) Servlets are platform-independent because they are written in java.
(3) The java security manager on the server enforces a set of restrictions to protect the resources on a server m/c.
(4) The full functionality of the java class libraries is available to a servlet.
(5) It can communicate with applets, databases, or other software via the sockets and RMI mechanisms that you have seen already.

(a) Better Performance (Because of thread)
(b) Portability (uses java language)
(c) Robust
(d) Secure

The Life cycle of a Servlet

Three methods are central to the life cycle of a Servlet.

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

Scenario : (1) Assume that a user enters a Uniform Resource Locator (URL) to a web browser. The browser then generates an HTTP request for this URL. This request is then sent to the appropriate server.

(2) This HTTP request is received by the web server. The server maps this request to a particular servlet. The servlet is dynamically retrieved & loaded into the address space of the server.

init() : (3) The server invokes the init() method of the servlet. This method is invoked only when the servlet is first loaded into memory.

It is possible to pass initialization parameters to the servlet so it may configure itself.

service() : (4) The server invokes service() method of the servlet. This method is

called to process the HTTP request. you will see that it is possible for the servlet to read data that has been provided in the HTTP request. It may also formulate an HTTP response for the client.

The Servlet remains in the servers address space & is available to process any other HTTP requests received from clients. The service() method is called for each HTTP request.

destroy() :⇒ (5) The Server may decide to unload the servlet from its memory. The algorithms by which this determination is made are specific to each server. The server calls the destroy() method to relinquish any resources such as file handles that are allocated for the servlet.

Important data may be saved to a persistent store. The memory allocated for the servlet and its objects can then be garbage collected.

init() : public void init() throws ServletException { }

destroy() : public void destroy() { }

Service() : public void destroy (ServletRequest req, ServletResponse res) throws ServletException, IOException { }

Using tomcat for servlet Development

① To create Servlets, we will need access to a Servlet development environment.

② We are using here tomcat.

③ Tomcat is an open-source product maintained by the Jakarta Project of the Apache Software foundation.

④ It contains the class libraries, documentation, & runtime support that you will need to create & test Servlet.

⑤ Default location for Tomcat is

c:\program files\Apache Software foundation\Tomcat.

⑥ If we load tomcat in different location, we need to set the environmental variable JAVA_HOME to top-level directory.

⑦ The first example, called HelloServlet, you will add the following lines in the section that defines the Servlets.

```
<Servlet>
    <Servlet-name> HelloServlet </Servlet-name>
    <Servlet-class> HelloServlet </Servlet-class>
</Servlet>
```

⑧ following lines defines the Servlet mappings.

```
<servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/Servlet1HelloServlet</url-pattern>
</servlet-mapping>
```

Steps to execute the Servlet programs:

(1) Download tomcat setup from tomcat.apache.org/download site

(2) Once setup download _ install tomcat.

(3) Open eclipse → Go to window tag → choose preference.

(4) On left hand side choose server tag → explore the tag → Select runtime environment Add appropriate tomcat setup → Next → browse tomcat directory Path → finish ok.

(5) New → Dynamic web Programming → write Name of your project. → next → finish

A Simple Servlet

The Basic steps for developing servlet program :

(a) Create & compile the servlet source code, Then, copy the servlet's class file to the proper directory, and add the servlet's name & mappings to the proper web.xml file.
(2) Start tomcat
(3) Start a web browser & request the servlet.

Program :

HelloServlet.java

```
import java.io.*;
import javax.servlet.*;
public class HelloServlet extends
GenericServlet
{

public void service (ServletRequest req,
ServletResponse res) throws ServletException,
IOException
{
res.setContentType ("text/html");
PrintWriter pw = response.getWriter();
pw.println ("<B> Hello ! ");
pw.close();
}
}
```

web.xml :

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

<servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/servlet1/HelloServlet</url-pattern>
</servlet-mapping>
</web-app>
```

① Generic Servlet :

GenericServlet class provides functionality that simplifies the creation of a servlet. eg: it provides init() and destroy(). You need supply only the service() method.

Above program defines HelloServlet as a subclass of GenericServlet.

② service() Method

This method handles requests from a client.

a) The first argument is a ServletRequest object. This enables the servlet to read data that is provided via the client request.

b) The second argument is a ServletResponse object. This enables the servlet to formulate a response for the client.

③ setContentType() :
It establishes the MIME type of the HTTP response. In this program, the MIME type is text/html. This indicates that the browser should interpret the content as HTML source code.

④ getWriter() & PrintWriter :
The getWriter() methods obtains a PrintWriter. Anything written to this stream is sent to the client as part of the HTTP response. Then println() is used to write same simple HTML source code as the HTTP response.

⭐ The Servlet API (Application Program Interface)

The javax.servlet and javax.servlet.http packages represent interfaces & classes for Servlet api.

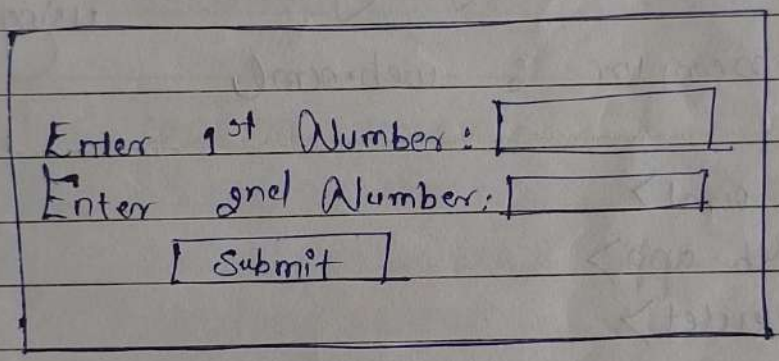
a) The javax.servlet package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.

b) The javax.servlet.http package contains interfaces and classes that are responsible for http requests only.

Home.html

```
<html>
  <body>
    <form action = "add">
      Enter 1st Number :
        <input type = "text" name = "num1">
      Enter 2nd Number :
        <input type = "text" name = "num2">
      <input type = "Submit">
    </form>
  </body>
</html>
```

output :



AddServelet.java

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class AddServelet extends HttpServelet
{
  public void Service(HttpServeletRequest req,
      HttpServeletResponse res) throw
    IoException,
```

```
{
    int i = Integer.parseInt(req.
                getParameter("num1"));
    int j = Integer.parseInt(req.
                getParameter("num2"));

    int K = i+j;
    System.out.println("Result is ":);
            // you are getting output on con

    PrintWriter p = res.getWriter();
    p.println("result is:"+p);
            // getting output on web page by
                        using response
}}
```

deployment Descriptor : web.xml

```
<?xml>
<web-app>
<Servlet>
  <servlet-name>    abc                </servlet-nam
  <Servlet-class> AddServlet           </Servlet-class
</Servlet>

<servlet-mapping>
  <Servlet-name> abc </servlet-name>
  <Servlet>
  <url-pattern> /add </url-pattern>
</Servlet-mapping>

</web-app>
```

chandra's

Servlet Config And Servlet Context Interface

web.xml

```
<web-app>
  <servlet>
    <servlet-name> abc </servlet-name>
    <servlet-class> MyServlet </servlet-class>
  </servlet>
```

Initial parameter

```
    <init-param>
      <param-name> name </param-name>
      <param-value> Sukash </param-value>
    </init-param>
  </servlet>

  <servlet-mapping>
    <servlet-name> abc </servlet-name>
    <url-pattern> /add </url-pattern>
  </servlet-mapping>
```

Initial parameter

```
  <context-param>
    <param-name> name </param-name>
    <param-value> Prachi </param-value>
  </context-param>

  <context-param>
    <param-name> Phone </param-name>
    <param-value> iphone </param-value>
  </context-param>
</web-app>
```

MyServlet.java

```
class Myservlet extends HttpServlet
{
  void service (HttpServletRequest req,
                HttpServletResponse res)
```

throws IOException, ServletException

{

PrintWriter pw = res.getWriter();
pw.print("Hi");

ServletContext ctx = getServletContext();

// tomcat provides object of servletContext for us
So don't need to connect it. By using getServletContext we are getting servletContext object free.

String s;
s = ctx.getInitParameter("name"); // output: Prachi

out.println(s); // prachi

String s2 = ctx.getInitParameter("phone");
out.println(s2) // iphone

ServletConfig cg = getServletConfig();

String str = cg.getInitParameter("name");
out.println(str);

}

}

Servlet API

- javax.servlet package
  - Interfaces
    1) Servlet
    2) ServletRequest
    3) ServletResponse
    4) ServletConfig
    5) ServletContext
  - classes
    1) GenericServlet
    2) ServletInputStream
    3) ServletOutputStream
    4) ServletException
    5) UnavailableException
- javax.servlet.http Package
  - Interfaces
    1) HttpServletRequest
    2) HttpServletResponse
    3) HttpSession
    4) HttpSessionBindingListener
  - classes
    1) Cookie
    2) HttpServlet
    3) HttpSessionEvent
    4) HttpSessionBindingEvent

javax.servlet Interfaces

1) Servlet : Interface :

1) All servlet must implement the servlet interface
2) It declares the init(), service(), destroy() method that are called by the server during cycle of a servlet.
3) getServletConfig() : It is called by the servlet to obtain initialization

parameter. like author name, version of Servlet, date, copyright information.

④ A servlet developer overrides the getServletInfo() method to provide a String with useful information. Information like author name, version of servlet, date, copyright details.

-Methods P.T.O.

② ServletRequest Interface

It enables a servlet to obtain info about a client request.

Methods

① object getAttribute (String attr)
: Returns the value of the attribute named attr.

② String getCharacterEncoding()
: Returns the character encoding of the request.

③ int getContentLength()
: Returns the size of the request. The value -1 is returned if the size is unavailable.

④ String getContentType()
: Returns the type of the request. A null value is returned if the type cannot be determined.

⑤ ServletInputStream getInputStream() throws IOException.

: Returns a ServletInputStream that can be used to read binary data from the request.

(6) String getParameter(String pname)
: Returns the value of the parameter named pname.

(7) Enumeration getParameterNames()
: Returns an enumeration of the parameter names for this request.

(8) String[] getParameterValues(String name)
: Returns an array containing values associated with the parameter specified by name.

(9) String getProtocol()
: Returns a description of the Protocol.

(10) String getRemoteAddr()
: Returns the string equivalent of the client IP address.

(11) String getRemoteHost()
: Returns the string equivalent of the client host name.

(12) String getScheme()
: Returns the transmission scheme of the URL used for the request (eg: http, ftp)

(13) String getServletName()
: Returns the name of the servlet.

(14) int getServerPort()
: Returns the port number.

(3) ServletResponse Interface
It enables a servlet to formulate a response for a client.

Methods

(1) String getCharacterEncoding()
: Returns the character encoding for the response.

(2) ServletOutputStream getOutputStream()
throws IOException
: Returns a ServletOutputStream that can be used to write binary data to the response.

(3) PrintWriter getWriter() throws IOExcepti
: Returns a PrintWriter that can be used to write character data to the response.

(4) Void setContentLength(int size)
: Sets the content length for the respon to size.

(5) Void setContentType(String type)
: Sets the content type for response type.

(4) ServletConfig Interface:

It allows a servlet to obtain configuration data when it is loaded.

Methods:

(1) ServletContext getServletContext()
: Returns the context for this servlet.

(2) String getInitParameter(String param)
: Returns the value of the initialization parameter named param.

(3) Enumeration getInitParameterNames()
: Returns an enumeration of all initialization parameter names.

(4) String getServletName()
: Returns the name of the invoking Servlet.

(5) ServletContext Interface

It enables Servlet to obtain info about their environment.

Methods

(1) Object getAttribute(String attr)
: Returns the value of the server attribute named attr.

(2) String getMimeType(String file)
: Returns the MIME type of file

(3) String getServerInfo()
: Returns information about the server

(4) Void log(String s)
: writes s to the servlet log.

(5) Void log(String s, Throwable e)
: writes s and the stack trace for e to the servlet log.

(6) Void setAttribute(String attr, Object val)
: Sets the attribute specified by attr to the value passed in val.

ServletInterface Methods

(1) void destroy() :
called when the servlet unloaded.

(2) ServletConfig getServletConfig()
: Returns a ServletConfig object that contains any initialization parameters

(3) String getServletInfo()
: Returns a string describing the servlet.

(4) Void init(ServletConfig sc) throws ServletException
: Called when the servlet is initialized. Initialization parameter for the servlet can be obtained from sc

An unavailableException should be thrown if the servlet can't be initialized.

(5) void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
: called to process a request from a client.
The request from the client can be written to res.
An exception is generated if a servlet or IO problem occurs.

Classes in javax.servlet package

1. GenericServlet
2. ServletInputStream
3. ServletOutputStream
4. ServletException
5. UnavailableException.

1. GenericServlet :

1) It provides implementations of the basic life cycle methods for a servlet.
2) It implements the Servlet and ServletConfig interfaces.
3) In addition, a method to append a string to the server log file is available.

```
void log(String s)
void log(String s, Throwable e)
```

2. ServletInputStream :

1. It extends InputStream.
2. It is implemented by the servlet container & provides an input stream that a servlet developer can use to read the data from a client request.

3. ServletOutputStream :

1. It extends OutputStream.
2. It is implemented by the Servlet container & provides an output stream that a servlet developer can use to

write data to a client response.

(4) Servlet Exception classes.

:) It indicates that a servlet problem has occured.

(5) Unavailable Exception

It extends ServletException, It indicates that a servlet is unavilable.

Reading Servlet Parameter

① getParameter(): To get value of a form parameter

② getParameterValues(): Call this method if the parameter appears more than once & return multiple value.

③ getParameterNames(): to get complete list of all parameters in current request.

Program:

PostParameters.html

```
<html>
<body>
<center>
<form name = "form1"
      method = "post"
      action = "http://localhost:8080/servlets-
                example/Servlet/PostParameters">
<table>
 <tr>
   <td> <B> Employee </td>
   <td> <input type = textbox name = "e"
        size = "25" value = "" ></td>
 </tr>
 <tr>
   <td> <B> Phone </td>
   <td><input type = textbox name = "p"
        size = "25" value = "" ></td>
 </tr>
</table>
```

```
<input type = Submit value = "Submit">
</body>
</html>
```

PostParameter.java.

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

public class PostParameter extends
        GenericServlet
{
  public void Service (ServletRequest request,
  ServletResponse response)
   throws ServletException, IOException
  {
    PrintWriter pw = response.getWriter();

    Enumeration e = request.getParameterNames();

    while (e.hasMoreElements())
    {
      String pname = (String)e.nextElement();
      pw.println (pname + "=");
      String pvalue = request.getParameter(pname
       pw.println (pvalue);

    }
      pw.close();
  }
}
```

The javax.servlet.http Package

(1) Interfaces : (Protocol oriented)

1) HttpServletRequest
2) HttpServletResponse
3) HttpSession
4) HttpSessionBindingListener

(2) Classes :

1) Cookie
2) HttpServlet
3) HttpSessionEvent
4) HttpSessionBindingEvent

(1) The HttpServletRequest Interface

The HttpServletRequest interface enables a servlet to obtain information about a client request.

Methods :

(1) String getAuthType()
: Returns authentication scheme.

(2) Cookie[] getCookies()
: Returns an array of the cookies in this request.

(3) long getDateHeader(String field)
: Returns the value of the date header field named field.

(4) String getHeader(String field)
: Return the value of the header

field named field.

(5) Enumeration getHeaderNames()
Returns an enumeration of the header names.

(6) int getIntHeader(String field)
:Returns the Int equivalent of the header field named field.

(7) String getMethod()
:Returns the HTTP method for this request

(8) String getQueryString()
: Returns any query string in the URL.

(9) String getRemoteUser()
: Returns the name of the user who issued this request.

(10) String getRequestedSessionId()
: Returns the ID of the session.

(11) String getRequestURI()
: Returns the URL.

(12) StringBuffer getRequestURL()
: Returns the URL.

(13) HttpSession getSession()
: Returns the session for this request. If a session does not exist, one is

created & then returned.

(14) HttpSession getSession(boolean new)
If new is true and no session exists, creates & returns a session for this request. otherwise, returns the existing session for this request.

(15) boolean isRequestedSessionIdFromCookie()
: Returns true if a cookie contains the SessionID. otherwise, returns false.

(16) boolean isRequestedSessionFromURL()
: Return true if the URL contains the session ID. otherwise, returns false.

(2) The HttpServletResponse Interface

1) The HttpServletResponse interface enables a servlet to formulate an HTTP response to a client.
2) Several constants are defined. These correspond to the different status codes that can be assigned to an HTTP response.

eg: "SC_OK" indicates that the HTTP request succeeded, & "SC_NOT_FOUND" indicates that the requested resource is not available.

Methods :

① Void addCookie(Cookie cookie)
: Adds cookie to the HTTP response.

(2) boolean containsHeader (String field)
: Returns true if the HTTP response header contains a field named field.

(3) void sendError (int c) throws IOException
: Sends the error code c to the client

(4) void sendError (int c, String s) throws IOExcep
: Redirects the client to url.

(5) void setDateHeader (String field, long msec)
: Adds field to the header with date value equal to msec (milliseconds since midnight, January 1, 1970, GMT)

(6) void setHeader (String field, long value)
: Add field to the header with value equal to value.

(7) void setStatus (int code)
: Sets the status code for this response to code.

(3) The HttpSession Interface

The HttpSession interface enables a servlet to read & write the state information that is associated with an HTTP session.

Methods :

(1) Object getAttribute (String attr)
: Returns the value associated with the name passed in attr. Returns null if attr is not found.

(2) Enumeration getAttributeNames()
: Returns an enumeration of the attribute names associated with the session.

(3) long getCreationTime()
: Returns the time when this session was created.

(4) String getId()
: Returns the session ID

(5) long getLastAccessedTime()
: Returns the time (in milliseconds since midnight, January 1, 1970) when the client last made a request for this session.

(6) void invalidate()
: Invalidates this session & removes it from the context.

(7) boolean isNew()
: Returns true if the server created the session & it has not yet been accessed by the client.

(8) void removeAttribute(String attr)
: Removes the attribute specified by attr from the session.

(9) void setAttribute(String attr, Object val)
: Associates the value passed in val wi

the attribute name passed in attr.

(4) The HttpSessionBindingListener (I)

The HttpSessionBindingListener interface is implemented by objects that need to be notified when they are bound to or unbound from an HTTP session.

void valueBound(HttpSessionBindingEvent e)
void valueBound(
void valueUnBound(HttpSessionBindingEvent e

e, is the event object that describes the binding.

Class

(1) The Cookie Class :

1) The cookie class encapsulates a cookie. A cookie is stored on a client & contains state information.
2) Cookies are valuable for tracking user activities. For example, assum that a user visits online store.
A cookie can save the username address, & other info. The user does n need to enter this data each time he or she visits the store.
3) A servelet can write a cookie to a users machine via the addCookie() method of the HttpServletResponse interface.

4) The data for that cookie is then included in the header of the HTTP response that is sent to the browser.
5) The names and values of cookies are stored on the user's machine. Some of the information that is saved for each cookie includes the following.
(a) The name of the cookie.
(b) The value of the cookie.
(c) The expiration date of the cookie.
(d) The domain & path of the cookie.
6) The expiration date determines when this cookie is deleted from the user's machine. If an expiration date is not explicity assigned to a cookie, it is deleted when the current browser session ends.
7) The domain & path of the cookie determine when it is included in the header of an HTTP request.
If the user enters URL whose domain and path match these values, the cookie is then supplied to the web server. Otherwise, it is not.
(8) There is one constructor for cookie.

Cookie(String name, String value).

Method :
(1) object clone() : Returns a copy of this obj
(2) String getComment() : Returns the comment.

(3) String getDomain(): Returns the domain

(4) int getMaxAge(): Returns the maximum age (in seconds).

(5) String getName(): Return the Name.

(6) String getPath(): Returns the path.

(7) boolean getSecure(): Returns true if the cookie is secure. Otherwise, returns fals

(8) String getValue(): Returns value.

(9) int getVersion(): Returns the version.

(10) void getComment(String c): sets the comment to

(11) void setDomain(String d): sets the domain to d.

(12) void setMaxAge(int secs): sets the maximum a of the cookie to secs. This is the number seconds after which the cookie is delet

(13) void setPath(String p): sets the path to p.

(14) void setSecure(boolean secure): sets the security flag to secure.

(15) void setValue(String v): set the value t

(16) void setVersion(int v): sets the version v.

(2) The HttpServlet class

The HttpServlet class extends GenericServlet. It is commonly used when developing Servlets that receive & process HTTP requests.

Methods :

(1) void doDelete(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException

:⇒ 1) Handles an HTTP Delete request.
2) It allows client to delete a document, webpage or information from the server.

(2) void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException

:⇒ (1) Handles an HTTP GET request.

(3) void doHead(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException

:⇒ (1) Handles an HTTP HEAD request.
(2) The client sends a Head request when it wants to see only headers of a response, such as content-type or content-length.

(4) void doOptions(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException

:⇒ Handles an HTTP OPTIONS request.

(2) Determines which http methods server supports & return an appropriate header.

(5) void doPost (HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException
:- Handles an HTTP POST request.

(6) void doPut (HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException
:) Handles an HTTP PUT request.
:) Similar to post() method unlike post() we send information to the servers, this method sends file to the server.

(7) void doTrace (HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException
:⇒ 1) Handles an HTTP TRACE request.
2) It used for debugging purpose.

(8) long getLastModified (HttpServletRequest req)
:⇒ Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when the requested resource was last modified.

(9) void service (HttpServletRequest req,
HttpServletResponse res)
throws IOException, ServletException

:⇒ Called by the server when an HTTP request arrives for this servlet.
The arguments provide access to the HTTP request & response, respectively.

Hint : doPost, doGet, doPut, doOption, doHead, doTrace, doDelete are also called as Big 7 HTTP Method or doXXX() method.

(3) HttpSessionEvent Class

HttpSessionEvent encapsulates session events.
It extends EventObject and is generated when a change occurs to the session.
It defines this constructor.

HttpSessionEvent(HttpSession session)

Here, session is the source of the event.
It defines one method, getSession(),

HttpSession getSession()
It returns the session in which the event occured.

(4) The HttpSessionBindingEvent class

1) This class extends HttpSessionEvent.
2) It is generated when a listener is bound to or unbound from a value in an HttpSession object.
3) It is also generated when an attribute is bound or unbound.

Constructors:

HttpSessionBindingEvent(HttpSession session, String name)

HttpSessionBindingEvent(HttpSession session, String name, Object val)

Here, session is the source of the event, & name is the name associated with the obj that is being bound or unbound.

Methods:

(A) String getName(): It obtains the name that is being bound or unbound.

(B) HttpSession getSession(): It shown next, obtains the session to which the listener is being bound or unbound.

(C) Object getValue(): It optains the value of the attribute that is being bound or unbound.

Handling HTTP Requests & Response

(1) The HttpServlet class provides specialized methods that handle the various types of HTTP requests.

(2) A servlet developer typically overrides one of these methods. These methods are

doDelete()
doGet()
doOptions()
doPost()
doPut()
doTrace()

(3) Handling HTTP GET Requests.

① Here we will develop a servlet that handles an HTTP GET request.

② The servlet is invoked when a form on a web page is submitted.

③ The example contains two files.

④ A web page is defined in ColorGet.htm & servlet is defined in ColorGetServlet.java

Program :

```
<html>
<body>
<center>
<form name = "Form1"
action = "http://localhost:8080/servlet-example
/Servlet/ColorGetServlet">
<B> Color : <B>
<Select name = "color" size = "1">
```

```
<option value = "Red" > Red </option>
<option value = "Green" > Green </option>
<option value = "Blue" > Blue </option>
</select>
<br> <br>
< input type = Submit value = "submit" >
</form>
</body>
</html>
```

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

public class ColorGetServlet extends HttpServlet
{
  public void doGet (HttpServletRequest req,
    HTTPresServletResponse res)
    throws ServletException , IOException
  {
    String color = req.getParameter ("color");
    res.setContentType ("text/html");
    PrintWriter pw = res.getWriter ();
    pw.println ("<B> The Selected color is: ");
    pw.println (color);
    pw.close ();
  }
}
```

In above program doGet() method is overridden to process any HTTP GET requests that are sent to this Servlet. It uses the getParameter() method of HttpServletRequest to obtain the selection that was made by the user.

(b) Handling HTTP POST Requests

① Here we will develop a servlet that handles an HTTP POST requests.

② The Servlet is invoked when a form on a web page is submitted.

```
<html>
<body>
<center>
<form name = "form1"
  method = "Post"
  action = "http://localhost:8080/Servlets-
      examples/servlet/ColorPostServlet">
<B>Color:</B>
<Select name = "colors" size = "1">
<option value = "Red">Red</option>
<option value = "Green">Green</option>
<option value = "Blue">Blue</option>
</Select>
<br><br>
<Input type = Submit value = "submit">
</form>
</body>
</html>
```

```
public class ColorPostServlet extends HttpServlet
{
public void doPost (HttpServletRequest req,
HttpServletResponse res) throws ServletException,
IOException
{
String color = request.getParameter ("color");
res.setContentType ("text/html");
PrintWriter pw = response.getWriter ();
pw.println ("<B>The Selected color is :");
pw.println (color);
pw.close ();
}
}
```

The doPost () method is overridden to process any HTTP POST requests that are sent to this servlet.

It uses the getParameter () method of HTTPServletRequest to obtain the selection that was made by the user.

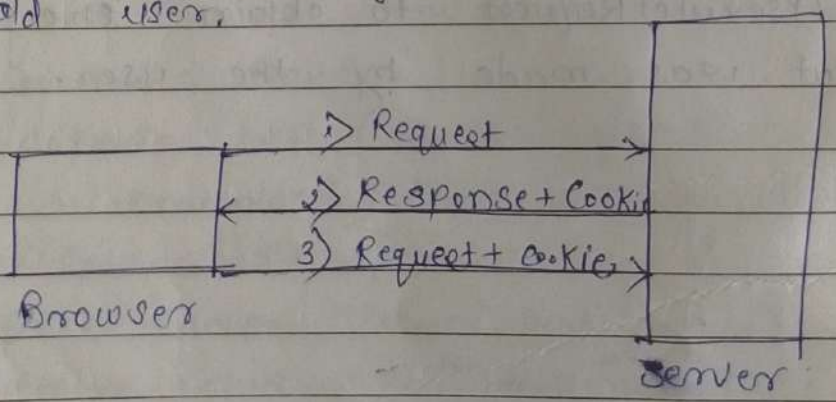
Cookie in Servlet

(1) A cookie is a small piece of information that is persisted between the multiple client requests.

(2) A cookie has a name, a single value, & optional attributes such as a comment, path & domain qualifiers, a maximum age, & a version number.

How cookie works

1) By default, each request is considered as a new request. In Cookie technique, we add cookie with response from the servlet.
2) So cookie is stored in the cache of the browser.
3) After that if request is sent by the user, cookie is added with request by default.
4) Thus, we recognize the user as the old user.



Types of Cookie

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

2) Persistent cookie : It is valid 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 Cookie

① Simplest technique of maintaining the state.
② Cookies are maintained at client side.

Disadvantage of Cookie

① It will not work if cookie is disabled from the browser
② Only textual information can be set in cookie object.

Cookie Class

javax.servlet.http.Cookie class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

Constructor of cookie class

① Cookie() : Constructs a cookie
② Cookie(String name, String value) : constructs a cookie with a specified name & value.

Program

① AddCookie.html

Allows a user to specify a value for the cookie named MyCookie.

```
<html>
<body>
<center>
<form name = "form1"
      method = "post"
      action =
>
<B> Enter a value for MyCookie : </B>
<input type = textbox name = "data" size=25
       value = "">
<input type = submit value = "submit">
</form>
</body>
</html>
```

② AddCookieServlet.java:

processes the submission of AddCookie.html

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

public class AddCookieServlet extends HttpServlet
{
public void doPost(HttpServletRequest req,
HttpServletResponse res) throws
```

```
ServletException, IOException
{
    String data = req.getParameter ("data");

    Cookie cookie = new Cookie ("MyCookie", data);

    res.addCookie (cookie);

    response.setContentType ("text/html");
    PrintWriter pw = res.getWriter();
     pw.println ("<B> My Cookie has been set to");
     pw.println (data);
      pw.close();
}
}
```

(3) GetCookiesServlet.java
display cookie values.

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

public class GetCookieServlet extends HttpServlet
{
    public void doGet (HttpServletRequest req,
    HttpServletResponse res) throws
    ServletException, IOException
{
```

```
Cookie [] cookies = request.getCookies();

res.setContentType("html/text");
PrintWriter pw = res.getWriter();
pw.println("<B>");

for(int i=0; i< cookies.length ;i++)
{
    String name = cookies[i].getName();
    String value = cookies[i].getValue();
    pw.println("name = " + name + ";
                value = " + value);
}

pw.close();
}
}
```

Session Tracking

1) HTTP is a stateless protocol. Each request is independent of the previous one.
2) However, in some applications, it is necessary to save state information so that info can be allocated from several interactions between a browser & a server. Session provide such a mechanism.
3) A session can be created via the getSession() method of HttpServletRequest.
4) An HttpSession object is returned.
5) This object can store a set of bindings that associate names with objects.
6) setAttribute()
   getAttribute()
   getAttributeNames()
   removeAttribute() methods of HttpSession manage these bindings.

program

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

public class DateServlet extends HttpServlet
{
    public void doGet(HttpServletRequest req,
        HttpServletResponse res) throws
        ServletException, IOException
```

```
{
    HttpSession hs = request.getSession(true);

    res.setContentType("text/html");
    PrintWriter pw = res.getWriter();
    pw.print("<B>");

    Date date = (Date)hs.getAttribute("date");
    if (date != null)
    {
      pw.print("Last access : " + date + "<br>");
    }

    date = new Date();
    hs.setAttribute("date", date);
    pw.println("Current date : " + date);
  }
}
```