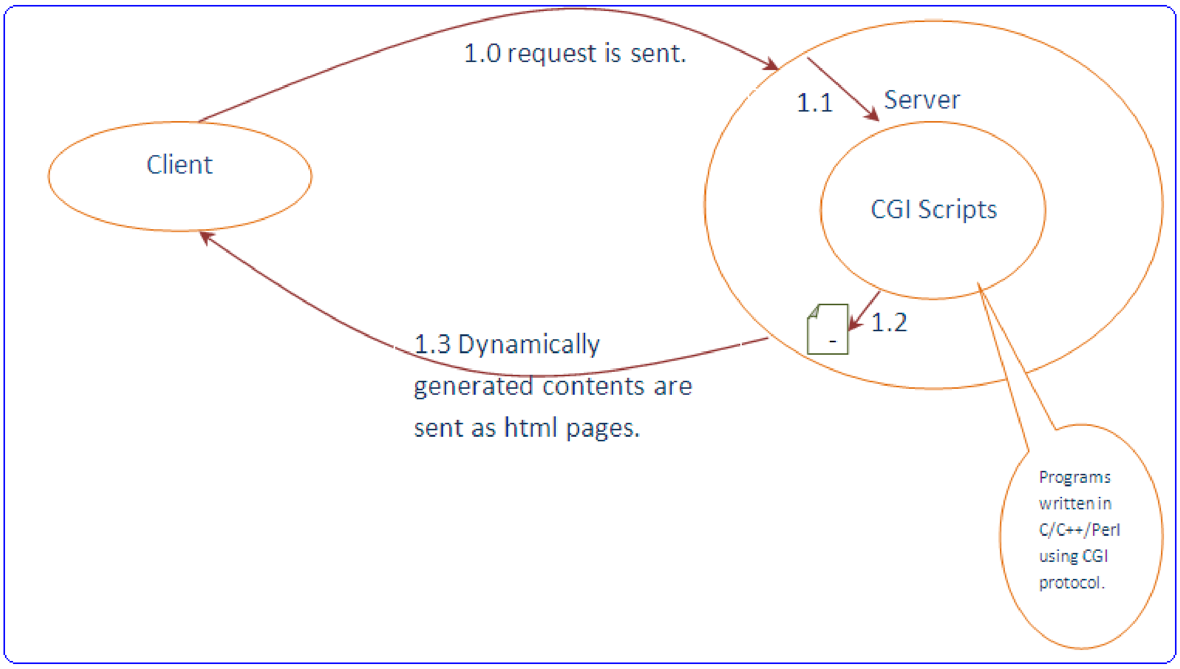
SERVLET

- Servlet is a technology for developing Web Applications in Java.
  Apart from technology, **Servlet** term is also used to describe those programs which are developed using this technology.
  As an Application Component, **Servlet** represents such programs which are executed within a Web Server and generates dynamic html.
  Before Servlets, Dynamic Web Applications used to be developed using CGI.
- CGI is a protocol that defines standard mode of communication between web server and program. In case of CGI based application, programs were written in C, C++ or perl. These programs were called CGI scripts.


1.0 request is sent.
Client
1.1
Server
CGI Scripts
1.2
1.3 Dynamically generated contents are sent as html pages.
Programs written in C/C++/Perl using CGI protocol.

  ✓ *CGI based applications had following two problems-*
    - CGI scripts were platform dependent.
    - CGI scripts were executed as processes which use to create lot of overhead.
- Servlet as *a technology of web development* solved both the problems associated with CGI. *javax.servlet* package & its sub-packages provide *classes & interfaces* that facilitate development of Web Applications. At the core of *Servlet API*, is an *interface javax.servlet.Servlet*. This *interface* defines life cycle methods of Servlets.

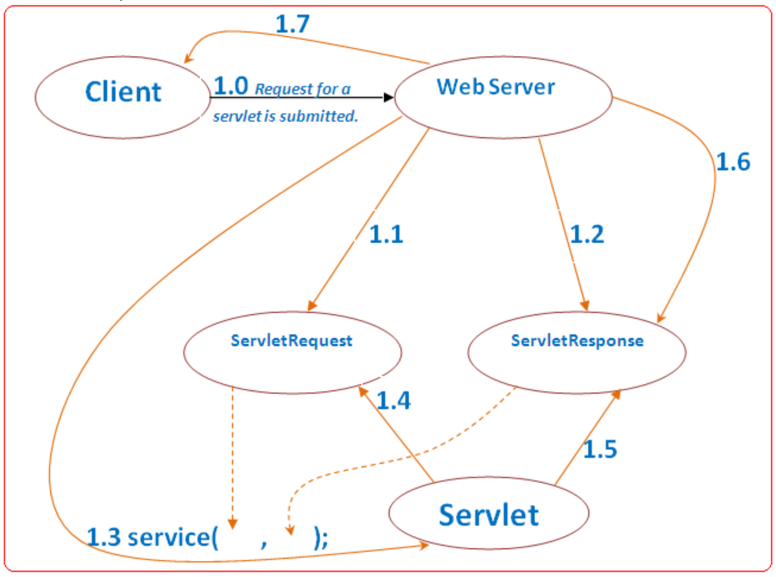
- *Life cycle methods of Servlet-*
  ✓ **init()** method is invoked only once just after a **Servlet** object is created. It is used by the server to provide the reference of an object of type **ServletConfig** to the servlet. **ServletConfig** is an *interface* of **Servlet API,** implementation of which is provided by *Server Vendor*.

  public void init(ServletConfig config);

  ✓ **service()** method is invoked by the server each time a request for the servlet is received. It is used by the Application Developer for processing request.

public void service (ServletRequest request, ServletResponse response) throws ServletException, IOException;

In this method, Web Server provides the reference of objects of type **ServletRequest** & **ServletResponse**. These are interfaces of **Servlet API** which are implemented by Server Vendor.
These two objects act as interface between WebServer and Servlet.



1.1 **ServletRequest** object is created for the request and request data is stored in it.

1.2 **ServletResponse** object is created to receive output of request processing from the servlet.

1.3 **service()** method is invoked and reference of **ServletRequest** and **ServletResponse** objects is provided to servlet.

1.4 Servlet reads request data from **ServletRequest** object.

1.5 Servlet stores dynamically generated HTML contents into **ServletResponse** object.

1.6 After request is processed, Web Server reads the contents of **ServletResponse**.

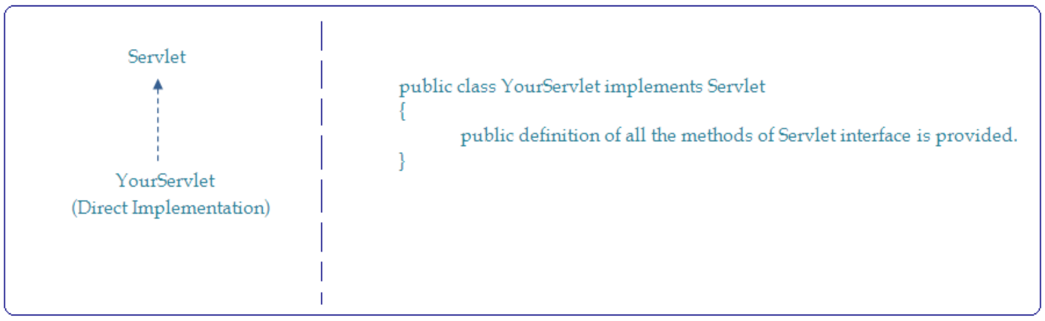
1.7 Contents of **ServletResponse** are sent to the client.

- ✓ **destroy()** method is invoked only once just before a servlet is unloaded.
  public void destroy();

- *Non-life cycle methods of servlet*
  - ✓ **getServletConfig()** method is used to obtain the reference of **ServletConfig** object from the servlet.
    public ServletConfig getServletConfig();

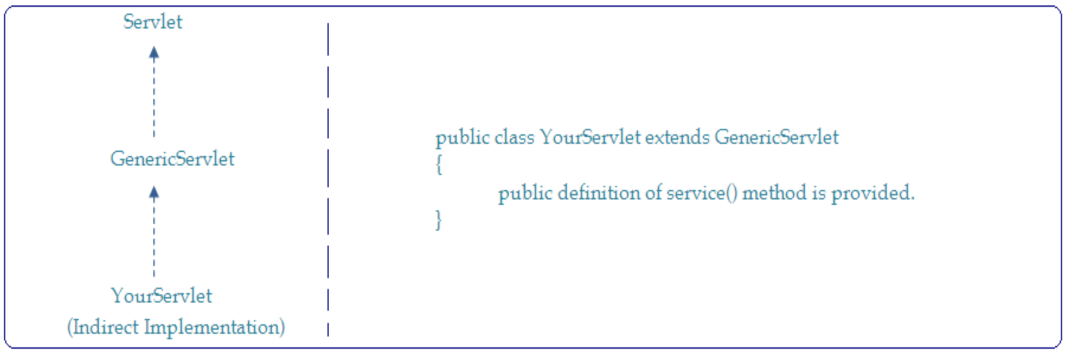
- ✓ **getServletInfo()** method is used by the Application Developer to describe a servlet.
  public String getServletInfo();

- In order to define a *Servlet*, implementation of *Servlet interface* need to be provided.
  - ✓ *It can be done in the following two ways –*



```
public class YourServlet implements Servlet
{
        public definition of all the methods of Servlet interface is provided.
}
```

  - **javax.servlet.GenericServlet** class is an abstract class that implements *Servlet* interface & defines all its methods except **service()** 1method. This class is used as a base class for defining service.



```
public class YourServlet extends GenericServlet
{
        public definition of service() method is provided.
}
```

Note: This method has also some problem.

- In Web Applications, **http** protocol is used for communication with Web Servers.

Http protocol supports following type of requests:

i. Get
ii. Post

Used for requesting data.

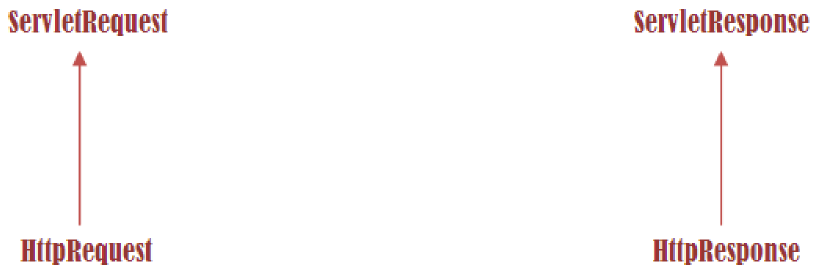
iii. Head
iv. Put
v. Delete
vi. Connect
vii. Trace
viii. etc.

Used for connection management.

- **javax.servlet.http.HttpServlet** class extends **GenericServlet** class and defines additional methods for **http** requests.
  - ✓ Most commonly used methods are **doGet()** & **doPost()** which are used to process **http** get & post requests respectively.

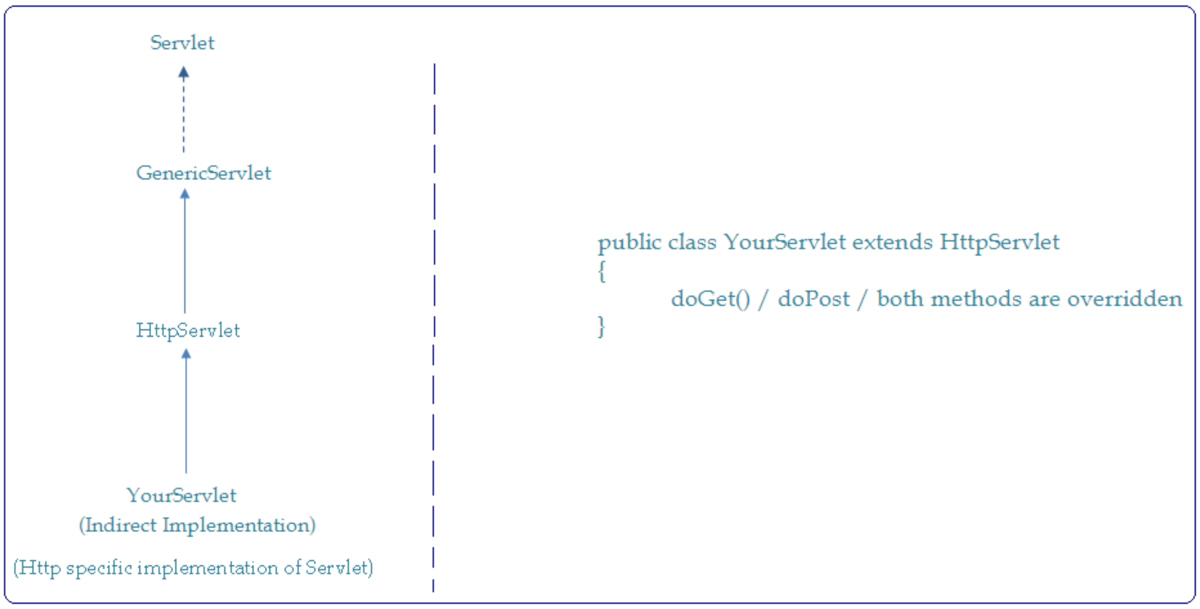
public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException;

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException;



(These interfaces adds Http specific methods)

- *Now we need to create Servlet in the following way:-*


Servlet
GenericServlet
HttpServlet
YourServlet
(Indirect Implementation)
(Http specific implementation of Servlet)

```
public class YourServlet extends HttpServlet
{
        doGet() / doPost / both methods are overridden
}
```

✓ **Difference between Http Get & Post requests:-**

✓ Conventionally, Get request is used for requesting static contents and *Post request is used for requesting dynamic contents.*

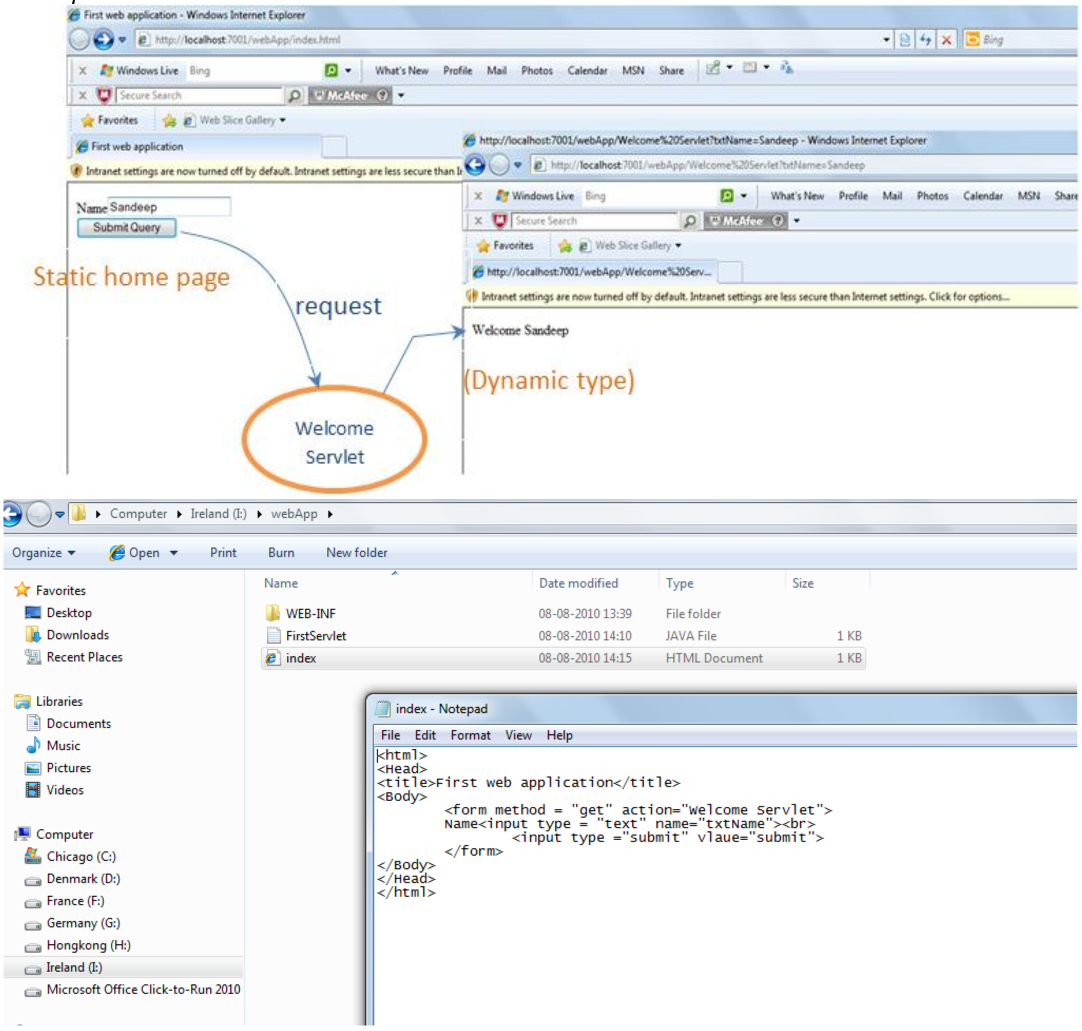
✓ *Technical Difference*

- In case of Get request, request data is sent as part of the header. Size of Http packet header is fixed. Hence, only limited amount of data can be sent as part of Get request.
  In case of Post request, request data is sent as part of body. Size of Http packet body is unlimited. Hence, unlimited amount of data can be sent as part of Post request.

- In case of Get request, data is appended to the URL. Hence, it is visible in the address bar of the browser.
  In case of Post request, request data is not shown in the address bar because it is sent separately in the URL.

- In case of Get request, request data is sent as submitted by the user whereas *in case of Post request, request data is sent in encrypted form.*

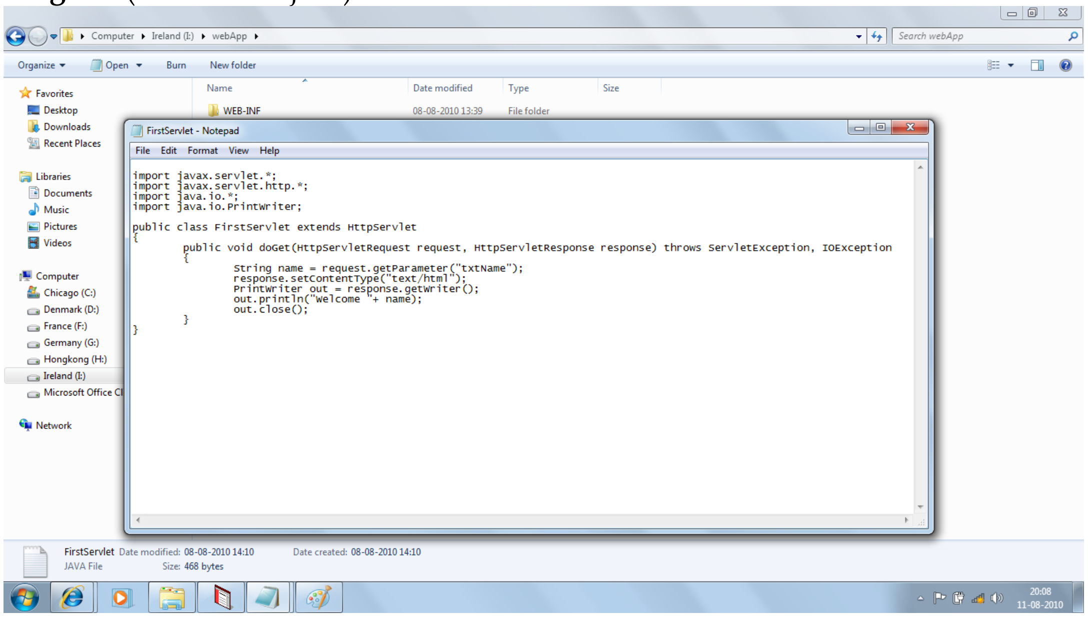
- *Example*:


First web application - Windows Internet Explorer
http://localhost:7001/webApp/index.html
Name Sandeep
Submit Query
Static home page
request
Welcome Servlet
http://localhost:7001/webApp/Welcome%20Servlet?txtName=Sandeep - Windows Internet Explorer
Welcome Sandeep
(Dynamic type)
Computer ▸ Ireland (I:) ▸ webApp ▸
index - Notepad

Name	Date modified	Type	Size
WEB-INF	08-08-2010 13:39	File folder	
FirstServlet	08-08-2010 14:10	JAVA File	1 KB
index	08-08-2010 14:15	HTML Document	1 KB

```
<html>
<Head>
<title>First web application</title>
<Body>
        <form method = "get" action="welcome Servlet">
        Name<input type = "text" name="txtName"><br>
                <input type ="submit" vlaue="submit">
        </form>
</Body>
</Head>
</html>
```

- **Program** (FirstServlet.java)


Computer ▸ Ireland (I:) ▸ webApp ▸
FirstServlet - Notepad
FirstServlet Date modified: 08-08-2010 14:10 Date created: 08-08-2010 14:10
JAVA File Size: 468 bytes

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

public class FirstServlet extends HttpServlet
{
        public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
        {
                String name = request.getParameter("txtName");
                response.setContentType("text/html");
                PrintWriter out = response.getWriter();
                out.println("Welcome "+ name);
                out.close();
        }
}
```

- *Elements of web.xml*
  <web-app> is the root element of web.xml

  <servlet> is a sub element of <web-app> that is used to describe a Servlet in the Web Application.

  It contains following sub elements:-
  - ✓ <servlet-name> is used to specify a unique name for the Servlet.
  - ✓ <servlet-class> is use to specify name of Servlet class.

  <Servlet-mapping> is a sub element of <web-app> that is used to map a request to a servlet.

  It has following sub elements:-
  - ✓ <servlet-name> is used to specify name of Servlet.
  - ✓ <url-pattern> is use to specify url used by the clients to invoke Servlet.

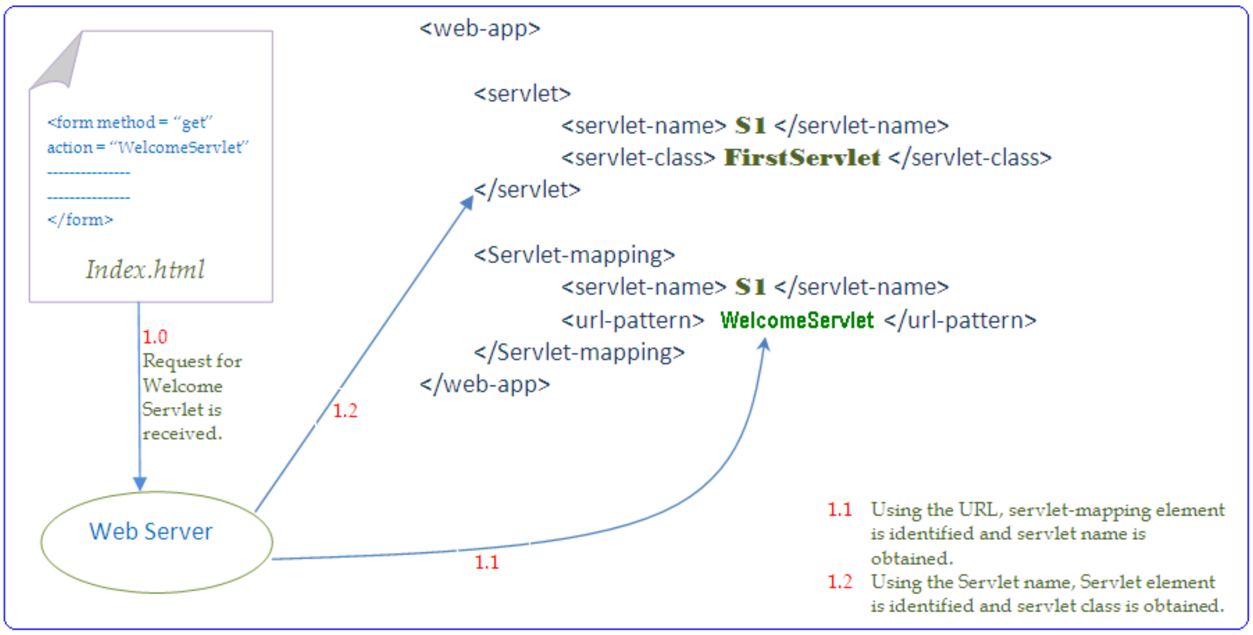
```
<web-app>

        <servlet>
                <servlet-name> Unique Identifier </servlet-name>
                <servlet-class> Fully qualified Servlet class </servlet-class>
        </servlet>

        <Servlet-mapping>
                <servlet-name> Unique Identifier </servlet-name>

                <url-pattern>  Url used to invoke Servlet </url-pattern>
        </Servlet-mapping>
</web-app>
```

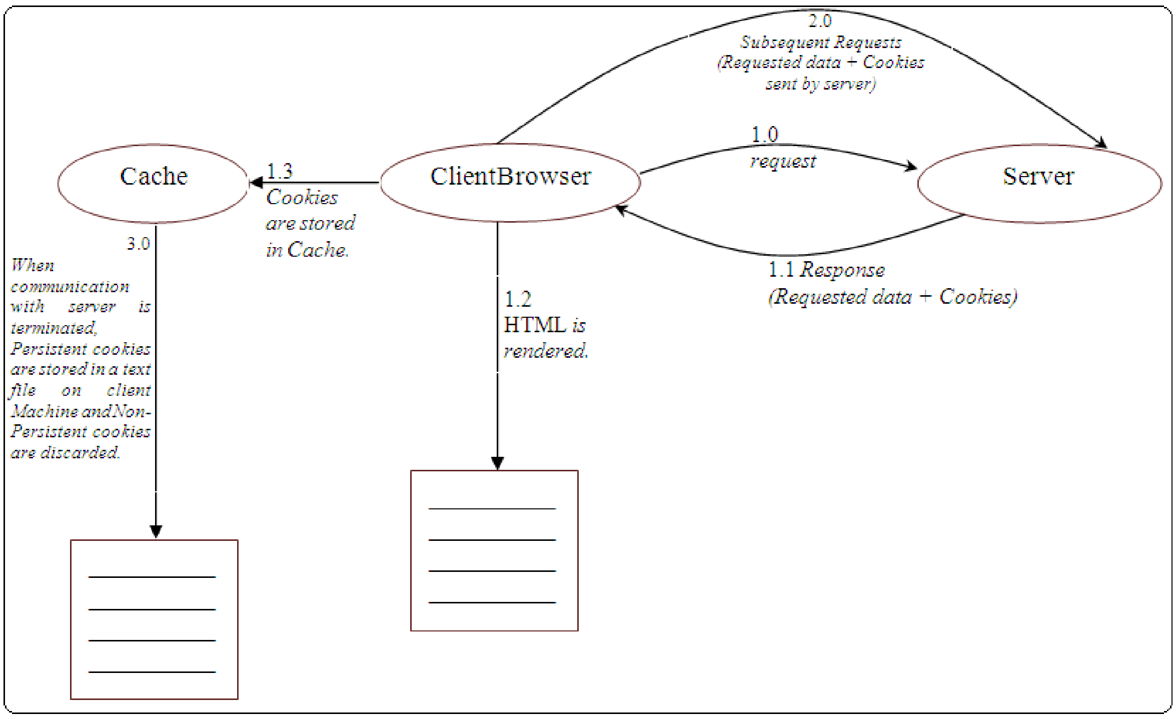
- See the figure below -



<form method = "get"
action = "WelcomeServlet"

</form>
Index.html
1.0
Request for Welcome Servlet is received.
Web Server
1.2
1.1
<web-app>
<servlet>
<servlet-name> S1 </servlet-name>
<servlet-class> FirstServlet </servlet-class>
</servlet>
<Servlet-mapping>
<servlet-name> S1 </servlet-name>
<url-pattern> WelcomeServlet </url-pattern>
</Servlet-mapping>
</web-app>
1.1 Using the URL, servlet-mapping element is identified and servlet name is obtained.
1.2 Using the Servlet name, Servlet element is identified and servlet class is obtained.

- **State Management** – Http is the protocol that is used for communication with Web Servers is a stateless protocol i.e. for each Web request a new Http connection is created which is closed as soon as response of the request is received by the client.
- The disadvantage of the stateless protocol is that server fails to recognize that a series of request coming from a client are logically related & represents a single operation from clients point of view. Problem of State Management deals with maintenance of information between a series of requests.
- *Following four methods are devised to deal with the problem of State Management -*
    - Cookies.
    - Hidden form fields.
    - URL Rewriting.
    - HttpSession.
- **Cookies** – A cookie represents textual information (in the form of key/value pair) that is sent by the server as part of response to the client machine so that it can be sent by the client to the server with subsequent request.
- Cookies provide a simple mechanism of maintaining state between a series of requests.
- *Cookies are of two types-*
    - Non-persistent - Non Persistent cookies remain valid only for a single session whereas Persistent Cookies remain valid for multiple sessions.
    - Persistent - Persistent Cookies are stored in a text file by the browser on the client machine.
    - Persistent Cookies remain valid for multiple sessions. When a session is completed browser storesare stored in a text file by the browser on the client machine.


2.0
Subsequent Requests
(Requested data + Cookies
sent by server)
1.0
request
Cache
1.3
Cookies
are stored
in Cache.
ClientBrowser
Server
3.0
When communication with server is terminated, Persistent cookies are stored in a text file on client Machine and Non-Persistent cookies are discarded.
1.2
HTML is
rendered.
1.1 Response
(Requested data + Cookies)

❖ **Disadvantage of using cookie-**
- This method of state management is Browser dependent.
- Only textual information can be persisted.
- Persistent cookies don't differentiate between user and machine.

- **javax.servlet.http.Cookie** class provides the functionality of cookies. A cookie object can be created using following constructor.

public Cookie (String name, String value);

- *Commonly used methods -*
  1. **getName()** method returns the name of the cookie.
     public String getName();

  2. **getValue()** method returns the value of the cookie.
     public String getValue();

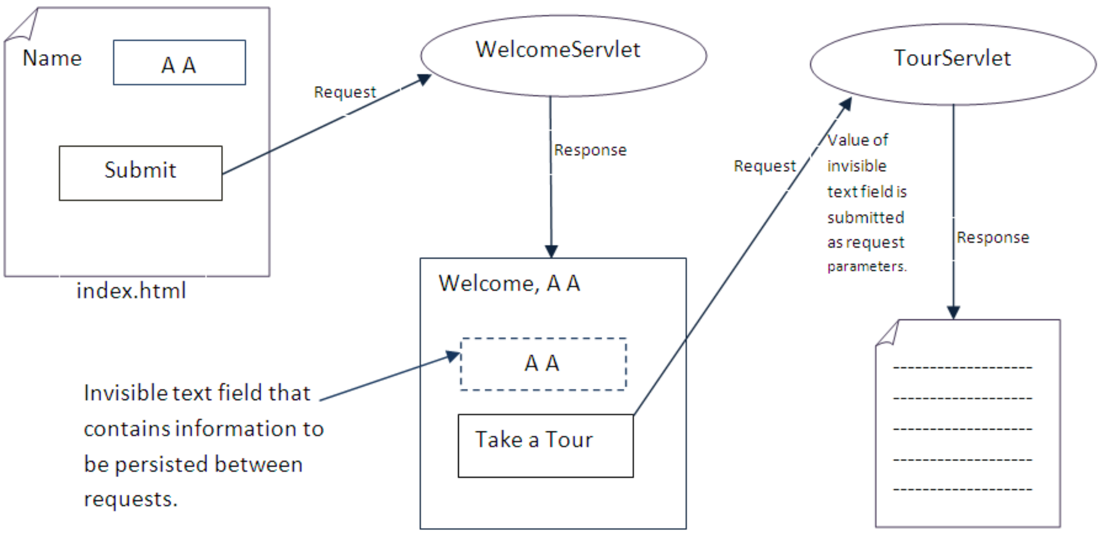
  3. **setMaxAge()** method is used to specify the time in seconds for which a cookie remains valid i.e. this method is used to make a cookie persistent.
     public void setMaxAge(long timeInSeconds);

- **addCookie()** method of **HttpServletResponse** interface is used to send cookies as part of a response.
  public void addCookie(Cookie ck);

- **getCookies()** method of **HttpServletRequest** interface is used to obtain the cookies received as part of a request.
  public Cookie[ ] getCookies();

  - *Disadvantage of this approach-*
    - This method of State Management is browser dependent.
    - Only textual information can be persistent.
    - Persistent cookies don't differentiate between user and machine.

- Second Approach (<u>Hidden form Field</u>)
  - Hidden Form Fields is an alternative method of State Management. This method is browser independent.
  - In this method information to be persisted between request is contained in invisible text fields which are added to the request is submitted from the response page value of invisible text fields is submitted as request parameters.


Name
A A
Submit
index.html
Request
WelcomeServlet
Response
Welcome, A A
A A
Take a Tour
Invisible text field that contains information to be persisted between requests.
Request
Value of invisible text field is submitted as request parameters.
TourServlet
Response

  - *Advantage of this approach-*
    - It is browser independent.
  - *Disadvantage of this approach –*
    - This method can only be used when an input form is used for submitting request.
    - Only textual information can be persisted.
- Example-

- **Third approach (URL Rewriting)**
    - In this approach, information to be persisted between requests is dynamically appended to the URL of pages which are requested from the response page. Whenever a request is submitted using these URL, request parameters are also submitted.
    - *Advantage –*
        - The advantage of this approach is that it can be used with input forms as well as hyperlinks.
    - *Disadvantage –*
        - Only textual information can be persisted.
- *Example-*

- **HttpSession** is an interface of Servlet API implementation of which is provided by server vendor. An object of type **HttpSession** is created by the object can be used by the Application Developer to store user information in the form of attributes between requests.
- *Methods of* **HttpSession** *interface-*

  - **setAttribute()** method is used to store an attribute.
    public void setAttribute(String name, Object value);

  - **getAttribute()** method is used to fetch the value of an attribute.
    public Object getAttribute(String name);

  - **getAttributeNames()** method returns an Enumeration for the name of all attributes.
    public Enumeration getAttributeNames();

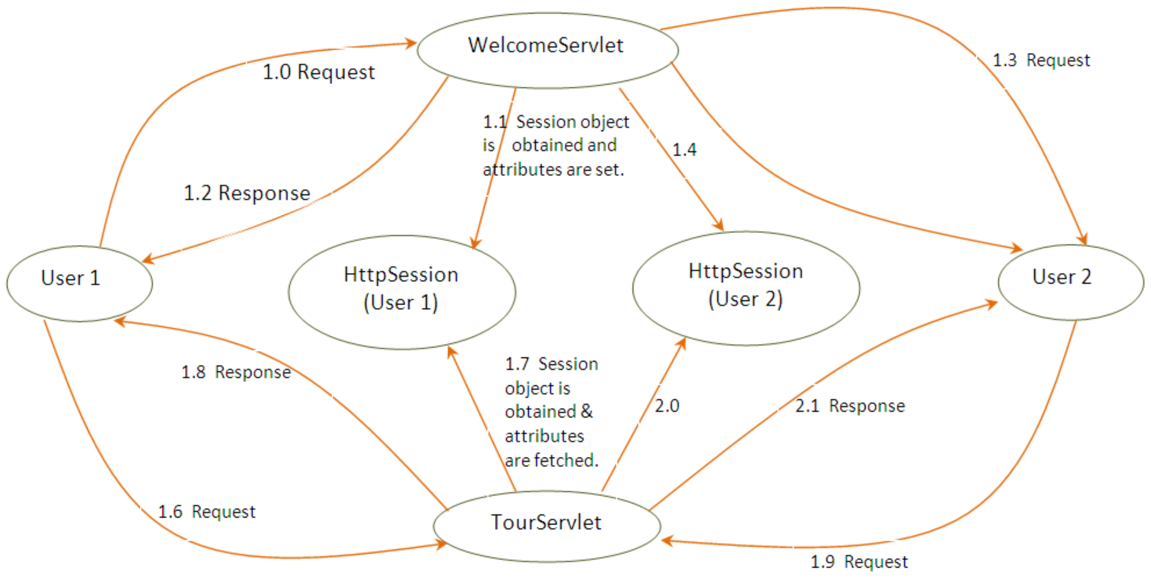
  - **removeAttribute()** method is used to remove an attribute.
    public boolean removeAttribute(String name);

- *Methods to maintain* **Session** *objects-*

  - **isNew()** method returns true if session Object is created in the current request.
    public boolean isNew();

  - **setMaxInactiveInterval()** method is used to specify the time for which a session object is maintained on the server even if a request is not received.
    public void setMaxInactiveInterval(long seconds);

  - **invalidate()** method is used to release the session.
    public void invalidate();
  - **getSession()** method of **HttpServletRequest** interface is used to request a session object.
    public HttpSession getSession();
    public HttpSession getSession(boolean createflag);
- *Example-*


WelcomeServlet
1.0 Request
1.3 Request
1.1 Session object is obtained and attributes are set.
1.4
1.2 Response
User 1
HttpSession (User 1)
HttpSession (User 2)
User 2
1.8 Response
1.7 Session object is obtained & attributes are fetched.
2.0
2.1 Response
1.6 Request
TourServlet
1.9 Request