UNIT-V: WEB SERVERS

5.1Introduction to Servlets:-

History of Web programming

In early days of web, CGI (common gateway interface) is used for managing server side applications.

Drawbacks of CGI:

- Creating separate process for each client request was expensive, in terms of process and memory resources.
- It was also expensive to open and close database connection for each client request.
- CGI programs were not platform independent.

Servlets

- Pure java programs which run at server side of web connection.
- Dynamically extend functionally of web server.

5.1.2 Advantages of Servlet over CGI:-

- Significantly better performance, because of executions in the address space of web server
- Creation of separate process to handle each client request is not necessary.
- Platform independence because of pure java programming.
- Java security manager on the server enforces a set of restrictions to protect resources on a server machine.
- Full functionality of java class libraries is available to servlet.

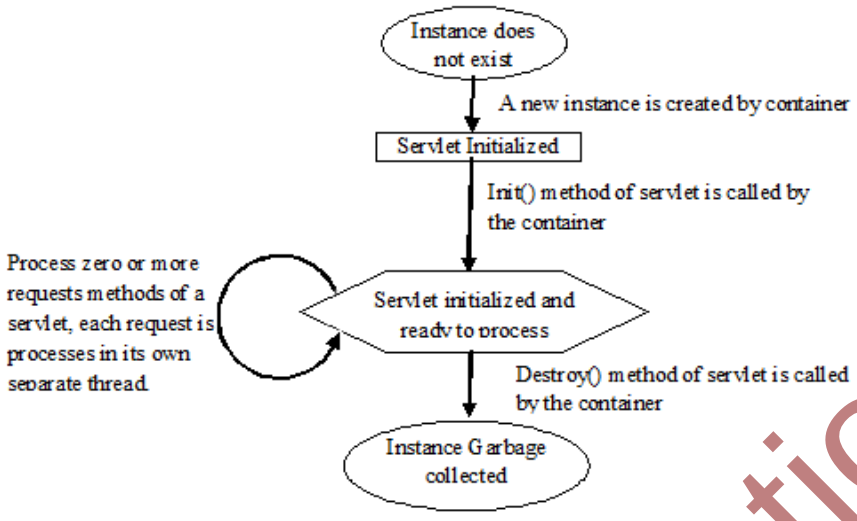
5.2 Life cycle of servlet:-

- Loading
- Initialization
- Ability to handle requests
- Unloading.

Steps

- User enters URL into web browser,
- It generates http request for this URL
- Send it to appropriate server.
- Webserver maps request to particular servlet
- Servlet is loaded into its address space.
- Servlet invokes init() method .
- Servlet invokes service() method
- Process request by reading data form request and send response to client.
- Finally, server invokes destroy() method to unload servlet from its memory.

Life Cycle of Servlet



5.3 JSDK (java server development kit):-

- It contains class libraries belonging to servlet API,ServletRunner to test servlet.
- It can be downloaded fromhttp://java.sun.com web site.
- Windows machine default location is C:\JSDK2.0.
- Directory C:\jsdk2.0\bin contains ServletRunner.exe.
- Directory C:\jsdk2.0\lib contains jsdk.jar .it contains classes and interfaces that are needed to build servlets.

5.4 ServletAPI:-

It consists of packages

(i) javax.servlet.*;

(ii)javax.servlet.http.*;

5.4.1 javax.servlet package contains following interfaces and classes:

It establishes framework in which servlets operate.

Interface	Description
Servlet	Declares lifecycle methods of servlet.
ServletConfig	allows servlet to get intialization parameters.
ServeltContext	Enables servlet to log events,access information about their environment.
ServletRequest	Used to read data form client request.
ServletResponse	Used to write data as client response
ServletThreadModel	Indicates that servlet is thread safe.

Class	Description
GenericServlet	Implements server ,servletConfig interfaces
ServletInputStream	Provides an input stream for reading request from client.
ServletOutputStream	Provides an output stream for writing responses to a client.
ServletException	Indicates servlet error occurred.
UnavailableException	Indicates servlet is permanently or temporarily unavailable

5.4.2 Reading Servlet parameters:-

ServletRequest has methods which reads parameters from client request.

Example program 5.4.2
Reading Servlet Parameters

Step1: Creating web.xml deployment descriptor file.

Web.xml

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
<servlet>
<servlet-name>PostParameterServlet </servlet-name>
<servlet-class>PostParameterServlet </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>PostParameterServlet </servlet-name>
<url-pattern>/postservlet/*</url-pattern>
</servlet-mapping>
</web-app>
```

step2:Creating html file

PostParameters.htm

```
<html>
<body>
<center>
<form name="Form1" method="post"
action="http://localhost:8080/servlet/PostParametersServlet">
<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”>
</form>
</body>
</html>
```

Step3:Create java source file

PostParametersServlet.java

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

public class PostParameterServlet extends GenericServlet {
public void service(ServletRequest req,ServletResponse res) throws
ServletException,IOException {
PrintWriter pw=res.getWriter( );
Enumeration e=req.getParameterNames( );
while(e.hasMoreElements( )){
String pname=(String)e.nextElement( );
pw.print(pname+” = “);
String pvalue=req.getParameter(pname);
pw.println(pvalue);
}
pw.close();
}
}
```

compile servlet and perform these steps to test this example

1.start serveltrunner.

- Display web page in a browser
- Enter an employee name and phone number in text fields.
- Submit web page.

5.4.3 Reading Initialization Parameters:-

Parameters are initialized to open files, create database connections and perform other actions.

Information can be accessed in two ways:

1. Init() method receives servletconfig object as its argument which enables to read initialization parameters.
2. getServletConfig() method declared by servlet interface returns a serveltconfig object.

Initialization parameters are provided to servlet is server dependent.

Example program 5.4.3

Reading Initialization parameters

Step1:Creating web.xml deployment descriptor file.

Web.xml

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE web-app PUBLIC  "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
<servlet>
<servlet-name>InitServlet </servlet-name>
<servlet-class>InitServlet </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>InitServlet </servlet-name>
<url-pattern>/initservlet/*</url-pattern>
</servlet-mapping>
</web-app>
```

step2: intailize parameter values using servelt.properties

servlet.properties

```
servlet.initservlet.code=InitServlet
servlet.initservlet.initArgs=\
country=Canada,\
city=Toronto
```

Step3:Create java source file

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

public class InitServlet extends GenericServlet {
public void service(ServletRequest req,ServletResponse res) throws
ServletException,IOException {
ServletConfig sc=getServletConfig( );
res.setContentType("text/html");
PrintWriter pw=res.getWriter( );
pw.println("<b>Country:"+sc.getInitParameter("country"));
pw.println("<b>City:"+sc.getInitParameter("city"));

pw.close();
}}
```

5.4.4 javax.servlet.http package

It builds servelts that work with HTTP requests and responses.

Interface	Description
HttpServletRequest	Enables servlets to read data from an HTTP request.
HttpServletResponse	Enables servlets to write data to an HTTP response.
HttpSession	Allows session data to be read and written.
HttpSessionBindingListener	Informs an object that is bound to or unbound from a session.
HttpSessionContext	Allows sessions to be managed.

Class	Description
Cookie	Allows state information to be stored on a client machine.
HttpServlet	Provides methods to handle HTTP requests and responses.
HttpSessionBindingEvent	Indicates when a listener is bound to or unbound from a session value.
HttpUtils	Declares utitlity methods for servlets.

5.5 Handling HttpRequests and Responses:-

To handle requests and responses the Httpservlet class provides specialized methods that handle various types of http request.

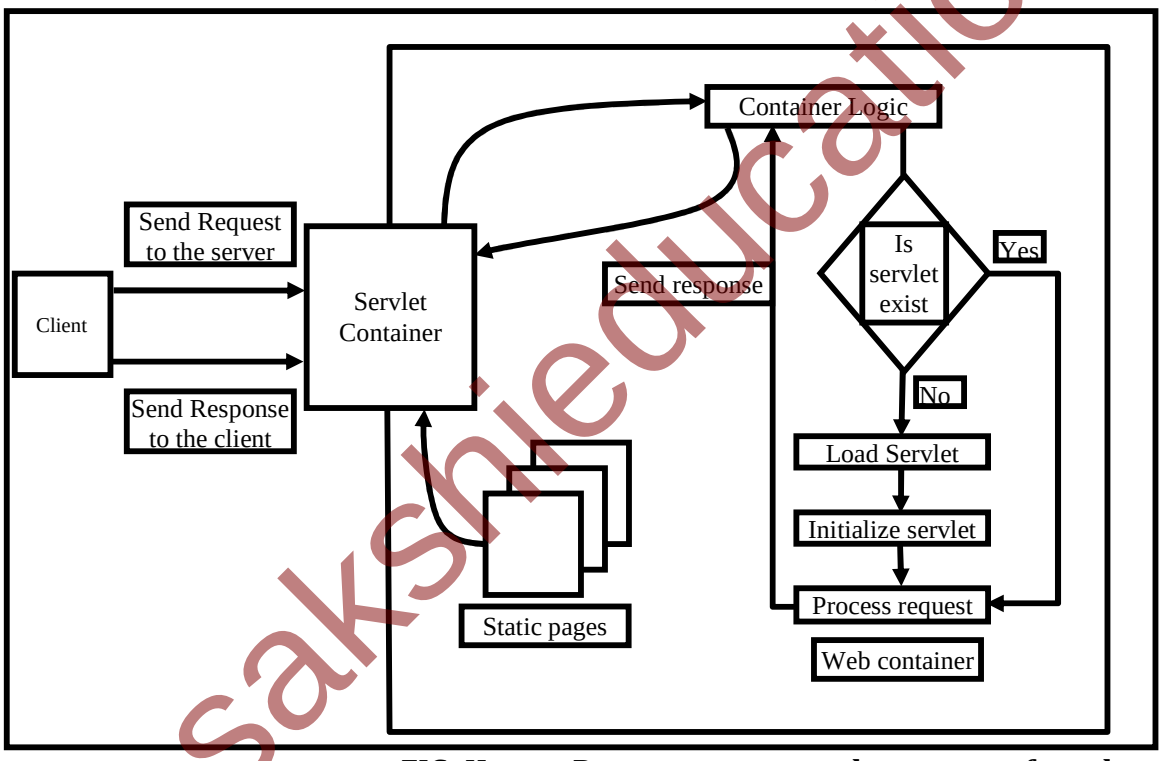


FIG: How to Process request and response of servlet

Example program that develops a servlet that handles an HTTP GET request ex 5.5.1

Step1:Creating web.xml deployment descriptor file.

Web.xml

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE web-app PUBLIC  "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
<servlet>
<servlet-name>ColorGetServlet </servlet-name>
<servlet-class>ColorGetServlet </servlet-class>
</servlet>
```

```
<servlet-mapping>
<servlet-name>ColorGetServlet </servlet-name>
<url-pattern>/colorservlet/*</url-pattern>
</servlet-mapping>
</web-app>
```

step2:Creating html file
ColorGet.html

```
<html>
<body>
<center>
<form name="Form1" method="post"
action="http://localhost:8080/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>
<input type="submit" value="submit">
</form>
</body>
</html>
```

Step3:create java source file

```
import  java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ColorGetServlet extends HttpServlet{
public void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException{
String color=req.getParameter("color");
res.setContentType("text/html");
pw.println("<B> The selected color is:");
pw.println(color);
pw.close();
}
}
```

compile servlet and perform these steps to test this example
1.start serveltrunner.

- Display web page in a browser
- Select color
- Submit web page.

Example program that develops a servlet that handles an HTTP POST request ex 5.5.2

Step1:Creating web.xml deployment descriptor file.
Web.xml

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE web-app PUBLIC  "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
<servlet>
<servlet-name>ColorPostServlet </servlet-name>
<servlet-class>ColorPostServlet </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>ColorPostServlet </servlet-name>
<url-pattern>/colorpservlet/*</url-pattern>
</servlet-mapping>
</web-app>
```

step2:Creating html file
Colorpost.html

```
<html>
<body>
<center>
<form name="Form1" method="post"
action="http://localhost:8080/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>
<input type="submit" value="submit">
</form>
</body>
</html>
```

Step3:create java source file

```
import  java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ColorPostServlet extends HttpServlet{
public void doPost(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException{
String color=req.getParameter("color");
res.setContentType("text/html");
pw.println("<B> The selected color is:");
pw.println(color);
pw.close();
}
}
```

Compile servlet and perform these steps to test this example

1. Start serveltrunner.
   - Display web page in a browser
   - Select color
   - Submit web page.

5.6 Cookie:

- It contains state information for tracking user activities.
- Servlet writes cookie to users machine through addCookie() of HttpServletResponse interface.
- Cookie is included in header of http response that is sent to browser.
- It contains name ,value,expiration date ,domain and path information.

Example program to add cookie and get cookie from servlet ex 5.6.1

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE web-app PUBLIC  "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
<servlet>
<servlet-name>AddCookieServlet </servlet-name>
<servlet-class>AddCookieServlet </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>AddCookieServlet </servlet-name>
<url-pattern>/addcookie/*</url-pattern>
</servlet-mapping>
</web-app>
```

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE web-app PUBLIC  "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
<servlet>
<servlet-name>GetCookieServlet </servlet-name>
<servlet-class>GetCookieServlet </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>GetCookieServlet </servlet-name>
<url-pattern>/getcookie/*</url-pattern>
</servlet-mapping>
</web-app>
```

AddCookie.html

```
<html>
<body>
<form name="form1" method="post"
action="http://localhost:8080/servlet/AddCookieServlet">
<b> Enter a vlaue for MyCookie:</b>
```

```
<input type=textbox name=”data” size=25 value=” “>
<input type=submit value=”submit”>
</form>
</body>
</html>
```

AddCookieServlet.java

```
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 cok=new Cookie(“Mycookie”,data);
res.addCookie(cok);
res.setContentType(“text/html”);
PrintWriter pw=res.getWriter();
pw.println(“<B> My cookie has been sent to:”);
pw.println(data);
pw.close();
}
}
```

GetCookieServlet.java

```
import  java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class GetCookieServlet extends HttpServlet{
public void doPost(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException{
Cookie[] cok=req.getCookies();
res.setContentType(“text/html”);
PrintWriter pw=res.getWriter( );
pw.println(“<B> ”);
for(int i=0;i<cok.length;i++){
String nm=cok[i].getName();
String val=cok[i].getValue();
pw.println(“name=”+nm+”;” value=”+val);
}
pw.close();
}
}
```

5.6 Session Tracking:-

- Http is stateless protocol.
- It maintains the state information for each request .
- HttpSession interface implements session mechanism.
- getSession() of HttpServletRequest creates session.

Example program on date servlet ex 5.6.1

```
<? xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE web-app PUBLIC  "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
<servlet>
<servlet-name>DateServlet </servlet-name>
<servlet-class>DateServlet </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>DateServlet </servlet-name>
<url-pattern>/dateservlet/*</url-pattern>
</servlet-mapping>
</web-app>
```

Java Program

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

public class DateServlet extends HttpServlet
{
public void doPost(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException{
HttpSession hs=req.getSession(true);
res.setContentType("text/html");
PrintWriter pw=res.geWriter( );
pw.println("<B> );
Date dt=(Date)hs.getValue("date");
if(dt !=null){
pw.print("last access :"+date+"<br>");
}
date =new Date();
hs.putValue("date",dt);
     pw.println("current date"+date);
}
 }
```