UNIT V

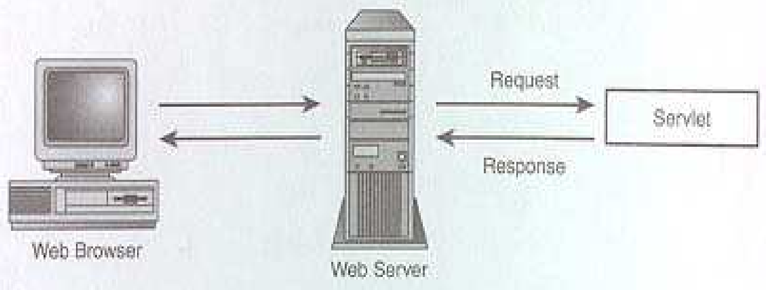
Introduction to Servlets

Servlets are used at server side. When a user request for a web page by entering the URL in the browser. The browser generate HTTP request to the appropriate web server. The web server maps this request with a specific file. The file is returned in the form of HTTP response. To handle these request at server side we require servlet.

There are several advantages of servlet:

1. Performance is significantly better. Servlets execute within the address space of a web server.
2. Servlets are platform independent, they are written in java . Several web servers, from different vendors such as Sun, Microsoft offer servlet API. Java security manager on the server enforces a set of restrictions to protect the resources on a server machine.
3. The full functionality of java class libraries is available to a servlet.
4. Servlets are generic extensions to Java-enabled servers
5. Servlets are secure, portable, and easy to use replacement for CGI
6. ***Servlet is a dynamically loaded module that services requests from a Web server***
7. Servlets are executed within the Java Virtual Machine
8. Because the servlet is running on the server side, it does not depend on browser compatibility

FIGURE 1.1
Execution of a Java servlet.



****

Life cycle of a servlet

Init()

1. Executed **once** when the servlet is first loaded.
2. Not called for each request.
3. Perform any set-up in this method.

• **Service()**

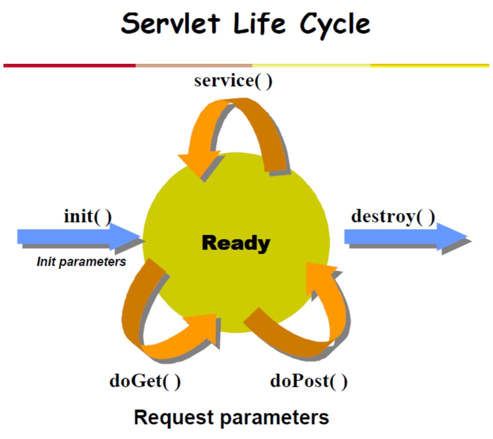
1. Called in a **new thread** by server for each request.
2. Dispatches to doGet, doPost, etc.
3. Do not override this method!

• **doGet(), doPost(), doXxx()**

1. Handles GET, POST, etc. requests.
2. Override these to provide desired behavior.

• **Destroy()**

1. Called when server deletes servlet instance.
2. Perform any clean-up, and save data to be read by the next inti()



Example Program:

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class LifeCycle extends GenericServlet
{
	public void init(ServletConfig config)throws ServletException
	{
		System.out.println("init");
	}
	public void service(ServletRequest req,ServletResponse res)throws
ServletException,IOException
	{
		System.out.println("from service");
		PrintWriter out=res.getWriter();
		out.println("LifeCycle\n");
		out.println("IV CSE Students");
	}
	public void destroy()
	{
		System.out.println("destroy");
	}
}
```

Java Servlet Development Kit

The Java Servlet Development Kit contains the class libraries that you will need to create servlet. JSDK is available from the Sun Microsystems web site at java.sun.com.

A Simple Servlet
HelloServlet.java

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

public class HelloServlet extends GenericServlet
{
public void service(ServletRequest req, ServletRespone res) throws ServletException,
IOException
{
res.setCcontentType(“text/html”); printerWriter pw = res.getWriter( );
pw.println(“Hello”);
pw.close( );
}
}
```

Servlet API

There are two packages that are required to build servlets i.e., javax.servlet and javax.servlet.http. These packages constitute servlet API.

javax.servlet package:

There are number of interfaces and classes present in this package, they are described below.

Interface	**Description**
Servlet	Declares life cycle methods for a servlet
ServletConfig	Allows servlets to get initialization parameters
ServletContext	Enables servlets to log events
ServletRequest	Used to read data from a client request
ServletResponse	Used to read data to a client response

Class	**Description**
GenericServlet	Implements the Servlet and ServletConfig
ServletInputStream	Provides an input stream for reading requests from a client
ServletOutputStream	Provides an output stream for writing responses to a client.
ServletException	Indicates that a servlet error occurred.

Following are the interfaces and their methods

Servlet Interface

void destroy	Called when the servlet is unloaded
ServletConfig getServletConfig()	Returns a ServletConfig object that contains any initialization parameters
String getServletInfo	Returns a string describing the servlet
void init()	Called when the servlet is initialized
void service	Called to process a request from a client

ServletConfig Interface

ServletContext getServletContext	Returns the context for this servlet
String getInitParmeter(String param)	Returns the value of the initialization parameter name param
getInitParameterNames()	Returns all initialization parameter names

ServletContext interface

getAtrribute(String attr)	Returns the value of the server attribute named attr.
String getServiceInfo()	Returns information about the server.
Servlet getServlet(String sname)	Returns the servlet named sname.
getServletNames()	Returns the names of servlets in the server

ServletRequest Interface

String getParameter(String pname)	Returns the value of the parameter named pname
getParameterNames()	Returns the parameter names for this request
String[ ] getParameterValues()	Returns the parameter values for this request
String getProtocol()	Returns a description of the protocol
String getServerName()	Returns the name of the server
Int getServerPort()	Returns the port number.

ServletResponse Interface

PrinterWriter getWriter()	Returns a PrintWriter that can be used to write character data to the response
ServletOutputStream getOutputStream()	Returns a ServletOutputStream that can be used to write binary data to the response

Following are the classes and their methods

GenericServlet class

This class implements Servlet and ServletConfig interfaces

ServletInputStream class

The ServletInputStream class extends InputStream. It is implemented by the server and provides an input stream that a servlet developer can use to read the data from a client request. In addition to this, one more method is added which returns the actual number of bytes read

Int readLine(byte[ ] buffer, int offset, int size)

ServletOutputStream class

ServletOutputStream class extends OutputStream. It defines the print() and println() methods, which output data to the stream.

ServletException class

This class indicates that a servlet problem has occurred. The class has the following contructor

ServletException( )
ServletException(String s)

Reading Servlet Parameters

Example 20: Servlet program showing reading servlet parameters

```
Import java.io.*; Import
java.servlet.*;
Public class ParameterServlet 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(); } }
```

Reading Initialization parameters

Example 21: Servlet program showing reading initialization parameters

```
import java.io.*;
import javax.servlet.*;
public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException
{
ServletConfig sc= getServletConfig();
res.setContentType("text/html"); PrintWriter
pw=res.getWriter();
pw.println(" Name : "+ sc.getInitParameter("name"));
pw.close();
 }
 }
```

javax.servlet.http package

There are number of classes and interfaces present in this package, they are as follows:

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
HttpSessionContext	Allows sessions to be managed

Following are the interfaces and their methods description

HttpServletRequest Interface

Cookie[ ] getCookies	Returns an array of the cookies in this request
String getMethod()	Returns the HTTP method for this request
String getQueryString()	Returns any query string in the URL
String getRemoteUser()	Returns the name of the user who issued this request.
String getRequestedSessionId()	Returns the ID of the session
String getServletPath()	Returns the part of the URL that identifies the servlet

HttpServletResponse Interface

Void addCookie(Cookie cookie)	Adds cookie to the HTTP response.
Void sendError(int c)	Send the error code c to the client
Void sendError(int c , String s)	Send the error code c and the message s
Void sendRedirect(String url)	Redirects the client to url

Cookieclass

The Cookie class encapsulates a cookie. A cookie is stored on a client and contains state information. Cookies are valuable for tracking user activities. A servlet can write a cookie to a user's machine via the addCookie() method of the HttpServletResponse interface. The names and values of cookies are stored on the user's machine. Some of the information that is used saved includes the cookie's

Name Value
Expiration date
Domain and path

Following are the methods that are used by the Cookie class

String getComment()	Returns the comment
String getDomain()	Returns the domain
Int getMaxAge()	Returns the age
String getName()	Returns the name
String getPath()	Returns the path
Boolean getSecure()	Returns true if the cookie is secure
Int getVersion()	Returns the version
Void setComment(String c)	Sets the comment to c
Void setDomain(String d)	Sets the domain to d
Void setPath(String p)	Sets the path to p
Void setSecure(boolean secure)	Sets the security flag to secure

HttpServlet Class

The HttpServlet class extends GenericServlet. It is commonly used when developing servlets that receive and process HTTP requests. Following are the methods used by HttpServlet class

Void doGet(HttpServletRequest req, HttpServletResponse res)	Performs an HTTP GET
Void doPost(HSR req, HSR res)	Performs and HTTP POST
Void service(HSR req, HSR res)	Called by the server when and HTTP request arrives for this servlet.

Example 22: Servlet program handling HTTP GET requests import

```
java.io.*;
import javx.servlet.*;
import javax.servlet.http.*;
 public class GetServlet extends HttpServlet
{
public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
{
String name=req,getParameter(“name”);
res.setContentType(“text/html”); PrintWriter
pw=res.getWriter(); pw.println(“The Name is “);
pw.println(name);
pw.close();
}
```

Note: Same program you can use for Handling HTTP POST requests instead of using doGet we can use doPost

Session Tracking Methods

A session is a conversation between the server and a client. A conversation consists series of continuous request and response

Why should a session be maintained?

When there is a series of continuous request and response from a same client to a server, the server cannot identify from which client it is getting requests. Because HTTP is a stateless protocol.

When there is a need to maintain the conversational state, session tracking is needed. For example, in a shopping cart application a client keeps on adding items into his cart using multiple requests. When every request is made, the server should identify in which client's cart the item is to be added. So in this scenario, there is a certain need for session tracking.
Solution is, when a client makes a request it should introduce itself by providing unique identifier every time. There are five different methods to achieve this.

Session tracking methods:

1. User authorization
2. Hidden fields
3. URL rewriting
4. Cookies
5. Session tracking API

`getAttribute(java.lang.String name)` Returns the object bound with the specified name in this session, or `null` if no object is bound under the name.
`getAttributeNames()` Returns an `Enumeration` of `String` objects containing the names of all the objects bound to this session.
`getCreationTime()` Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
`getId()` Returns a string containing the unique identifier assigned to this session.
`getLastAccessedTime()` Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT, and marked by the time the container received the request.
`getMaxInactiveInterval()` Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.
`getServletContext()` Returns the ServletContext to which this session belongs.
`getSessionContext()` **Deprecated.** *As of Version 2.1, this method is deprecated and has no replacement. It will be removed in a future version of the Java Servlet API.*
`getValue(java.lang.String name)` **Deprecated.** *As of Version 2.2, this method is replaced by* `getAttribute(java.lang.String)`.
`getValueNames()` **Deprecated.** *As of Version 2.2, this method is replaced by* `getAttributeNames()`
`invalidate()` Invalidates this session then unbinds any objects bound to it.
`isNew()` Returns `true` if the client does not yet know about the session or if the client chooses not to join the session.
`putValue(java.lang.String name, java.lang.Object value)` **Deprecated.** *As of Version 2.2, this method is replaced by* `setAttribute(java.lang.String, java.lang.Object)`

`removeAttribute(java.lang.String name)` Removes the object bound with the specified name from this session.
`removeValue(java.lang.String name)` **Deprecated.** *As of Version 2.2, this method is replaced by* `removeAttribute(java.lang.String)`
`setAttribute(java.lang.String name, java.lang.Object value)` Binds an object to this session, using the name specified.
`setMaxInactiveInterval(int interval)` Specifies the time, in seconds, between client requests before the servlet container will invalidate this session.

Example Program

```
import java.io.*;
import javax.servlet.*;
import java.util.*;
import javax.servlet.http.*;
public class Sessiondemo extends HttpServlet
{
        public void doGet(HttpServletRequest req,HttpServletResponse res)throws
IOException,ServletException
        {
                res.setContentType("text/html");
                HttpSession session=req.getSession();
                String heading;
                Integer cnt=(Integer)session.getAttribute("cnt");
                if(cnt==null)
                {
                        cnt=new Integer(0);
                        heading="welcome first time";
                }
                else
                {
                        heading="welcome once again";
                        cnt=new Integer(cnt.intValue()+1);
                }
                session.setAttribute("cnt",cnt);
                PrintWriter out=res.getWriter();
                out.println("<html>");

out.println("<body>");
out.println("<h1>"+heading);
out.println("the number of previous access="+cnt);
out.println("</body>");
out.println("</html>");
}
}
```

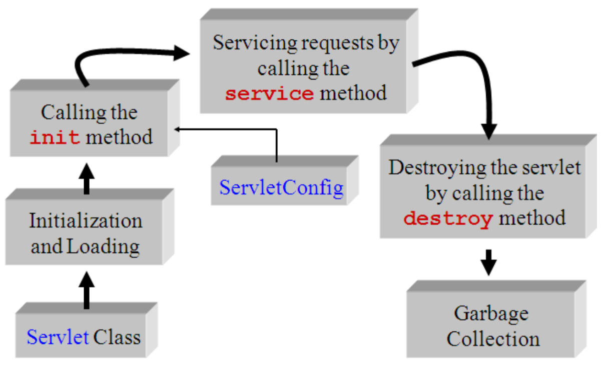
Example Servlet Program:

```
<html>
        <head>
        <title>The servlet example </title>
        </head>
        <body>
                <h1>A simple web application</h1>
                <form  action="HelloWorld" method="get">
                        Enter your name
                        <input type="text" id="name" name="name"/><br><br>
                        <input type="submit" value="Submit Form"/>
                        <input type="reset" value="Reset Form"/>
                </form>
        </body>
</html>
****************************************
<web-app>

<servlet>
     <servlet-name>HelloWorld</servlet-name>
     <servlet-class>HelloWorld</servlet-class>
</servlet>

<servlet-mapping>
     <servlet-name>HelloWorld</servlet-name>
     <url-pattern>/HelloWorld</url-pattern>
</servlet-mapping>

</web-app>
**********************************************
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

     public void doGet ( HttpServletRequest request, HttpServletResponse response )
     throws ServletException, IOException    {
                          String name=request.getParameter("name");
                    response.setContentType("text/html");
          PrintWriter out = response.getWriter();
          out.println("<html>");
          out.println("<head><title>Hello, Cruel World!</title></head>");
          out.println("<body>");
          out.println("<h1>"+name+"<br>Hello, Cruel World !</h1>");
          out.println("This is my first servlet.");
          out.println("</body>");
     }// end doGet

}///
```

Servlets most common usages:

- 1. Used to extend Web servers
- 2. Used as replacement for CGI that is
    - secure, portable, and easy-to-use
- A servlet is a dynamically loaded module that services requests from a Web server
- A servlet runs entirely inside the Java Virtual Machine

Servlet Life Cycle

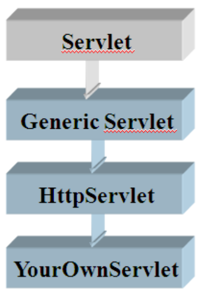

Servicing requests by calling the service method
Calling the init method
ServletConfig
Destroying the servlet by calling the destroy method
Initialization and Loading
Servlet Class
Garbage Collection

HTTP Methods

- POST:
    - Data sent in two steps
    - Designed for Posting information
        - Browser contacts server
        - Sends data
- GET:
    - Contacts server and sends data in single step
    - Appends data to action URL separated by question mark

Designed to get information

Architecture



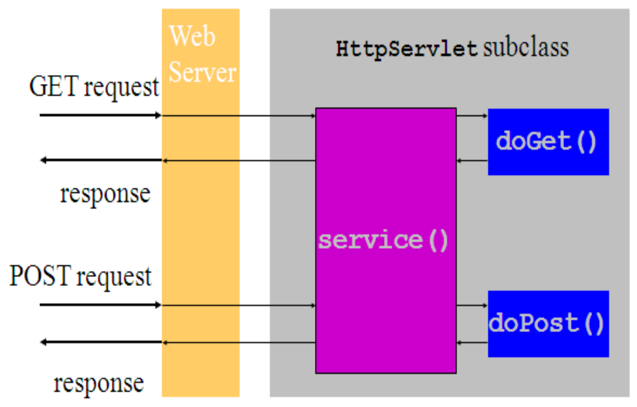
Other Http Methods

- HEAD: Client sees only header of response to determine size, etc…
- PUT: Place documents directly on server
- DELETE: Opposite of PUT
- TRACE: Debugging aid returns to client contents of its request
- OPTIONS: what options available on server

Servicing a Servlet

1. Every call to the servlet creates a new thread that calls the service method
2. The service methods check the type of request (GET, POST, PUT, DELETE, TRACE, OPTION) and call the appropriate method: doGet, doPost, doPut, doDelete, DoTrace, doOption
3. It is recommended to implement doPost and doGet instead of implementing service. Why?
4. The method doPost can call doGet in order to reuse code

`HttpServlet` Request Handling


Web Server
HttpServlet subclass
GET request
response
POST request
response
service()
doGet()
doPost()