UNIT - II

CHAPTER 3

Servlets

Syllabus Topics

Introduction, Web application Architecture, Http Protocol & Http Methods, Web Server & Web Container, Servlet Interface, GenericServlet, HttpServlet, Servlet Life Cycle, ServletConfig, ServletContext, Servlet Communication, Session Tracking Mechanisms.

Syllabus Topic : Introduction

3.1 Introduction

Need for Dynamic Content

Q. Discuss the need of Dynamic Content.

- During the initial stage of web developments in World Wide Web (WWW) web pages were generally designed as static pages. A client requests a resource such as a web page or a file and server return back the static content back to the client. As business domain start using web applications, for enterprise activities server require to send the dynamic data back to the client like
- Displaying the bank account details for a particular customer.
- Performing the bank transactions.
- Online booking of tickets like railways reservation, movie ticket booking etc.
- Weather reporting etc.

3.1.1 Common Gateway Interface (CGI)

Q. Write a short note on CGI. Describe its advantages and disadvantages.

- Common Gateway Interface (CGI) is the initial and oldest method of server side programming techniques that server side programs can access the database servers for creating dynamic content on the Internet. CGI programs get the input from the user using http protocol, access the database, fetch the result from the table storage and return dynamic result back to the Web browser.

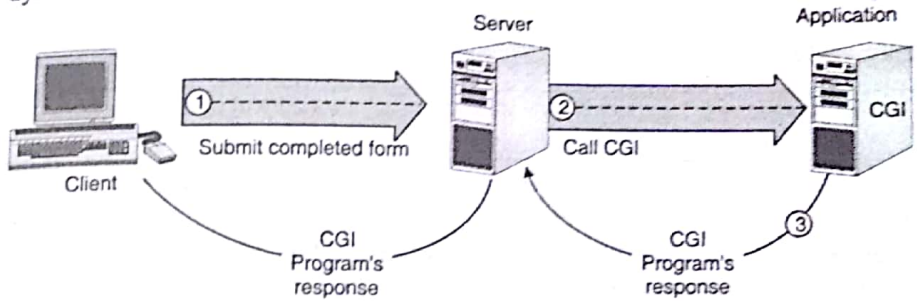


Fig. 3.1.1 : Format of a CGI Program

Advantages of CGI

1. **Different programming languages :** CGI script can be written using multiple programming languages and techniques, that generates the dynamic web pages.
2. **Can determine the Browser :** CGI programs can be written to retrieve the various information about the browser which currently used to access the application.

   Example : If different web browsers like Netscape Navigator or Microsoft Internet Explorer are used to access the same CGI application, it can identify the various features of the browser and return the dynamic content separately designed for the browser which is currently used.
3. **Display dynamic content :** A CGI program return the dynamic content back to the client. If the user submit an HTML form to the Web server :
   - CGI script obtains the data sent from the client.
   - Processes the data by accessing database table as and when needed to produce the result.
   - Display the dynamic result back to the standard output. Data written on the standard output by the CGI script will be visible to the client.

Disadvantage of CGI

1. **Lack of scalability and reduced speed :** Every time a request is received by the web server, a separate process thread is created to handle the same.
2. CGI script is platform dependent.
3. CGI process thread consumes more server side resources than any other server side scripting technique.

4. **It does not support program modularity :** Large business or enterprise applications cannot be designed if the language does not support the concept of modularity.
5. As resources cannot be shared like a database connection between scripts or multiple calls to the same script is not available, same routine will get executed multiple times that leads to the reduced efficiency.

3.1.2 Servlet

Q. What are servlets ? Explain the request/response paradigm of servlets.
Q. Write a short note on Servlet with its general format.

- **Servlet** Technology is used to create web applications. **Servlet** technology uses Java language to create web applications.
- A servlet is a server side scripting techniques supported by java programming language. Servlet is a java class file that can derive the features of web servers which host applications and can be accessed through request-response programming model.
- There are two packages named javax.servlet and javax.servlet.http that can be used for implementing a servlet application. These packages consist of interfaces and classes for servlet API.
- Every servlet implements the Servlet interface that defines Servlet life-cycle methods initialization, servicing and destruction.
- GenericServlet class can be used to implement a generic service() method to process the client request.
- HttpServlet class can be used to implement the HTTP-specific services using either doGet() or doPost() method.

Servlets and Web Servers

Servlets can extend the features of a Java enabled web server. Web servers can use the servlet technology to provide features as :

1. Access the database table.
2. Retrieve the table data and produce the dynamic result and send back to the client browser.

Servlet Clients

Servlet client is an application that receives the output of a servlet program. Servlet client can be designed in any language even other than Java Servlets act as the middle tier of a distributed application.

Servlet Container

The Servlet container or servlet engine provides the runtime environment in which a servlet executes. It manages the servlet lifecycle.

Servlets – Request/Response Paradigm

Servlets are based on request-response paradigm where :

1. A web browser or Client sends the request to the web server.
2. Web server forward the request to a servlet.
3. Servlet accept the request send by the client, access the database table to retrieve the dynamic data and send the response back.
4. Through web server data is send back to the client.

The general format of a Servlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class servlet1 extends GenericServlet
{
   public void service(ServletRequestreq, ServletResponse res) throws ServletException,
   IOException
   {
         :
         :
   }
         :
}
```

The service() method of a generic servlet takes two parameters :

1. **Request** - used to send data from client to the server.
2. **Response** - used to send reply back to the client.

Syllabus Topic : Web Application Architecture

3.2 Web Application Architecture

Q. Write a short note on Web Application Model.

- Web application architecture defines the communications between applications, middleware systems and databases. It ensures multiple applications can work together.
- When a user types in a URL and taps "Search," the browser will findthe page and fulfills the request. The server always responds by sending files over to the browser.
- Thereafter, the browser executes those files to show the requested page to the user. Now, the user gets access to website. This all happens in fractions of seconds.

- The code, which has been parsed by the browser is important. Web applica architecture is designed to work efficiently. It deals with scale, efficiency, robustness, a security.
- Majority of global network traffic, every single app and device uses web-ba communication makes web application architecture more serious.

☞ **Working**

- With web applications, we have the server side program and the client side program b runs concurrently.
- Client side program is -the code which lives in the browser and responds to user inp And sever side program is- the code which lives on the server and responds to HT requests.

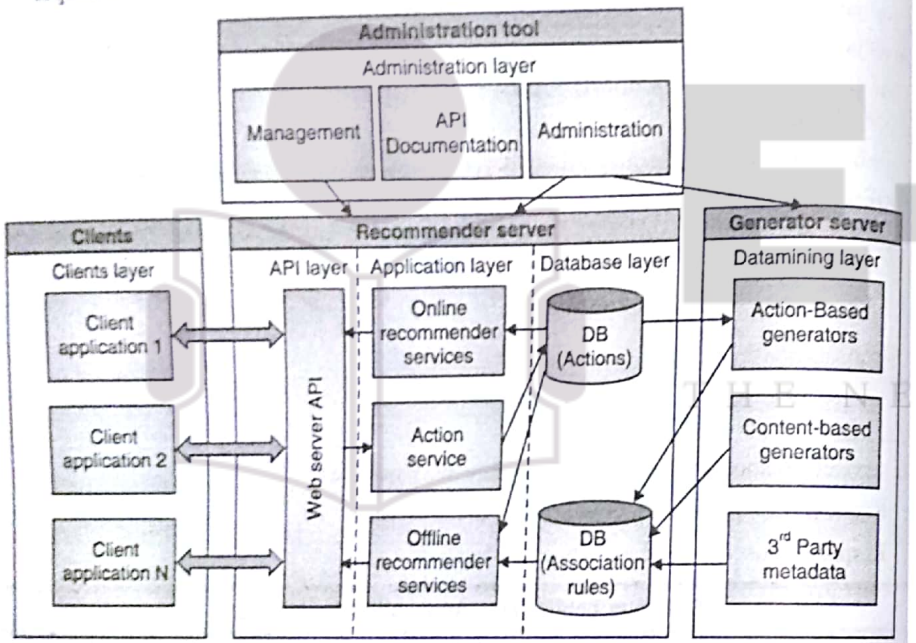


Fig. 3.2.1

- It is up to the programmer to decide what the code on the server should do in relation t user's request. Server-side programs can be written in: Java, Python, Ruby on Rails, PHP C#, JavaScript language. This server-side code is never seen by the user. It creates th page the user requested and it also stores data such as user profiles, pages, etc…
- Client-side programs can be written in : HTML,CSS, JavaScript. This code is then parse by the user browser. Client-side code can be viewed and modified by the user.

3.3 Advantages of Servlets

Q. Discuss the advantages of Servlet.

1. It is an object oriented programming methodology to HTTP protocol communication, where servlet acts as the middle tier of web applications.
2. Support multi-thread model; where each request is handled by a new lightweight thread not by individual process.
3. Servlet applications are portable across multiple web servers and platforms.
4. Applications run within Java Virtual Machine (JVM). Thus reliability and security is achieved.
5. Servlet applications are scalable that helps to develop real time business application.
6. It is supported by numerous web Servers like Apache, Web Logic, GlassFish etc.
7. Native and ODBC based database drivers can be used with servlet techniques to access the database directly.
8. Resources can be shared and allows the synchronized execution of program sections, which is not supported by CGI script.
9. Servlet technique can use all the enterprise Java API's like JNDI, JDBC, RMI, Enterprise JaveBeans etc.
10. Provides the state management technique using Cookies, Sessions or Java Beans for stateless HTTP protocol communication.
11. Requests can be forwarded to other servers or servlets, for load balancing in the server farms.

3.4 Servlet API

Q. Write a short note on Servlet API.

- Servlet Application Programming Interface (API) consists of library of classes and interfaces for designing and implementing the servlet based web applications.
- Servlet API defines the interactions of a Web container and a servlet.

☞ **Java Servlet API**

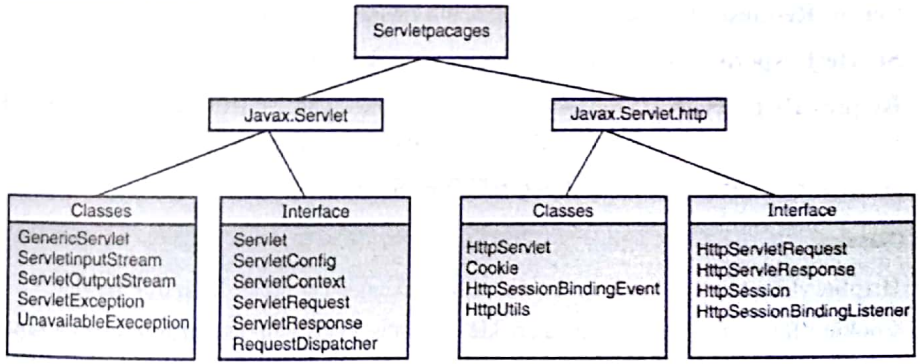


Fig. 3.4.1 : Java Servlet API

3.4.1 Packages

(I) **javax.servlet API :** It is the core Servlet API that consists of basic Servlet interfaces which all Servlets must implement. It also includes GenericServlet abstract class for creating non-protocol specific servlets. The two classes in the servlet API that implement the Servlet interface are GenericServlet and HttpServlet.

(II) **javax.servlet.http :** It consists of classes and interfaces for developing http protocol specific servlets.

(I) Javax.servlet API (Generic Servlet Class)

☞ **Classes**

1. **GenericServlet :** Defines a generic, protocol-independent servlet.
2. **ServletInputStream :** It is an input stream for reading binary data from a client request using readLine( ) method.
3. **ServletOutputStream :** It is an output stream for sending binary data back to the client.
4. **ServletException :** Include various servlet exceptions that can throw during runtime errors.
5. **UnavailableException :** Exception that a servlet or filter throws when a servlet is permanently or temporarily not available for processing the client request.

☞ **Interfaces**

1. **Servlet :** Consists of servlet life cycle methods that all servlets must implement.

   The basic Servlet Interface defines the service( ) method for handling client requests.
2. **ServletConfig :** It is used during the initialization phase to pass the configuration information to a servlet.
3. **ServletContext :** Used to get the information about the servlet container. It can be used to retrieve MIME type of a file, dispatch requests, or write data to a log file.
4. **ServletRequest :** It is used to send client information to a servlet.
5. **ServletResponse :** It is used to send response back to the client.
6. **RequestDispatcher :** It can be used to receive the client request and redirect it to other resource like another servlet, HTML file, or JSP file etc. on the server.

(II) Javax.servlet.http (Servlets Using HTTP Protocol)

☞ **Classes**

1. **HttpServlet :** It is an abstract class that can be extended to create an HTTP servlet.
2. **Cookie :** It can be used to create a cookie, which is a small amount of information sent by a servlet to a web browser. It can be saved into the client machine, and can be retrieved back to the server for session management.

3. **HttpSessionBindingEvent :** It sent an object that implements HttpSessionBindingListener when it is bound or unbound from a session.
4. **HttpUtils :** It is a deprecated class of Java Servlet API 2.

☞ **Interfaces**

1. **HttpServletRequest :** Extends the ServletRequest interface to provide HTTP-specific request information.
2. **HttpServletResponse :** Extends the ServletResponse interface to provide HTTP-specific response.
3. **HttpSession :** Used to identify a user across more than one page request for session management.
4. **HttpSessionBindingListener :** Causes an object to be notified when it is bound to or unbound from a session.

Syllabus Topic : HTTP Protocol and HTTP Methods

3.5 Servlet HTTP Protocol and Methods

☞ **HTTP (Hyper Text Transfer Protocol)**

- The Hypertext Transfer Protocol (HTTP) is application-level protocol for collaborative, distributed, hypermedia information systems. It is the data communication protocol used to establish communication between client and server.
- HTTP is TCP/IP based communication protocol, which is used to deliver the data like image files, query results, HTML files etc on the World Wide Web (WWW) with the default port is TCP 80. It provides the standardized way for computers to communicate with each other.

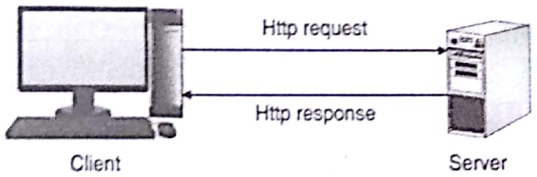


Fig. 3.5.1

☞ **Characteristics of HTTP (Hyper Text Transfer Protocol)**

- This is a dedicated protocol allows for exchange data over between web server and browser over a web.
- It is a request response protocol.
- By default Http requires reliable TCP connections on TCP port 80.

- Http is stateless, means each request is considered as the new request.

Basic Features of HTTP (Hyper Text Transfer Protocol)

- There are three fundamental features that make the HTTP a simple and powerful protocol used for communication:
  - **HTTP is media independent :** It refers to any type of media content can be sent by HTTP as long as both the server and the client can handle the data content.
  - **HTTP is connectionless :** It is a connectionless approach in which HTTP client i.e., a browser initiates the HTTP request and after the request is sends the client disconnects from server and waits for the response.
  - **HTTP is stateless :** here stateless indicates, the client and server are aware of each other during a current request only. Once the request is over, both of them forget each other. Due to the stateless nature of protocol, neither the client nor the server can retain the information about different request across the web pages.

3.5.1 Architecture of HTTP (Hyper Text Transfer Protocol)

- The Fig. 3.5.2 represents the basic architecture of web application and depicts where HTTP stands :

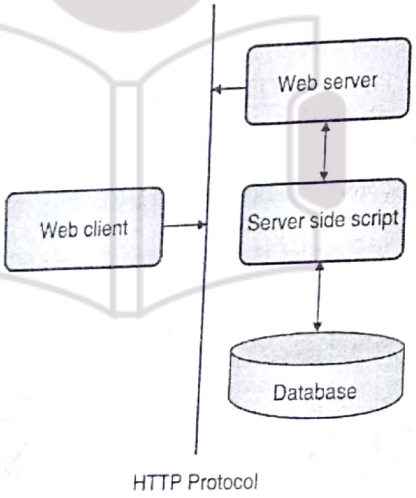


Fig. 3.5.2 : Architecture of web application

- HTTP is request/response protocol which is based on client/server based architecture. In this web browser, search engines, etc behaves as a HTTP clients, and the Web server like Servlet behaves as a server.

HTTP Requests

- The request sends by the computer to a web server that contains all sorts of potentially interesting information is known as HTTP requests.

- The HTTP client sends the request to the server in the form of request message which includes following information :
  - The Request-line
  - The analysis of source IP address, proxy and port
  - The analysis of destination IP address, protocol, port and host
  - The Requested URI (Uniform Resource Identifier)
  - The Request method and Content
  - The User-Agent header
  - The Connection control header
  - The Cache control header
- The HTTP request method indicates the method to be performed on the resource identified by the **Requested URI (Uniform Resource Identifier)**. This method is case-sensitive and should be used HTTP request methods in uppercase.

HTTP request methods

HTTP Request	Description
GET	Asks to get the resource at the requested URL.
POST	Asks the server to accept the body info attached. It is like GET request with extra info sent with the request.
HEAD	Asks for only the header part of whatever a GET would return. Just like GET but with no body.
TRACE	Asks for the loopback of the request message, for testing or troubleshooting.
PUT	Says to put the enclosed info (the body) at the requested URL.
DELETE	Says to delete the resource at the requested URL.
OPTIONS	Asks for a list of the HTTP methods to which the thing at the request URL can respond

Syllabus Topic : Web Server and Web Containers

3.6 Web Server and Web Containers (or Servlet Containers)

3.6.1 Web Server

- HTTP (Hyper Text Transfer Protocol) is a simple request /response protocol underpinning most web applications on the Internet, regardless of whether they are written in Java.
- A web server is package/sofware that helps to deliver web content (web page data) to the clients (e.g. web browser) through the Internet using HTTP protocol.

- An HTTP request has seven HTTP methods: GET, POST, HEAD, OPTIONS, TRA PUT and DELETE.
- An HTTP server can handle HTML and other client side technologies which don't req any server capability. The Apache HTTP Server is an example of an HTTP server.
- The server side technologies like Servlets, ASP, and PHP etc. will require their softw libraries to be installed at the web server. Without these libraries a web server won't able to execute those server technologies and is just an HTTP Server.

3.6.2 Web Containers (or Servlet Containers)

- Servlets cannot be executed on a HTTP Server. To deploy and run Servlets, a compati web server with a servlet container, such as Apache Tomcat or Jetty, is required.
- A servlet container is also called as web container. A web container is responsible for
  1. Managing the lifecycle of servlets,
  2. Mapping a URL to a particular servlet and
  3. Ensuring that the URL requester has the correct access rights.
- Servlets are java classes executed on the server and are used to generate dynamic we pages. We can use print statements within servlet to print html tags and html data to th browser.
- Java Server Page (JSP) is a server side technology based on java that help us to crea dynamically generated web pages based on HTML, XML etc. easily.
- While servlets can be considered as java classes with html, JSPs can be considered htm pages with java.
- JSPs are eventually converted into servlets by the server before execution.

Syllabus Topic : Servlet Interface

3.7 Servlet Interface

- Servlet interface consists of life cycle methods to initialize a servlet, service the request and remove a servlet from the server.
- Every servlet indirectly implements the Servlet interface.

☞ **Format of Servlet interface**

```
public interface Servlet
```

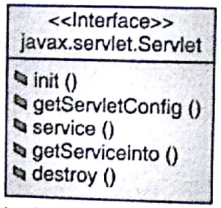
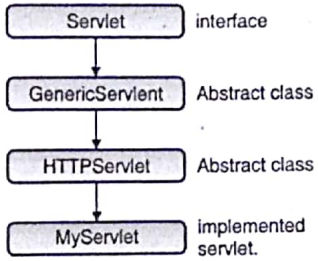


Fig. 3.7.1 : Servlet information format

- To implement the Servlet interface, extends either GenericServlet or an HTTP specificHttpServlet class.



- getServletConfig() method of servlet interface is used to get startup information, and getServletInfo() method allows servlet to return basic information about the servlet like author, version, and copyright.

3.7.1 Methods of Servlet Interface

1. public void init(ServletConfigconfig) throws ServletException

- The init method is called only once by the servlet container after instantiating the servlet. It is called before the servlet process any client request.
- The servlet cannot service the client request if the init method
  1. Throws a ServletException.
  2. Does not return within a time period defined by the Web server.

☞ **Parameter**

- config is a ServletConfig object used to retrieve the servlet's configuration and initialization parameters.
- The initialization arguments are defined in the configuration file by servlet engine. **For example :** An initialization argument can be a database identifier, that can be used to open a database connection before processing any client.

2. public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException

- The service() method is called to process a client request. It can be called zero, or many times after the servlet's init() method has completed successfully.
- Servlets which run inside multithreaded servlet containers, can handle multiple requests concurrently. Thus servlets are highly scalable.
- Synchronized block of code can be used to access the shared data like files, network connections, instance variables etc., by multiple threads. A synchronized block or a synchronized method guarantees that only one thread enters into the critical section code at a time.

Syllabus Topic : GenericServlet

3.8 GenericServlet Class

Q. Explain GenericServlet class with its methods.
Q. Write a short note on GenericServlet class.

GenericServlet class

- GenericServlet class is used to create protocol independent servlet program. It implements Servlet, ServletConfig and Serializable interfaces.
- It implements all the methods of the interfaces without the service() method.

Format

```
public abstract class GenericServlet extends java.lang.Object
implements Servlet, ServletConfig, java.io.Serializable
```

- A servlet program can extends GenericServlet to create a servlet application.
- The log method of the ServletContext interface is implemented by GenericServlet class.
- A generic servlet can be written, by overriding only the service() life cycle method.

Methods of GenericServlet

1.	**public GenericServlet()** Does nothing as GenericServlet is abstract class.
2.	**public String getInitParameter(String name)** – Returns the value of the initialization parameter, or null if the parameter does not exist. – It fetches the value from ServletConfig object. **Parameter** **(i) name** - indicate the name of the initialization parameter.
3.	**public Enumeration getInitParameterNames()** – Returns the names of the initialization parameters as an Enumeration of String objects, or an empty if the servlet has no initialization parameters.
4.	**public ServletContext getServletContext()** – Returns the ServletContext object in which the servlet is running.
5.	**public ServletConfig getServletConfig()** – Returns the servlet's ServletConfig object.
6.	**public String getServletInfo()** – Returns information about the servlet, such as author, version, and copyright. – By default, it returns an empty string. – Program can override the getServletInfo() method to return the respective values.

7.	**public String getServletName()** – Returns the name of the servlet instance.
8.	**public void init(ServletConfig config) throws ServletException** – It is called by the servlet container for performing the initialization process. **Parameter** **(i) config** - ServletConfig object that provides configuration information.
9.	**public abstract void service(ServletRequest req, ServletResponse res) throws ServletException, IOException** – Used to process the client request and send the response back. – As it is an abstract method; subclasses, must override it. **Parameters** **(i) req** - the ServletRequest object for client's request details. **(ii) res** - the ServletResponse object for the servlet's response back to the client.
10.	**public void destroy()** – It is called by the servlet container to remove all the resources used by the servlet application.
11.	**public void log(String msg)** – It writes the information to a servlet log file. **Parameter** **(i) msg** - a String message to be written to the log file.
12.	**public void log(String message, Throwable t)** – It writes the information and a stack trace for Throwable exception to the log file. **Parameters** **(i) message** - A String message for error or exception description. **(ii) t** - Throwable error or exception

Syllabus Topic : HttpServlet

3.9 HttpServlet Class

Q. Explain HttpServlet class with its constructor and methods.
Q. Write a short note on HttpServlet class.

- The HttpServlet class can be used to create http protocol specific servlet applications. It extends the GenericServlet class and implements Serializable interface.
- It includes http specific methods such as doGet(), doPost(), doHead(), doTrace() etc.
- HttpServlet extends the GenericServlet thus derive all the features of GenericServlet.

☞ **Format**

```
public abstract class HttpServlet extends GenericServlet implements java.io.Serializable
```

- When a subclass is created by extending from HttpServlet; it must override at least one of the method : doGet(), doPost(), doPut(), doDelete(), init() and destroy() or getServletInfo().
- While overriding doGet() or doPost() method it can perform the following task
  1. Read the request data.
  2. Write the response headers.
  3. Get the response's writer object.
  4. Write the response data.
  5. Include content type and encoding.
  6. If PrintWriter object is used to return the details back to the client, set the content type before accessing the PrintWriter object.
  7. The Content Length header can be set by ServletResponse.setContentLength(int) method.

☞ **Methods of HttpServlet**

1.	**public HttpServlet()** – Does nothing, as it is an abstract class.
2.	**protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException** – It handles a GET request. Overriding doGet() also supports an HTTP HEAD. – A HEAD request is a GET request that returns only the request header without body details. – The header fields need to written prior sending the response, as HTTP headers need to be sent before the response body. – The content length is set automatically. – If GET request is incorrect, doGet() returns an "Bad Request" message.
3.	**protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException** – It handles a POST request. – Unlimited data can send to the web server using HTTP POST method. It can be used for passing secure information such as credit card numbers from client to the server.
4.	**protected void doPut(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException** – It handles a PUT request. – The PUT method can be used to transfer a file from client to the server like FTP file transfer method. – Content header need not be changed if doPut() method is overriding.

5.	**protected void doTrace(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException** – It handles a TRACE request. – A TRACE method is used to send the headers back to the client, for debugging purpose. Overriding the doTrance() method is not required.
6.	**protected void doOptions(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException** – It handles an OPTIONS request. – The OPTIONS request is used to send back the header details to the client that specify various HTTP methods the server supports. – **Example :** If a servlet overrides doGet()method; doOptions() will return the header as : "Allow : GET, HEAD, TRACE, OPTIONS". – Overriding the doOptions () method is not required.
7.	**protected void doHead(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException** – It handles the HTTP HEAD request. – The client can send a HEAD request to know the header of a response to find the information like Content-Type or Content-Length. – The HTTP HEAD method sets the Content-Length header automatically depending on the size. doHead() method can override to change the response headers.
8.	**protected void doDelete(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException** – It handles a DELETE request. – The DELETE HTTP operation can be used by a client to delete a document or Web page from the server.
9.	**protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException** – It receives HTTP requests and forward o the respective do method.
10.	**protected long getLastModified(HttpServletRequest req)** – It returns the time the HttpServletRequest object was last modified, in milliseconds. – If the time is not known, it returns a negative number. – It allows the browser and proxy to reduce the load on server and network resources.

All service method of HttpServletservice(), doGet(), doPost(), doPut(), doTrace(), doOptions(), doHead() and doDelete() has :

☞ **Parameters**

1. **req-request :** data from client to server.
2. **resp-response :** from servlet back to the client.

☞ **Throws**

1. **IOException :** occurs when an input or output error is detected while the servlet tries to process the request.
2. **ServletException :** occurs if the request could not be handled.

Syllabus Topic : Servlet Life Cycle

3.10 Servlet Life Cycle

Q. Explain Servlet Life Cycle in detail.
Q. Explain different phases of Servlet Life Cycle.

☞ **Servlet :** A servlet is a server side scripting techniques supported by java programming language. Servlet is a java class file that can derive the features of web servers which host applications and can be accessed through request-response programming model.

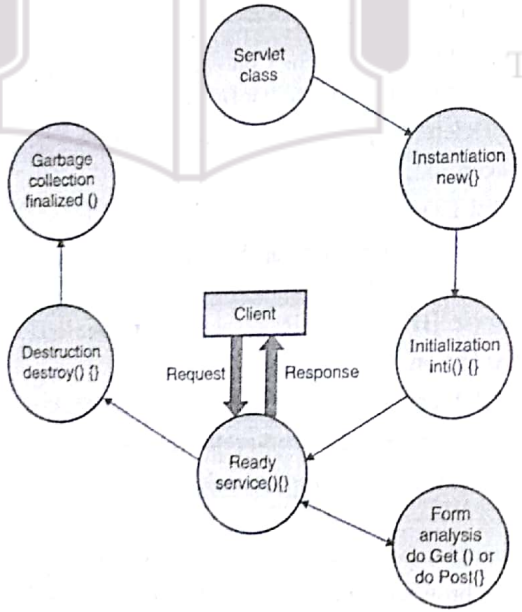


Fig. 3.10.1 : Model of Servlet life Cycle

- Web container within which the servlet is deployed handles the servlet life cycle methods.
- When a servlet is requested by a client web Container performs the following task as :

1. **Initialization :** Web container first check whether a servlet instance exists or not. If not it will first load the servlet class and creates an instance of the servlet class. Once a servlet object is ready it will call the init( ) life cycle method to initializes the servlet
2. **Servicing :** Each client request can be processed by service() life cycle method which takes request and response objects as parameters.
3. **Destroying :** Servlet can be removed by calling the destroy() method if it is no longer used.

(1) Initialization

- After the Web container load and create an instance of the servlet class it initializes the servlet.
- Initialization is performed before processing any client request.
- Servlet can perform various task during the initialization process like
  - Read the persistent configuration data.
  - Initialize the required resources.
  - Override the init( ) method that need to be executed once in the servlet life cycle.

(2) Servicing

- After the initialization process, servlets serve the client request using the service() method.
- For HTTP specific servlets, the method of generating the response is
  - First provide the response header information.
  - Retrieve an output stream object from the response; to send the data back to the client.
  - Write the resultant content to the output stream.

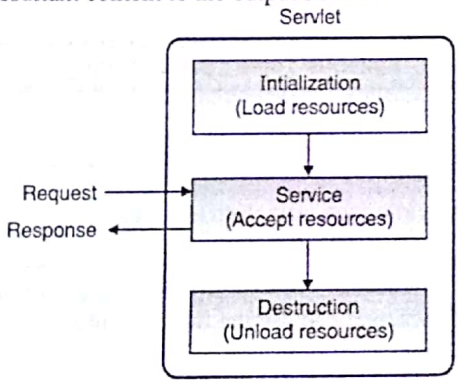


Fig. 3.10.2 : Servlet life cycle

(a) **Request :** Data passed from a client to the server.

(b) **Responses :** Data passed from a server to the client.

(3) Destroying or Finalizing a Servlet

- destroy() life cycle method is used to release all the resources occupied by the servlet once all client requests are processed and the servlet is idle for a specific amount of time.
- All servlet's service( ) methods should complete when a servlet is removed.
- If the servlet has long-running service requests : Call the destroy() method to notify the long-running threads to shutdown and wait them to complete.

☞ **Life cycle method of a generic servlet**

```
public class test extends GenericServlet
{
   public void init(ServletConfigconfig) throws ServletException
   {
         //override initialization
   }
   public void service(ServletRequestreq, ServletResponse res) throws ServletException,
   IOException
   {
     //override the servicing
   }
   public void destroy()
   {
         //override the destroying
   }
}
```

Syllabus Topic : ServletConfig

3.11 ServletConfig Interface

Q. Explain ServletConfig interface along with its methods.
Q. Write a short note on ServletConfig interface.

- ServletConfig is used to **initialize and specify the context information to the** init() life cycle method. Servlet container can use the ServletConfig for initialization of parameters using web.xml deployment descriptor.
- <init-param> tag is used to initialize parameter in web.xml deployment file.

- **Example :** To give the administrator value as "xyz" within initialization parameter :

```
<init-param>
        <param-name>AdminName</param-name>
        <param-value>xyz</param-value>
</init-param>
```

- It can be also be used to access the ServletContext object that gives the sever information.

☞ **Format of the ServletConfig interface**

```
public interface ServletConfig
```

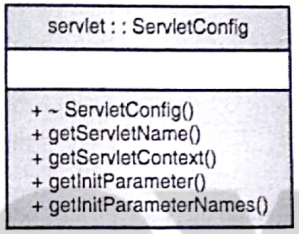


Fig. 3.11.1

☞ **Methods**

1. **public String getServletName()**
   - It returns the name of the servlet object.
   - The name is generated and assigned by the deployment descriptor of a web application.
   - For unregistered servlet the function return the servlet's class name.
2. **public ServletContext getServletContext()**
   It returns the ServletContext object in which the servlet is executing which provides the server environment details.
3. **public String getInitParameter(String name)**
   - It returns the value of an initialization parameter. It returns null if the parameter does not exist.
   - parameter "name" is the name of the initialization parameter.
4. **public Enumeration getInitParameterNames()**
   - It returns all the initialization parameters names.
   - It returns empty Enumeration if the servlet has no initialization parameters.

Difference between the ServletContext and ServletConfig

Q. Differentiate between ServletConfig and ServletContext.

Sr. No.	ServletConfig	ServletContext
1.	ServletConfig is one per Servlet.	ServletContext is one per web application.
2.	It can be used to pass the deployment time parameters to the servlet during the servlet initialization like database name, file name etc.	It can be used to access the Web application parameters configured in the deployment descriptor file (WEB.xml).
3.	It can be used to access the ServletContext object.	It can be used for the inter application communication between servlets, JSPs and other components of a web application.
4.	Can access the initialization parameters for a servlet instance.	Can be used to access the server information about the container and the version of the API it supports.

Program 3.11.1 : Write a servlet program to display the name of the servlet and its initialization parameters with its respective values.

Soln. :

Program displaying the name of servlet and parameters

test.java

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class test extends GenericServlet
{
    public void service(ServletRequest req, ServletResponse res) throws ServletException,
    IOException
    {
        PrintWriter out = res.getWriter();
        res.setContentType("text/html");
        try
        {
            ServletConfigconfig=getServletConfig();
```



```
            out.print("<br>Servlet Name : "+config.getServletName());
            Enumeration ename = config.getInitParameterNames();
            while(ename.hasMoreElements())
            {
                String name=ename.nextElement().toString();
                out.print("<br> Parameter Name : "+name);
                out.print("<br> Parameter Value : "+config.getInitParameter(name));
            }
        }
        catch(Exception e)
        {
            out.println("Error "+e.getMessage());
        }
    }
}
```

web.xml

```
<web-app>
<servlet>
<init-param>
<param-name>AdminName</param-name>
<param-value>xyz</param-value>
</init-param>
<servlet-name>test</servlet-name>
<servlet-class>test</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>test</servlet-name>
<url-pattern>/test</url-pattern>
</servlet-mapping>
<session-config>
<session-timeout> 30</session-timeout>
</session-config>
</web-app>
```

Output

```
Servlet Name : test
Parameter Name :AdminName
Parameter Value : xyz
```

Parameters

- **req :** The ServletRequest object for client's request.
- **res :** The ServletResponse object for servlet's response.

1. **public void destroy()**
   - The destroy() method is called by the servlet container to remove the servlet so that won't be available for the further client service.
   - It is called once all threads within the servlet's service() method is completed or after a timeout period.
   - It clean up all the resources that are used by the servlet like memory, file handles, threads, etc.
   - It also ensures that persistent state is synchronized by the servlet's current state.
2. **public ServletConfig getServletConfig()**

   It returns a ServletConfig object, for retrieving the initialization and startup parameters.
3. **public String getServletInfo()**

   It is used to fetch the information about the servlet, like author name, version, and copyright.

Program 3.11.2 : Write a servlet program to insert a record into the database table named depttable with the attributes deptno, deptname, and deptloc. Use servlet life cycle methods to initialize the connection, insert a record and release the database connection.

Soln. :

Program to insert a record into database table

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class test extends GenericServlet
   {
        Connection con;
        public void init(ServletConfigconfig) throws ServletException
        {
            try
            {
                Class.forName("com.mysql.jdbc.Driver");
```



```
                con=DriverManager.getConnection("jdbc:mysql://localhost/db","root",
                "shajilpa");
            }
            catch(Exception e)
            { }
        }
public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException
   {
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        try
        {
            Statement  st = con.createStatement();
            String sql="insert into depttable values(2,'sales','Mumbai')";
            st.executeUpdate(sql);
            out.println("Data Added...");
        }
        catch(Exception e){ out.println("Error..."+e.getMessage()); }
   }

public void destroy()
   {
   try
   {
        con.close();
   }
   catch(Exception e) {   }
   }
}
```

In the program

- Database connection is established in the initialization method.
- Inserting the record into the table is performed in the service method
- In the destroy method the established database connection is closed.

Syllabus Topic : ServletContext

3.12 ServletContext Interface

Q. Write a short note on ServletContext interface.
Q. Discuss the various methods of ServletContext interface.

- ServletContext object is created by the web container at time of deploying the project. This object can be used to get configuration information from web.xml file. There is only one ServletContext object per web application.
- If we want share information with many , it is better to provide it from the web.xml file using the <context-param> element.

Advantage of ServletContext

- It's very easy to maintain if any information is shared to all the servlet, mostly we make it available for all the servlet.
- When provide information from the web.xml file, and suppose the information is changed, we don't need to modify the servlet. Thus it ignores maintenance problem.

Usage

1. ServletContext object provides an interface between the container and servlet.
2. it can be used to get configuration information from the web.xml file.
3. it can be used to set, get or remove attribute from the web.xml file.
4. The ServletContext object can be used to provide inter-application communication.

ServletContext interface methods

1. **public String getInitParameter(String name) :** Returns the value for the specified parameter name.
2. **public Enumeration getInitParameterNames() :** Returns the names of the context's initialization parameters.
3. **public void setAttribute(String name,Object object) :** sets the given object in the application scope.
4. **public Object getAttribute(String name) :** Returns the attribute for the specified name.
5. **public Enumeration getInitParameterNames() :** Returns the names of the context's initialization parameters as an Enumeration of String objects.

6. **public void removeAttribute(String name) :** Removes the attribute with the given name from the servlet context.

How to get the object of ServletContext interface

1. **getServletContext()** method of ServletConfig interface returns the object of ServletContext.
2. **getServletContext()** method of GenericServlet class returns the object of ServletContext.

Syllabus Topic : Servlet Communication

3.13 Servlet Communication

- Communication between Java servlets is called as Servlet communication. Here, user's requests are sent from one servlet and the response object is passed by another servlet.
- So, when a request object is passed from one servlet to another servlet, then you can use getParameter() method to get the input that the user has given in a HTML/JSP form.

```
Webapps
  |_ MyServlet
       |_ index.html
       |_ WEB-INF
            |_ web.xml
            |_ classes
                 |_ Servlet_1.java
                 |_ Servlet_1.class
                 |_ Servlet_2.java
                 |_ Servlet_2.class
```

- Let's design **index.html** page :

```
<html>
<head>
<title>Demo Program for Servlet Communication</title>
</head>
<body>

<form action="/MyServlet/pass" method="post">

Student Name: <input type="text" name="sname" width="50" />
Marks : <input type="text" name="marks" width="5" />
<input type="submit" name="submit" value="Show" />
```

```
</form>

</body>
</html>
```

Let's write two classes Servlet_1 and Servlet_2 in web.xml for servlet communication:

```
<web-app>

<servlet>
<servlet-name>comservlet</servlet-name>
<servlet-class>Servlet_1</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>comservlet</servlet-name>
<url-pattern>/pass</url-pattern>
</servlet-mapping>

<servlet>
<servlet-name>lservlet</servlet-name>
<servlet-class>Servlet_2</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>lservlet</servlet-name>
<url-pattern>/we</url-pattern>
</servlet-mapping>

</web-app>
```

- Here, as */pass* is the url pattern pointing that the corresponding **action** controlling class is *Servlet_1*, and the user's data sent to this url pattern, *Servlet_1*will be called first by the servlet container. Next one, */we*, when servlet container gets request to the url-pattern */we*, the *Servlet_2*class will be invoked.
- Let's write Servlet_1.java :

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class Servlet_1 extends HttpServlet
{
   public void doPost(HttpServletRequestreq,HttpServletResponse res) throws
   ServletException,IOException
   {
        String sn=req.getParameter("sname");
        String m=req.getParameter("marks");
```



```
        getServletContext().getRequestDispatcher("/we").forward(req,res);
   }
}
```

Let's write **Servlet_2.java**

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class Servlet_2 extends HttpServlet
{

   public void doPost(HttpServletRequestreq,HttpServletResponse res) throws
   ServletException,IOException
   {

        String sn=req.getParameter("sname");
        String m=req.getParameter("marks");

        PrintWriter pw=res.getWriter();
        pw.println("<h1>STUDENT NAME IS"+sn+"</h1>");
        pw.println("<h1>MARKS IS "+m+"</h1>");

   }
}
```

3.14 Session Tracking Mechanisms

Session Tracking

- Session tracking mechanism is used to identify a user across multiple pages or for a web site and store all the information about a particular user.
- The servlet can maintain a session in different ways as :

(i)	HttpSession	(ii)	Cookie
(iii)	URL rewriting	(iv)	Hidden form fields

(i) HttpSession

It can be used to create a session between an HTTP client and an HTTP server. The session persists for a specified time period to store a user details.

Format

```
public interface HttpSession
```

It allows the servlets to :

- View and manipulate session details like session identifier, creation time, and l accessed time
- Used to bind objects to sessions, across multiple user connections
- Session details are stored only to the current application context;it won't be direc visible for another.
- An HttpSession can be retrieved by invoking the getSession() method HttpServletRequest as:

```
HttpSession session = request.getSession();
```

	Methods of HttpSession
1.	**public ObjectgetAttribute(String name)** It returns the object bound with the name in the session.
2.	**public EnumerationgetAttributeNames()** It returns an Enumeration of String containing all the session names bound to the session.
3.	**public void setAttribute(String name,Object value)** It is used to store name and its corresponding value into the session.
4.	**public void removeAttribute(String name)** It removes the object from the session storage.
5.	**publicintgetMaxInactiveInterval()** It returns the maximum time interval, servlet container will preserve the session details.
6.	**public void setMaxInactiveInterval(int interval)** It is used to set the maximum time, in seconds, to store the details in session.
7.	**public StringgetId()** It is used to return unique identifier assigned to the session.
8.	**public booleanisNew()** It returns true if the client not yet joined with the session.
9.	**public void invalidate()** Clear all the session details and its values.

(ii) Cookie

- Cookie is named value pair of information, sent by the server to the client browser that gets saved at the client computer which can be used for session tracking.
- Browser can send request to the server along with cookie to identify the client

- addCookie () method of the response object can be used to add a cookie into the client browser as :

```
Cookie c1 = new Cookie("city", "Mumbai");
response.addCookie(c1);
```

- If the browser disables cookies, cookie cannot be saved at client computer and session tracking fails.

	Methods of Cookie
1.	publicString getName() It returns the name of the cookie.
2.	publicString getValue() It returns the value of the cookie.
3.	public void setValue(String newValue) It is used to assign a new value to the cookie after its creation
4.	public long getMaxAge() It is used to return the maximum age of the cookie, in seconds
5.	public void setMaxAge(long expiry) It is used to set the maximum age of the cookie in seconds.
6.	publicString getDomain() It is used to return the domain name set for the cookie.

(iii) URL rewriting

- URL rewriting method adds extra details at the end of the URL to identify the session. The added information's are generally sessionid or useridthat can be used for session tracking.

☞ **Example**

```
Original URL : "http://server:port/servlet/ServletName"
Rewritten URL : " http://server:port/servlet/ServletName?sessionid=1234"
```

- URL rewriting can be used for session tracking if the browsers does not support cookies.
- The disadvantage of URL rewriting is that it needs to generate every URL dynamically to assign a session ID.

(iv) Hidden form fields

- Hidden fields can be used to store details within the webpages that gets hidden from the user but sent to the server for session tracking.
- Hidden form details get passed to the server through http request.

☞ **Example :** For hidden form field

```
<INPUT TYPE="hidden" NAME="city" VALUE="Mumbai">
```

- It cannot be used for session tracking if large set of details need to be stored for a user session.

Review Questions

Q. 1 Write a short note on CGI. Describe its advantages and disadvantages.

Q. 2 What are servlets ? Explain the request/response paradigm of servlets.

Q. 3 Write a short note on Web Application Model.

Q. 4 Write a short note on Servlet API.

Q. 5 Write a short note on GenericServlet class.

❑❑❑

UNIT - II

CHAPTER 4

Java Server Pages (JSP)

Syllabus Topics

Introduction, JSP LifeCycle, JSP Implicit Objects and Scopes, JSP Directives, JSP Scripting Elements, JSP Actions : Standard actions and customized actions.

Syllabus Topic : Introduction to JSP

4.1 Introduction to JSP (Java Server Pages)

Q. What is JSP ?
Q. What is a role of JSP? Explain it in detail.
Q. Explain the processing of JSP pages in detail.

- JSP (Java Server Pages) technology can be used to create dynamic web application. Its methodologies are more efficient then servlets.
- Basically it separates the presentation logic from the business logic and thereby it provides more work independency to the designer and developer of the web application.
- Every JSP page includes HTML tags and JSP tags.
- We can separate designing and development very easily in JSP, that's why jsp pages are easier to maintain than servlet.
- JSP also provides some additional features such as Expression Language, Custom Tag etc.

JSP Processing

Process of creating the web page using JSP in a web server :

(i) Browser send HTTP request to the web server for a JSP page, and redirected to JSP engine.

(ii) The JSP engine loads the JSP page and converts it into a servlet if it is not created yet.