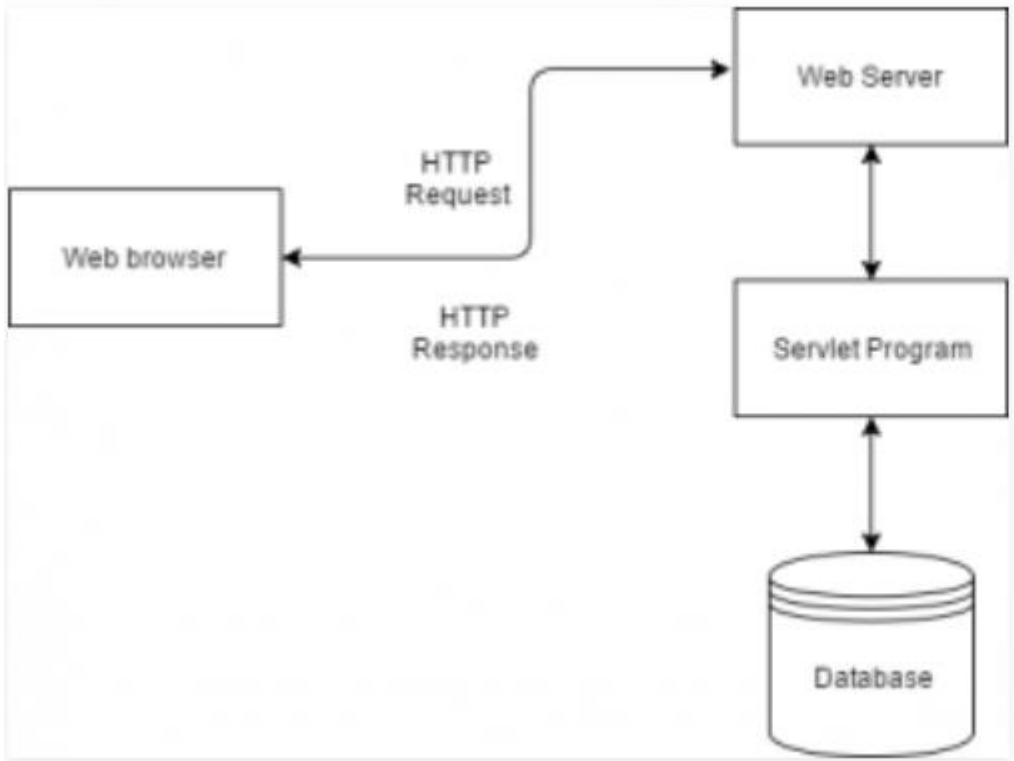
# MIDDLEWARE TECHNOLOGY

Dr.S.Vaaheedha Kfatheen
Assistant Professor
Department of computer science & IT
Jamal Mohamed College(A)
Trichy

# 2.1 INTRODUCTION

- Servlets are the Java programs that run on the Java-enabled web server or application server.
- They are used to handle the request obtained from the webserver, process the request, produce the response, then send a response back to the webserver.
- Properties of Servlets are as follows:
  - Servlets work on the server-side
  - Servlets are capable of handling complex requests obtained from the webserver.


Web Server
HTTP
Request
Web browser
HTTP
Response
Servlet Program
Database

- Execution of Servlets basically involves six basic steps:
  - 1. The clients send the request to the webserver
  - 2. The web server receives the request
  - 3. The web server passes the request to the corresponding Servlet
  - 4. The servlet processes the request and generates the response in the form of output
  - 5. The servlet sends the response back to the webserver.
  - 6. The web server sends the response back to the client and the client browser displays it on the screen

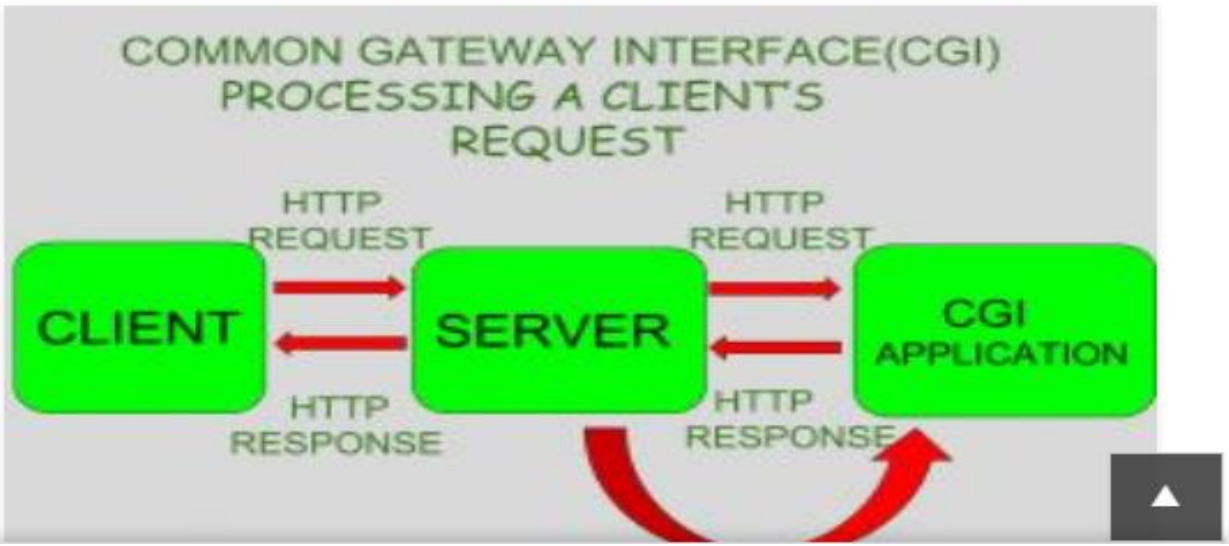
The **server-side extensions** are nothing but the technologies that are used to create dynamic Web pages. Actually, to provide the facility of dynamic Web pages, Web pages need a container or Web server. To meet this requirement, independent Web server providers offer some proprietary solutions in the form of **APIs**(Application Programming Interface).
These **APIs** allow us to build programs that can run with a Web server. In this case, **Java Servlet** is also one of the component APIs of **Java Platform Enterprise Edition** which sets standards for creating dynamic Web applications in Java.

Before learning about something, it's important to know the need for that something, it's not like that this is the only technology available for creating dynamic Web pages. The Servlet technology is similar to other Web server extensions such as **Common Gateway Interface**(CGI) scripts and **Hypertext Preprocessor** (PHP). However, Java Servlets are more acceptable since they solve the limitations of **CGI** such as low performance and low degree scalability.

**CGI** is actually an external application that is written by using any of the programming languages like **C** or **C++** and this is responsible for processing client requests and generating dynamic content.

In CGI application, when a client makes a request to access dynamic Web pages, the Web server performs the following operations :

- It first locates the requested web page *i.e* the required CGI application using URL.
- It then creates a new process to service the client's request.
- Invokes the CGI application within the process and passes the request information to the application.
- Collects the response from the CGI application.
- Destroys the process, prepares the HTTP response, and sends it to the client.



- Servlets are programs that run on a Web Server and build Web pages.
- Java technology is very helpful for this kind of CGI programming.
- Building Web pages is useful for a number of reasons:
  - The web page is based on data submitted by the user
    - For example, the results pages from search engines are generated this way.
  - The data changes frequently
    - For example, a weather-report or news headlines page might build the web page dynamically.
  - The web page uses information from corporate databases or other such sources.
    - For example, building a web page at an on-line store that lists current prices and number of items in stock.

- Servlets are designed to work within a request/response processing model.
- In a request/response model, a client sends a request message to a server and the server responds by sending back a reply messages.
- Requests can come in the form of an HTTP, FTP, or a custom protocol.
- A common request/response model is between World Wide Web browsers and World Wide Web servers.
- When a user selects a website through the browser, a request is sent to the corresponding Web Server.

- The Web server responds to the HTML content of that website.
- The servlet enhances the functionality of the Web server.
- They are the most suitable for database applciations and other applications that require secure access to a website, that dynamically generate custom HTML documents to be displayed by the browsers and that maintain unique session information for each client.

| Servlet | CGI (Common Gateway Interface) |
| --- | --- |
| Servlets are portable and efficient. | CGI is not portable |
| In Servlets, sharing data is possible. | In CGI, sharing data is not possible. |
| Servlets can directly communicate with the webserver. | CGI cannot directly communicate with the webserver. |
| Servlets are less expensive than CGI. | CGI is more expensive than Servlets. |
| Servlets can handle the cookies. | CGI cannot handle the cookies. |

# 2.2 Advantages of Servlets over CGI

1. Efficiency

   In CGI, a new process is started for each HTTP request.

   With Servlets, the JVM stays up, and each request is handled by lightweight java thread.

   In CGI, if there are N simultaneous request to the same CGI program, then the code for the CGI program is loaded into memory N times.

   With Servlets, there are N threads but only a single copy of the servlet class.

2. Convenience

Servlets have an extensive infrastructure for automatically parsing and decoding HTML data, reading and setting HTTP headers, handling Cookies, tracking sessions, and many other such utilities.

3. Quality of power

Servlets can talk directly to the Web server

Servlets can also share data among each other, making useful things like database connection Pools easy to implement.

Maintain information from request to request, simplifying things

like session tracking and catching of previous computations.

4. Portability

Servlets are written in Java and follow a well-standardized API.

Servlets are supported directly or via a plugin on almost every major Web server.

5. Low cost

There are a number of free or very inexpensive web servers available that are good for personal use or low volume websites.

# 2.3 The Servlet Life Cycle

- Servlets run on the Web server platform as part of the same process as the Web server itself.
- The Web server is responsible for initializing, invoking and destroying each servlet instance.
- A Web server communicates with a servlet through a simple interface, javax.servlet.Servlet.
- This interface consists of three main methods
  - Init( )
  - Service( )
  - Destroy( )

And two ancillary methods:

getServletConfig( )

getServletInfo( )

Servlets are to Web servers and Applets are to Web browsers

An Applet runs in a web browser, performing actions it requests through a specific interface.

A Servlet does the same, running in the web server.

| Sr. No. | Key | Applets | Servlets |
|---|---|---|---|
| 1 | Execution | Applets are executed on client-side i.e applet runs within a Web browser on the client machine. | Servlets on other hand executed on the server-side i.e servlet runs on the web Page on server. |
| 2 | Parent packages | Parent package of Applet includes java.applet.* and java.awt.* | Parent package of Servlet includes javax.servlet.* and java.servlet.http.* |
| 3 | Methods | Important methods of applet includes init(), stop(), paint(), start(), destroy(). | Lifecycle methods of servlet are init( ), service( ), and destroy( ). |
| 4 | User interface | For the execution of the applet, a user interface is required such as AWT or swing. | No such interface is required for the execution of servlet. |
| 5 | Required Bandwidth | The applet requires user interface on the client machine for execution so it requires more bandwidth. | On the other hand, Servlets are executed on the servers and hence require less bandwidth. |
| 6 | Secure | Applets are more prone to risk as execution is on the client machine. | Servlets are more secure as execution is under server security. |

1. The init( ) method

When a servlet a first loaded, its init( ) method is invoked.

This allows the servlet to perform any setup processing such as opening files or establishing connections to their servers.

If a servlet has been permanently installed in a server, it loads when the server starts to run.

Otherwise, the server activates a servlet when it receives the first client request for the services provided by the servlet.

The init( ) method takes one argument a reference to a ServletConfig object which provides initialization arguments for the servlet.

This object has a method getServletContext( ) that returns ServletContext object containing information about the environment of the servlet.

2. The Service( ) method

The Service( ) method is the heart of the servlet.

Each request message from a client results in a single call to the servlet's service( ) method.

- The service( ) method reads the request and produces the response message from its two parameters:
  - A ServletRequest object with data from the client.
  - The data consists of name/value pairs of parameters and an InputStream.
  - Several methods are provided that return the client's parameter information.
  - The InputStream from the client can be obtained via the getInputStream( ) method.

- A ServletResponse represents the servlet’s reply back to the client.
- When preparing a response, the method setContentType( ), is called first to set the MIME type of the reply.

There are two ways for a client to send information to  a Servlet.

The first is to send parameter values and the second is to send information via the InputStream.

The destroy( ) method

This method is called to allow the servlet to clean up any resources such as open files or database connections before the servlet is unloaded.

If clean up method is not required, this can be an empty method.

The server waits to call the destroy( ) method until either all service calls are complete, or a certain amount of time has passed.

# 2.4 Servlet API

The Java Servlet API defines the interface between servlets and servers.

This API is packaged as a standard extension to the JDK under javax:

- Package javax.servlet
- Package javax.servlet.http

The API provides support in four catagories:

- Servlet life cycle management
- Access to servlet context
- Utility classes
- HTTP-specific support classes

# 2.4.1 – The javax.servlet Package

This package contains interfaces and classes to establish a framework in which servlets operate.

| *Interface* | *Description* |
|---|---|
| `Servlet` | An interface that declares the life cycle methods for a servlet. |
| `ServletConfig` | An interface that describes the configuration parameters for a servlet. This is passed to the servlet when the Web server calls its `init()` method. |
| `ServletContext` | An interface that enables servlets to register events and access information about their environment. |
| `ServletRequest` | An interface that describes how to get information about a client request. |
| `ServletResponse` | An interface that describes how to pass information back to the client. |
| `GenericServlet` | A base servlet implementation. It takes care of saving the `ServletConfig` object reference, and provides several methods that delegate their functionality to the `ServletConfig` object. It also provides a dummy implementation for `init()` and `destroy()`. |
| `ServletInputStream` | A subclass of `InputStream` used for reading the data part of a client's request. It adds a `readLine()` method for convenience. |

# 2.4.2 The javax.servlet.http Request

| Interface | Description |
|---|---|
| ServletOutputStream | An OutputStream to which responses for the client are written. |
| ServletException | Should be thrown when a servlet problem is encountered. |
| UnavailableException | Should be thrown when the servlet is unavailable for some reason. |

**10.5.2 The javax.servlet.http Package**

| Interface | Description |
|---|---|
| HttpServletRequest | A subclass of ServletRequest that defines several methods that parse HTTP request headers. |
| HttpServletResponse | A subclass of ServletResponse that provides access and interpretation of HTTP status codes and header information. |
| HttpServlet | A subclass of GenericServlet that provides automatic separation of HTTP request by method type. For example, an HTTP GET request will be processed by the service() method and passed to a doGet() method. |
| HttpUtils | A class that provides assistance for parsing HTTP GET and POST requests. |

# 2.4.3 Servlet Interface

**`Servlet` Interface**

All servlets must implement the `Servlet` interface. This interface defines the following methods that are invoked automatically by the server on which the servlet is installed.

| *Method* | *Description* |
|---|---|
| `void init (ServletConfig conf)` | Automatically called once during a servlet's execution life cycle to initialize. |
| `ServletConfig get ServletConfig()` | Obtains the servlet's configuration information. |
| `void service ()` | First method called by every servlet to respond to a client. |
| `String getServletInfo()` | Returns servlet's information such as servlet's author and version. |
| `Void destroy()` | Called when a servlet is terminated. |

# 2.4.4 HttpServlet Class

- This is an abstract class form servlet package.
- This class overrides the method service to distinguish the request received from a client Web browser.
- The class defines methods doGet and doPost to respond to GET and POST requests from a client respectively.
- GET and POST are the two common request types.
- The GET request is used to retrieve content from a web server such as an HTML document.
- The POST requests are used to send the server information from an HTML form in which the client enters data, or to send authenticated information to the server.

- These two methods use HttpServletRequest object and HttpServletResponse object as arguments.
- The methods of HttpServletRequest access the data requested by the client.
- The methods of HttpServletResponse return the results in the form of HTML document to the web client.
- The HttpServlet class has some other methods:
  - dodelete – used to delete a file from the server
  - doOptions – returns information about server options to client
  - doPut – used to store a file on the server
  - dotrace – used for debugging

# 2.4.5 HttpServlet Request Interface

| *Method* | *Description* |
|---|---|
| `String getParameter (String name)` | Returns the value associated with a parameter represented by name argument. |
| `Enumeration getParameterNames()` | Returns names of all the parameters |
| `String[] getParameterValues (String name)` | Returns an array containing the values for a specified servlet parameter. |
| `Cookie[] getCookies()` | Returns an array of cookie objects stored on the client by the server. |
| `HttpSession getSession (Boolean create)` | Returns an `HttpSession` object associated with the present (current) browsing session of the client. |

# 2.4.6 HttpServletResponse Interface

| *Method* | *Description* |
|---|---|
| `Void addCookie (Cookie cookie)` | Used to add cookie to the header of the response to the client |
| `ServletOutputStream getOutputStream ()` | Gets byte output stream to send binary data to the client |
| `PrintWriter getWriter()` | Gets character output stream to send text data to the client |
| `Void setContentType (String type)` | Specifies the MIME type of the response to the browser. For example, "text/html" |

# 2.5 A Simple Servlet Program

# 2.6 Handling HTTP GET Requests

The problem of supporting simultaneous updates from multiple clients has been solved by database systems (DBMSs); unfortunately the HTTP protocol does not work well with database systems. This is because DBMSs need to maintain a persistent connection between a client and the DBMS to determine which client is trying to update the data. The HTTP protocol does not support this type of a connection as it is a message based, stateless protocol. Solving this problem is not easy and never elegant! Fortunately, the Servlet API defines a means to track client/server sessions. This is covered in theMaintaining Session Informationsection later in this course. Without session management tracking, one can resort to several different strategies. They all involve writing data to the client in hidden fieldswhich is then sent back to the server. The simplest way to handle updates is to use an optimistic locking scheme based on date/time stamps. One can use a single date/time stamp for a whole form of data, or one could use separate date/time stamps for each "row" of information. Once the update strategy has been selected, capturing the data sent to the server via an HTTP POSTmethod is straightforward. Information from the HTML form is sent as a series of parameters

# 2.7 Handling HTTP POST Requests

HTTP POSTmethod processing differs from HTTP GETmethod processing in several ways. First, because POSTis expected to modify data on the server, there can be a need to safely handle updates coming from multiple clients at the same time. Second, because the size of the information stream sent by the client can be very large, the doPost()method must open an InputStream(or Reader) from the client to get any of the information. HTTP POSTdoes not support sending parameters encoded inside of the URL as does the

# 2.8 Cookies

- A cookies is a named piece of data maintained by a browser sent by the Web server in a communication, normally for session management.
- HTTP connections are stateless, a cookie can store persistent information across multiple HTTP connections.
- The advantages of setting cookies in the client computer are four-fold.
- It can store information on the user’s computer for later retrieval.

# Cont.,

- Examples
- If the user is doing online shopping, and cookie is employed in his/her computer, it can store the user's preferences and later in the next communication the servlet can examine the cookies, identify the client's preferences and immediately display the product of interest to the client.
- Second, cookie can be employed to store the username and password by sending a unique user ID from the site which requires frequent registration using username and password.
- Third, they are very much useful for customizing a Web page like customizing the search results from a search engine, or a weather report etc.,

# Cont.,

- Fourth, they are employed for focusing advertisements on certain websites like search engines, online book stores, etc.,
- The cookies let the web server to remember the sensitive information like username, password which can be misused by hackers.
- The cookie class has the following constructor and methods.
- Cookie (String name, String value)

| Method | Description |
|---|---|
| `Object clone()` | Returns a copy |
| `String getDomain()` | Returns the domain |
| `Int getMaxAge()` | Returns the age |
| `String getName()` | Returns the name of the cookie |
| `String getPath()` | Returns the path |
| `String getValue()` | Returns the value |
| `boolean getSecure()` | Returns true if the cookie is secure |
| `void setDomain(String d)` | Sets the domain to d |
| `void setMaxAge(int sec)` | Sets the maximum age to sec |
| `void setPath(String p)` | Sets the path to p |
| `void setSecure(boolean sec)` | Sets the security flag to sec |
| `void setValue(String v)` | Sets value to v |

# 2.9 Session Tracking

- The HTTP protocol is a stateless protocol which dose not store any state information.
- But in some applications, it is necessary to save the state information.
- This can be done using sessions mechanism provided by servlets.
- A session can be created using getSession( ) method of HttpServletRequest.
- An HttpSession object is created if already one does not exist.

| *Method* | *Description* |
|---|---|
| `HttpSessionContextgetSessionContext()` | Returns the context associated with the session |
| `Object getValue(String name)` | Returns the object bound to name |
| `String[] getValueNames()` | Returns the names of all objects bound in the session |
| `void invalidate()` | Invalidates the session and removes for the context |
| `void putValue(String name, Object object)` | Binds the object to name in the session |
| `void removeValue(String name)` | Removes the object bound to name from the session |

- References
- N.P.Gopalan & J.Akilandeswary, Web Technology A developer Perspective, PHI Learning Private Limited, 2009.

Thank You