

Servlet Lecturer-1

Outline

❖ Introduction to Java

❖ Introduction to Java EE

- Server
 - WebServer
 - Application Server
- Servlet/Web Container
- ServerSide Technologies
- Servlet
 - javax.servlet package
- Servlet LifeCycle method.



Introduction to JAVA

Java is a high level, robust, object-oriented and secure programming language.

Applications of Java:

1. Desktop Applications such as acrobat reader, media player, antivirus, etc.
2. Web Applications such as irctc.co.in, etc
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games, etc.



Java Platforms and Edition

1) Java SE (Java Standard Edition)

It is a Java programming platform. It includes Java programming APIs such as `java.lang`, `java.io`, `java.net`, `java.util`, `java.sql`, `java.math` etc. It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.

2) Java EE (Java Enterprise Edition)


It is an enterprise platform which is mainly used to develop web and enterprise applications. It is built on the top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB etc.

3) Java ME (Java Micro Edition)

It is a micro platform which is mainly used to develop mobile applications.

4) JavaFX

It is used to develop rich internet applications. It uses a light-weight user interface API.



Continue...

- ❖ JSE/ JDK
 - JRE
 - JVM
 - Libraries
 - Development Tools



Java EE(Java Enterprise Edition)

- ❖ JEE is the large API that developed by **Sun Microsystems** that contains the multiple API as the standard to develop the enterprise, scalable and distributed application.
- ❖ **Enterprise Application**
 - Application that automate the task of any enterprise(Business Organisation) over the internet.
- ❖ **Scalable Application**
 - Application that grow itself with the least effort, time and cost.
- ❖ **Distributed Application**
 - Application that can distribute their logic(Functionality) among the different application.
- ❖ **Note:**
 - These different application can be the web application, window based application or even other distributed application.


Continue...

❖ The different API's contained by the Java EE are categorised in the following ways:

➤ **Component Technologies**

- Web Components
 - Servlet
 - JSP
- EJB Components

➤ **Service Technologies**

- JMS(JAVA Message Service)
 - JNDI(JAVA Naming and Directory Interface)
 - JavaMail API
 - JAAS(Java Authentication and authorisation service)
 - JTA(Java Transaction API)
 - Web Services
 - JDBC(javax.sql)
- 

Continue...

➤ **Communication Technologies**

- HTTP Protocol
- SMTP Protocol(Simple Mail Transfer)
- IIOP(Internet Inter-ORB protocol)
- RMI(Remote Method Invocation)



Server

- ❖ Server is a device or a computer program that accepts and responds to the request made by the other program, known as client.
- ❖ It is used to manage the network resources and for running the program or software that provides services.
- ❖ There are two types of servers:
 - Web Server
 - Application Server



Web Server

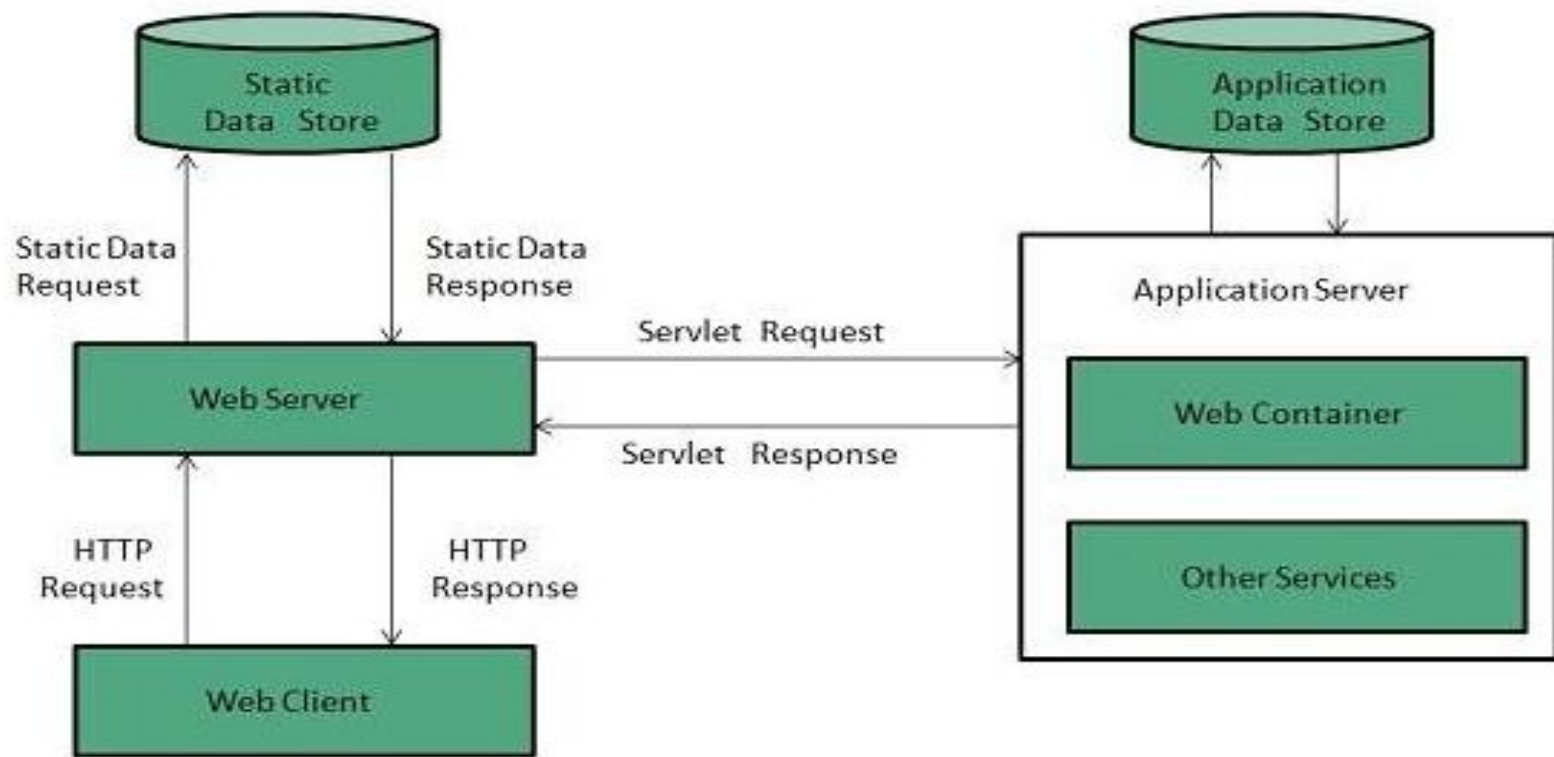
- ❖ Web server contains only web or servlet container. It can be used for servlet, jsp, struts, jsf etc. It can't be used for EJB.
- ❖ It is a system where the web content can be stored. In general web server can be used to host the web sites.
- ❖ Example of Web Servers are:
 - Apache Tomcat
 - Resin



How Web Server Works

- ❖ It can respond to the client request in either of the following two possible ways:
 - Generating response by using the script and communicating with database.
 - Sending file to the client associated with the requested URL.





Important Points

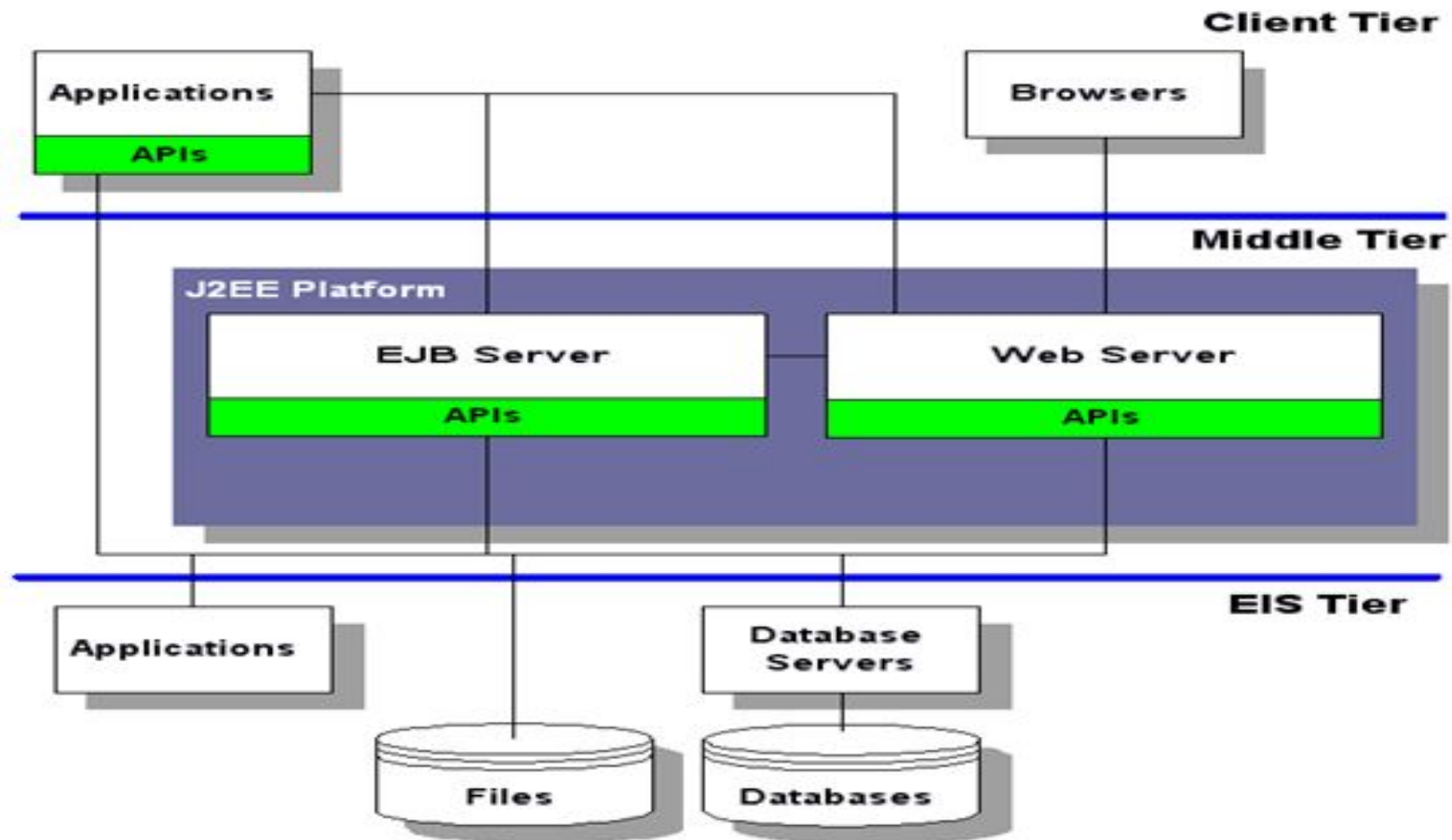
- ❖ If the requested web page is not found, then web server will send the HTTP response: Error 404 Not Found.
- ❖ When the web server searches the requested page if requested page is found then it will send to the client with an HTTP response.
- ❖ If the client requests some other resources then web server will contact to application server and store data to construct the HTTP response.



Application Server

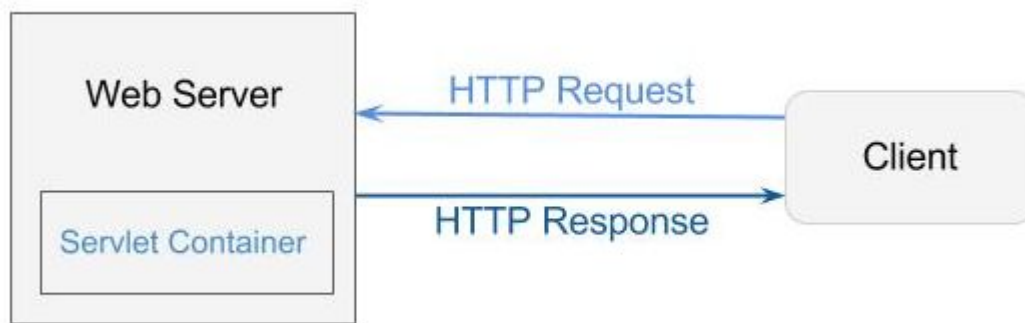
- ❖ Application server contains Web and EJB containers. It can be used for servlet, jsp, struts, jsf, ejb etc.
- ❖ It is a component based product that lies in the middle-tier of a server centric architecture.
- ❖ It provides the middleware services for state maintenance and security, along with persistence and data access.
- ❖ It is a type of server designed to install, operate and host associated services and applications for the IT services, end users and organisations.
- ❖ The Examples of Application Servers are:
 - JBoss: Open-source server from JBoss community.
 - Glassfish: Provided by Sun Microsystems. Now acquired by Oracle.
 - Weblogic: Provided by Oracle. It is more secure.
 - Websphere: Provided by IBM.





Web Container

- ❖ Web container is the part of a web server.
- ❖ A web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.



Continue...

- ❖ Servlet container performs many operations that are given below:
 - Life Cycle Management
 - Multithreaded support
 - Object Pooling
 - Security etc.



Server Side Technologies

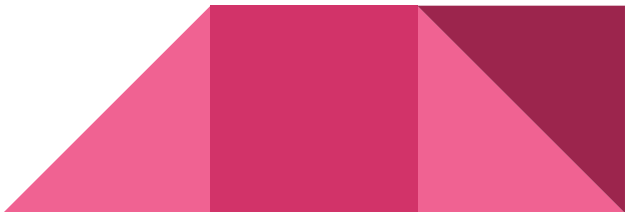
❖ **Server Side JavaScript**

- It is not powerful server side technology; rather it is mainly for client side.

❖ **PHP**

- It is light weighted and does not provide all the features corresponding to the Java EE.

❖ **ASP.net**

- This is developed by Microsoft and it is used to generate the dynamic pages.
 - The limitation is it can only be run within the single server known as the IIS server(Internet information server).
- 

Continue...

❖ CGI

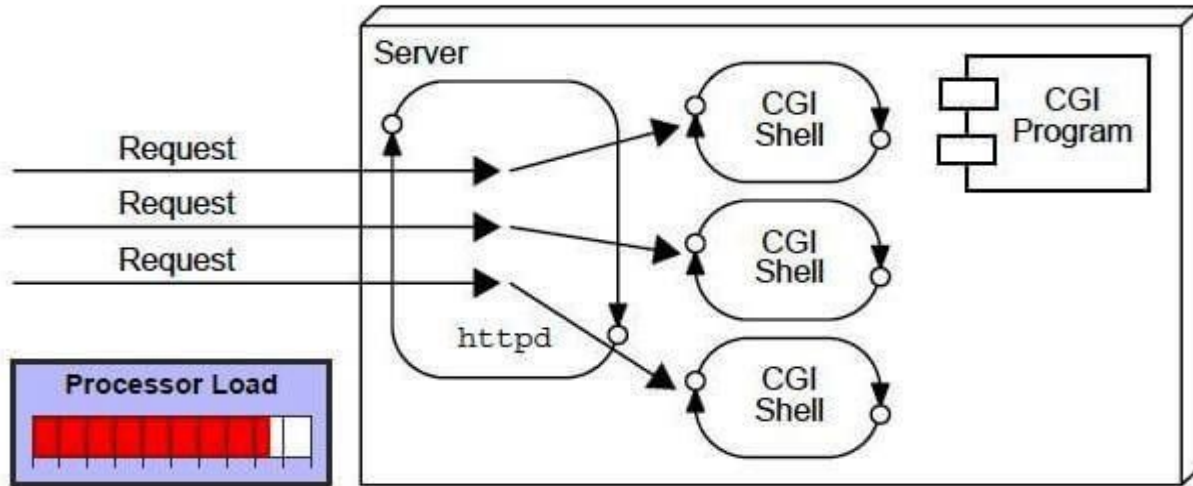
- Common Gateway Interface is the server side technology which is only a protocol and uses the C/C++/perl programming language to write the program.
- These programs were known as the CGI script.
- CGI had the following limitations
 - It is the platform dependent.
 - It is the process based.

❖ Servlet

- Servlet is the java API comes under the Java EE and use to prepare the java server side programs.
- Servlet program always execute under the web-container by the special tool known as the servlet engine.

CGI

- ❖ CGI technology enables the web server to call an external program and it also pass HTTP request information to the external program to process the request. For each request, it starts a new process.



Disadvantages of CGI

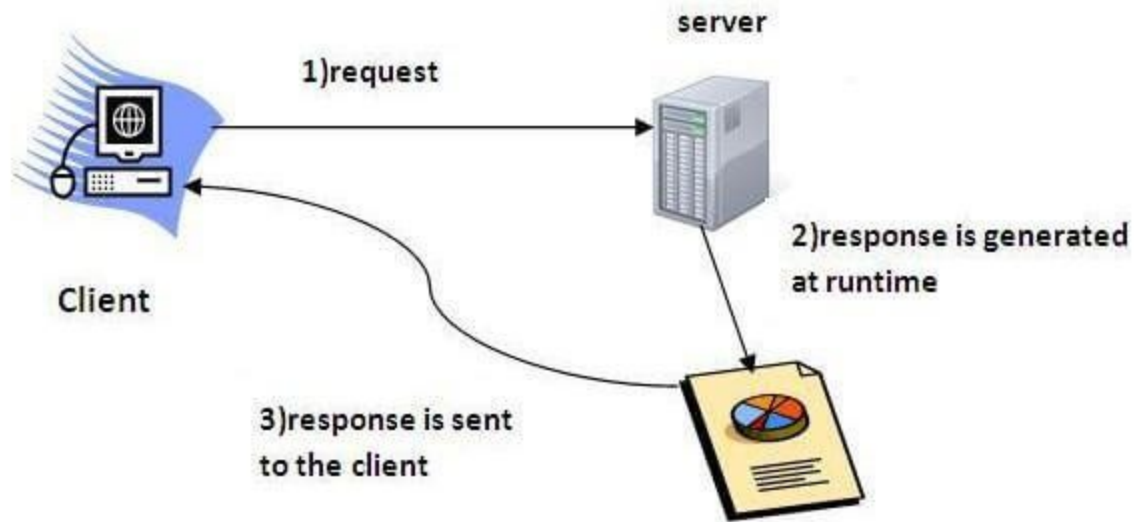
- ❖ If the number of clients increases, it takes more time for sending the response.
- ❖ For each request, it starts a process, and the web server is limited to start processes.
- ❖ It uses platform dependent language e.g. C, C++, perl etc.



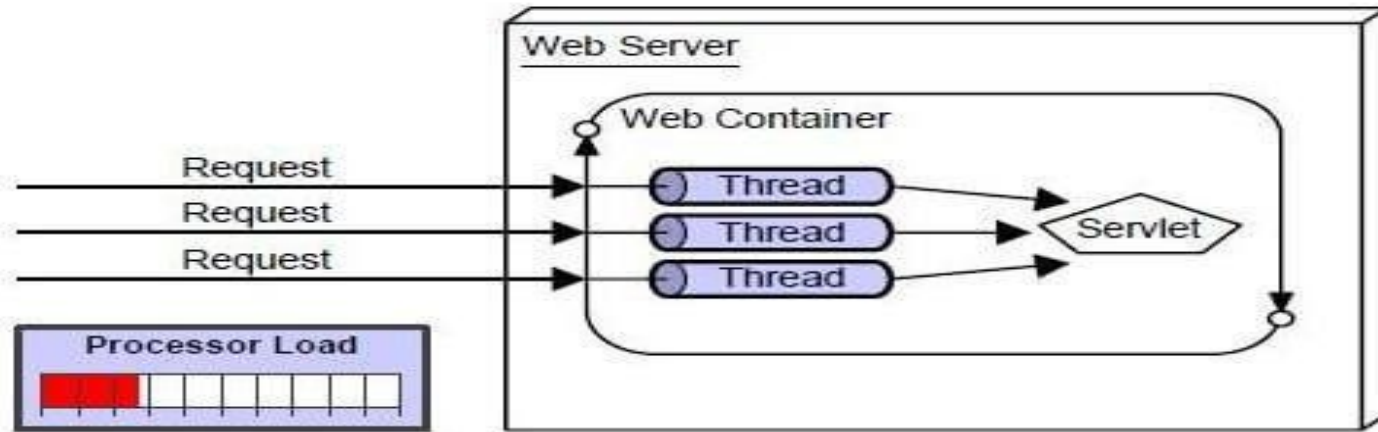
Servlet

- ❖ Servlet is a technology which is used to create a web application.
- ❖ Servlet is an API that provides many interfaces and classes including documentation.
- ❖ Servlet is an interface that must be implemented for creating any servlet.
- ❖ Servlet extends the capabilities of the server and responds to the incoming requests.
- ❖ Servlet is a web component that is deployed on the server to create a dynamic web page.





- ❖ The web container creates threads for handling the multiple requests to the Servlet.
- ❖ Threads have many benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low.



Advantages of Servlet

❖ **Better performance**

- It creates a thread for each request, not process.

❖ **Robust**

- JVM manages Servlets, no need to worry about the memory leak, garbage collection, etc.

❖ **Portability**

❖ **Secure**



Javax.servlet Package

This package provides the classes and interfaces used for the creation and development of the servlet program.



javax.servlet.Servlet interface

- ❖ This interface provides the lifecycle methods required for the execution of the servlet programs.
- ❖ There are three lifecycle methods in this interface
 - `init()`
 - `service()`
 - `destroy()`



public void init(ServletConfig config)

- ❖ This method is invoked by the webserver just after the creation of the servlet object.
- ❖ This method will be invoked only once in the life of servlet object.
- ❖ The developer can use this method to specify the initialization task of the servlet object such as creating the database connection, open any specific file etc.



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

- ❖ This method is invoked by the web-server each and every time when the request for this servlet is received from the client.
- ❖ This is the important method to define the functionality of the servlet.
- ❖ ServletRequest and ServletResponse are the interfaces in the javax.servlet package and their implemented classes are provided by the webserver.



public void destroy()

- ❖ This method will be invoked by the webserver just before the destruction of the servlet object.
- ❖ The servlet developer can use this method to define the cleanup task such as closing the **database connection** etc.



javax.servlet.ServletConfig Interface

- ❖ Implemented class of this interface provided by the web server. An instance of **ServletConfig** contains the configuration of the servlet.



Important Notes

- ❖ javax.servlet.Servlet interface also provides the two non-lifecycle method that will be invoked by the developers according to the requirement.
- ❖ Non-lifecycle method-
 - **public String getServletInfo()**
 - returns information about servlet such as writer, copyright, version etc.
 - **public ServletConfig getServletConfig()**
 - returns the object of ServletConfig.
- ❖ WebServer create only one object of Servlet class when the first request is received from the client.
- ❖ Whenever any subsequent request from any client received then webserver creates a new thread and invokes the service method with the existing servlet object.

Example

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

public class FirstServlet implements Servlet{
    ServletConfig config=null;

    public void init(ServletConfig config){
        this.config=config;
        System.out.println("servlet is initialized"); }

    public void service(ServletRequest req, ServletResponse res) throws IOException,ServletException{
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        out.print("<html><body>");
        out.print("<b>Hello Servlet </b>");
        out.print("</body></html>"); }

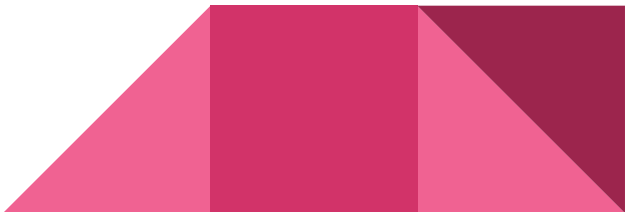
    public void destroy() {System.out.println("servlet is destroyed");}

    public ServletConfig getServletConfig() {return config;}

    public String getServletInfo() {return "copyright 2019-2020";}
}
```



How Servlet Execute in the WebServer

1. Object of requested servlet is created.
 2. Object of ServletConfig created.
 3. Init method invoked and reference of ServletConfig is passed.
 4. Object of ServletRequest is created.
 5. Parameter sent by the client in the request are stored in the object of ServletRequest.
 6. Object of ServletResponse created.
 7. New thread started.
 8. Service method is invoked and the references of ServletRequest and ServletResponse object are passed.
 9. From the service method response content is generated and stored into servlet response.
 10. Dynamically generated content forwarded towards the web-browser by loading into the http packet.
 11. Object of ServletRequest and ServletResponse destroyed.
- 

GenericServlet Class

- ❖ GenericServlet class implements Servlet, ServletConfig and Serializable interfaces. It provides the implementation of all the methods of these interfaces **except the service method**.
- ❖ GenericServlet class can handle any type of requests so it is protocol-independent.
- ❖ We can create generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.



GenericServlet Class Methods

- ❖ **public void init(ServletConfig config)**
 - is used to initialize the servlet.
- ❖ **public abstract void service(ServletRequest request, ServletResponse response)**
 - provides service for the incoming request. It is invoked at each time when user requests for a servlet.
- ❖ **public void destroy()**
- ❖ **public ServletConfig getServletConfig()**
- ❖ **public String getServletInfo()**
- ❖ **public String getInitParameter(String name):-**
 - returns the parameter value for the given parameter name.
- ❖ **public String getServletName()**
 - returns the name of the servlet object.
- ❖ **public void log(String msg)**
 - writes the given message in the servlet log file.

HttpServlet Class

- ❖ The HttpServlet class extends the GenericServlet class and implements Serializable interface.
- ❖ It provides http specific methods such as doGet, doPost, doHead etc.



HttpServlet Class Methods

- ❖ **public void service(ServletRequest req, ServletResponse res)**
 - dispatches the request to the protected service method by converting the request and response object into http type.
- ❖ **protected void service(HttpServletRequest req, HttpServletResponse res)**
 - receives the request from the service method, and dispatches the request to the do____() method depending on the incoming http request type.
- ❖ **protected void doGet(HttpServletRequest req, HttpServletResponse res)**
 - handles the GET request. It is invoked by the web container.
- ❖ **protected void doPost(HttpServletRequest req, HttpServletResponse res)**
 - handles the POST request. It is invoked by the web container.
- ❖ **protected void doPut(HttpServletRequest req, HttpServletResponse res)**
- ❖ **protected void delete(HttpServletRequest req, HttpServletResponse res)**

Example

```
//MyServlet.java
```

```
import javax.servlet.*;
```

```
import java.io.*;
```

```
import javax.servlet.http.*;
```

```
public class MyServlet extends HttpServlet{
```

```
    public void doGet(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException  
    {
```

```
        // to set response content type  
        res.setContentType("text/html");  
        PrintWriter out = res.getWriter();  
        out.print("<html><body>");  
        out.print("Hello Servlet");  
        out.print("</body></html>");
```

```
    }
```

```
}
```

