Servlets

Website

Website is a collection of related web pages that may contain text, images, audio and video. The first page of a website is called home page. Each website has specific internet address (URL) that you need to enter in your browser to access a website.

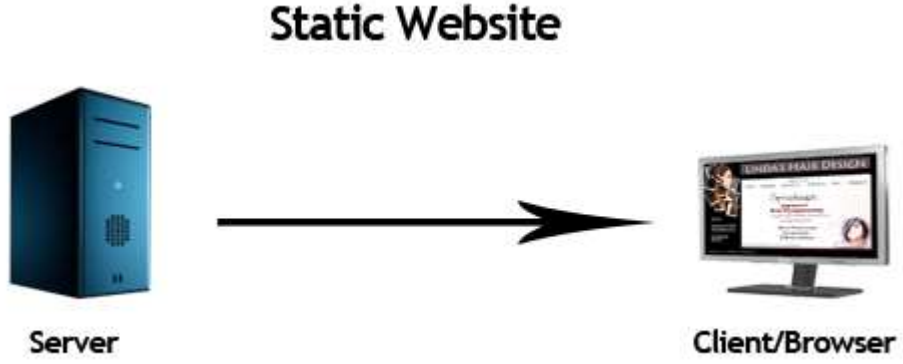
A website can be of two types:

- Static Website
- Dynamic Website

Static website

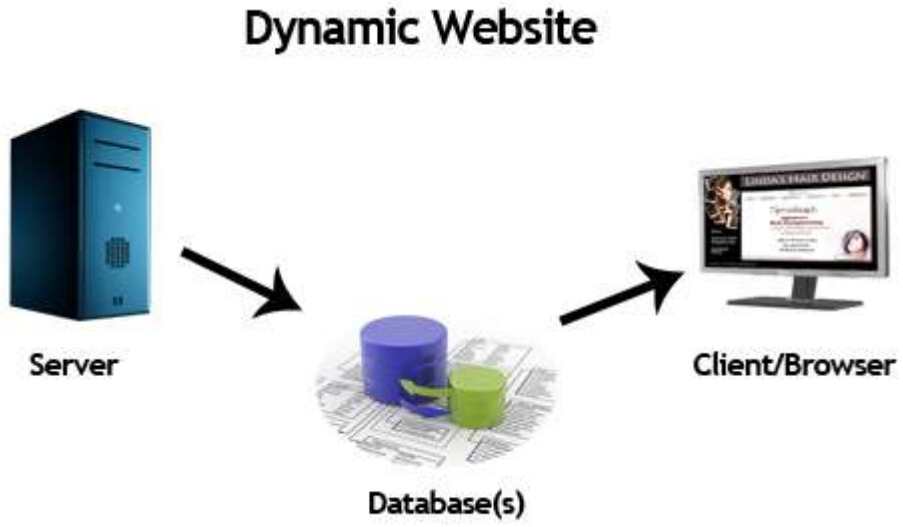
Static website is the basic type of website that is easy to create. You don't need the knowledge of database design to create a static website. Its web pages are coded in HTML.

The codes are fixed for each page so the information contained in the page does not change and it looks like a printed page.



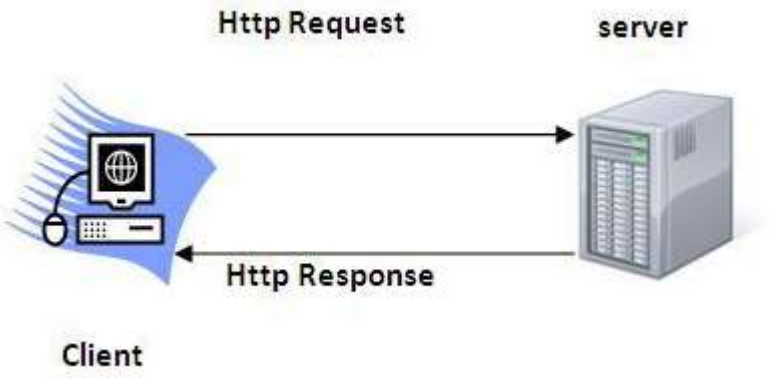
Dynamic website

- Dynamic website is a collection of dynamic web pages whose content changes dynamically. It accesses content from a database. Therefore, when you alter or update the content of the database, the content of the website is also altered or updated.
- Dynamic website uses client-side scripting or server-side scripting, or both to generate dynamic content.

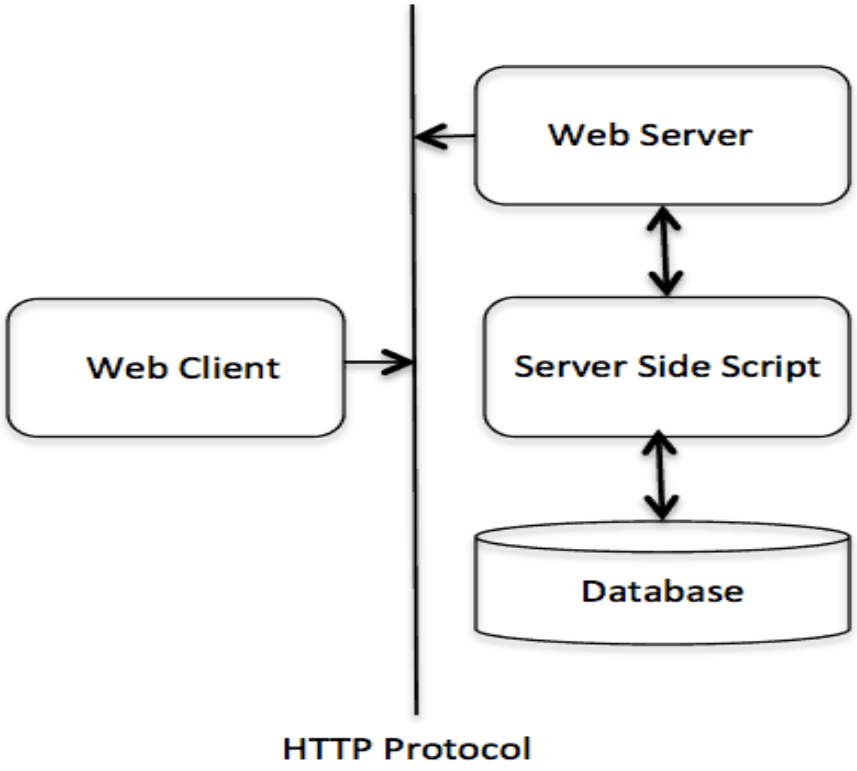


HTTP (Hyper Text Transfer Protocol)

- The Hypertext Transfer Protocol (HTTP) is the data communication protocol used to establish communication between client and server.



Architecture of HTTP



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

HTTP Requests

- The HTTP client sends the request to the server in the form of request message

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 loop back 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

The two most common HTTP methods are: GET and POST.

GET and POST

Two common methods for the request-response between a server and client are:

- **GET**- It requests the data from a specified resource
- **POST**- It submits the processed data to a specified resource

GET method

The query string (name/value pairs) is sent inside the URL of a GET request:

```
GET/RegisterDao.jsp?name1=value1&name2=value2
```

Data is sent in request header in case of get request.

Post method

- The query string (name/value pairs) is sent in HTTP message body for a POST request:

POST/RegisterDao.jsp HTTP/1.1
Host: www. Google.com
name1=value1&name2=value2

post request original data is sent in message body.

GET vs POST

GET	POST
1) In case of Get request, only **limited amount of data** can be sent because data is sent in header.	In case of post request, **large amount of data** can be sent because data is sent in body.
2) Get request is **not secured** because data is exposed in URL bar.	Post request is **secured** because data is not exposed in URL bar.
3) Get request **can be bookmarked.**	Post request **cannot be bookmarked.**
4) Get request is **idempotent** . It means second request will be ignored until response of first request is delivered	Post request is **non-idempotent.**
5) Get request is **more efficient** and used more than Post.	Post request is **less efficient** and used less than get.

Introduction to Servlets

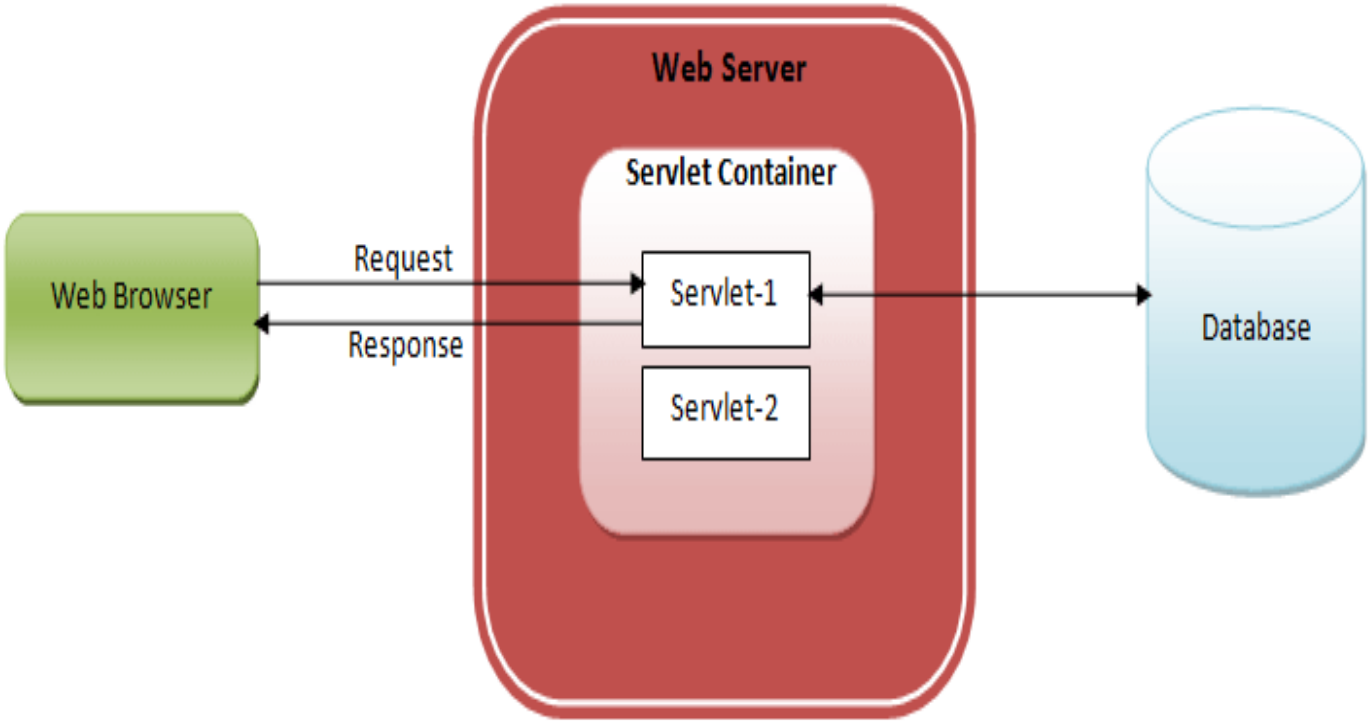
- **Servlet** technology is used to create a web application (resides at server side and generates a dynamic web page).
- **Servlet** technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language.
- There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

What is a Servlet?

Servlet can be described in many ways, depending on the context.

- 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 is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page

Java Servlet Architecture

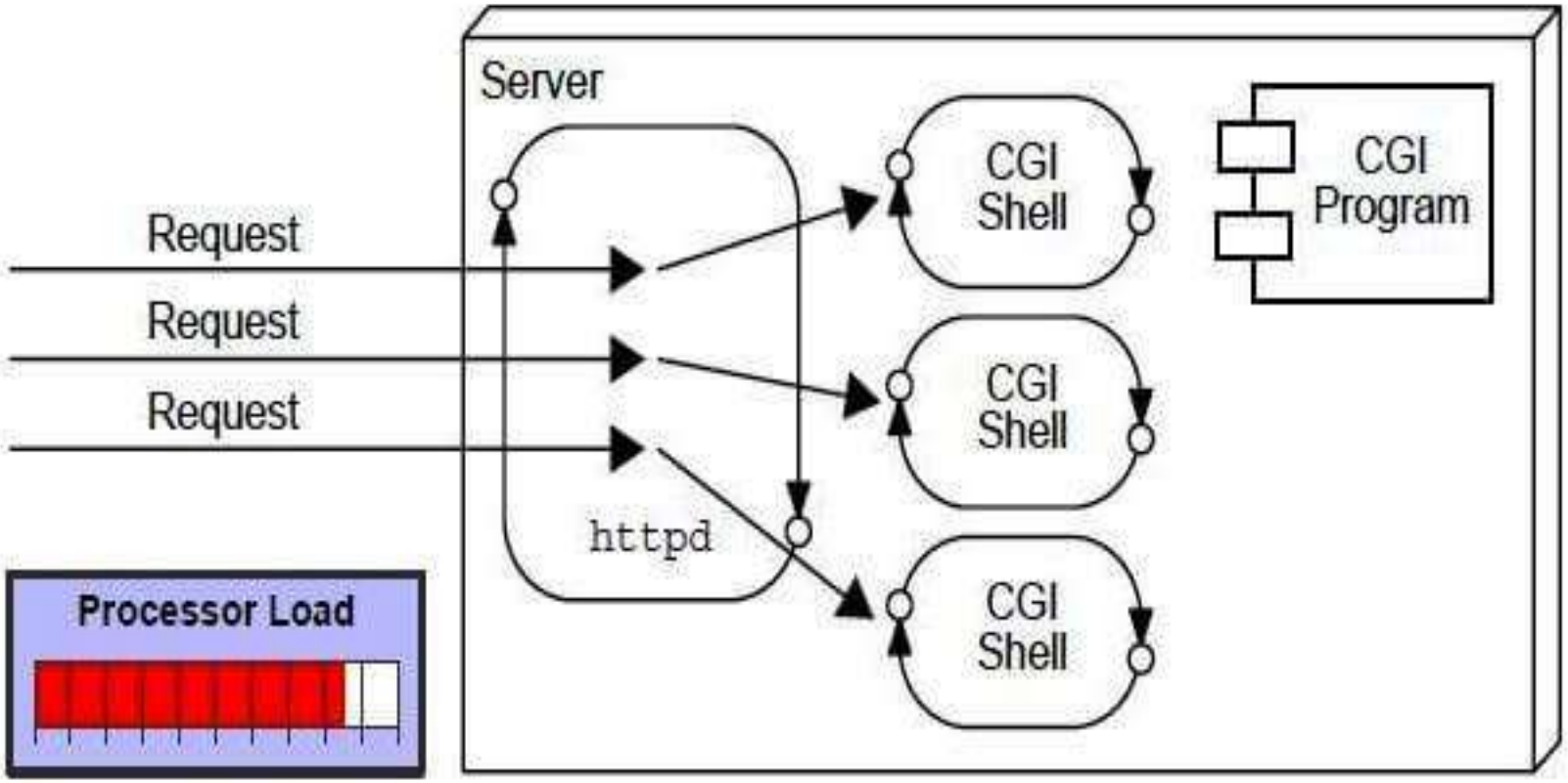


javaconceptoftheday.com

CGI(Commmon Gateway Interface)

- CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request.

- For each request, it starts a new process.


Server
CGI
Shell
CGI
Program
Request
Request
Request
CGI
Shell
httpd
CGI
Shell
Processor Load

Disadvantages of CGI

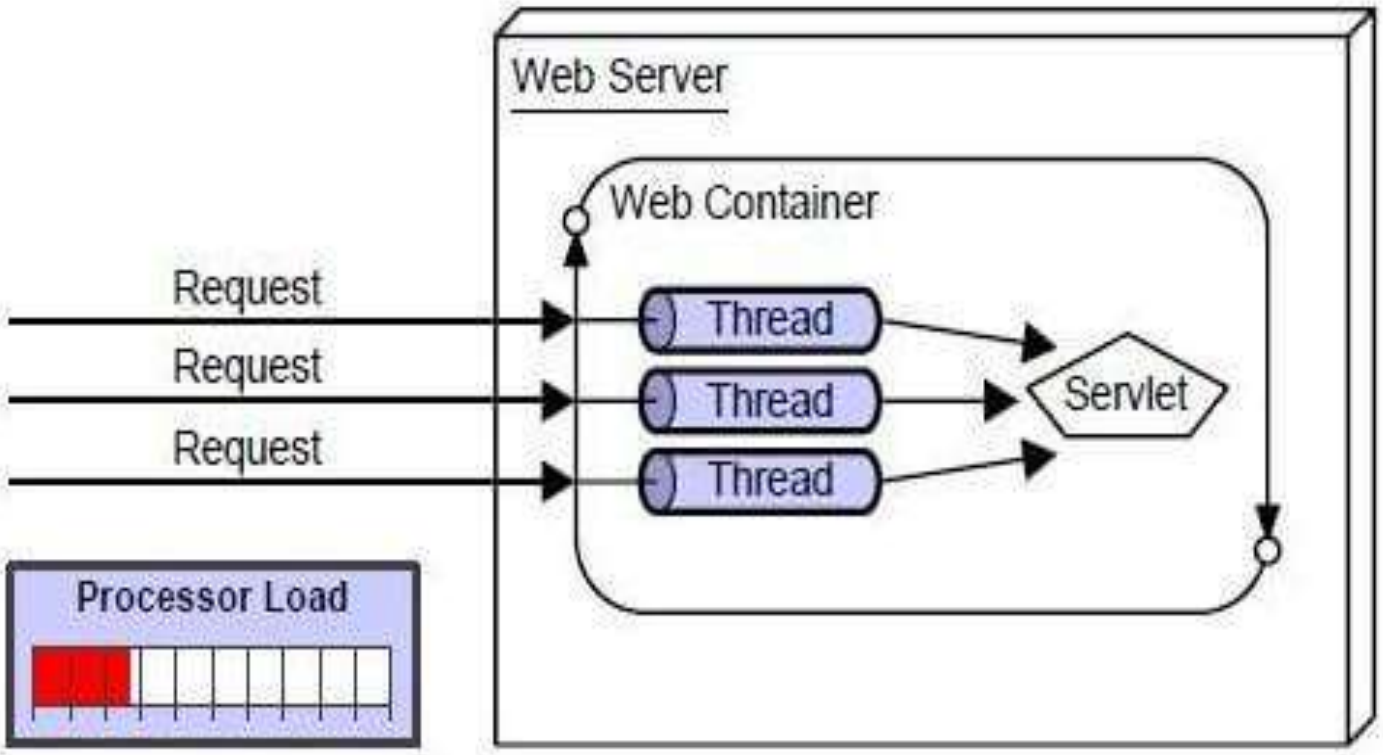
There are many problems in CGI technology:

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

Advantage of Servlet

- The web container creates threads for handling the multiple requests to the servlet.

- Threads have a lot of benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low.


Web Server
Web Container
Request
Request
Request
Thread
Thread
Thread
Servlet
Processor Load

The basic benefits of servlet are as follows:

- **Better performance:** because it creates a thread for each request not process.

- **Portability:** because it uses java language.

- **Robust:** Servlets are managed by JVM so we don't need to worry about memory leak, garbage collection etc.

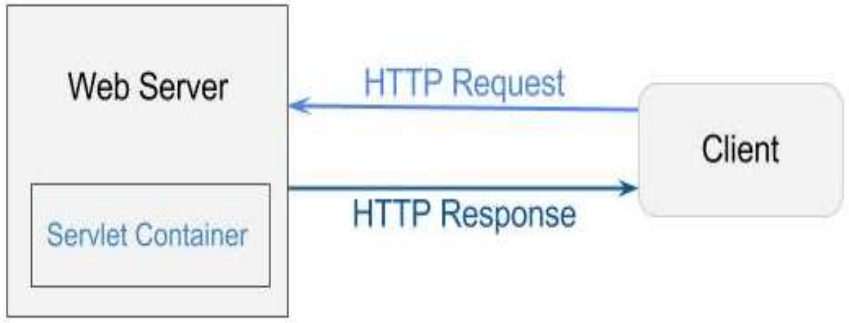
- **Secure:** because it uses java language

Servlet Container

- It provides the runtime environment for JavaEE (j2ee) applications. The client/user can request only a static WebPages from the server. If the user wants to read the web pages as per input then the servlet container is used in java.

  The Servlet Container performs many operations that are given below:

  - Life Cycle Management
  - Multithreaded support
  - Object Pooling
  - Security etc.



Servlet API

- The javax.servlet and javax.servlet.http packages represent interfaces and classes for servlet api.
- The **javax.servlet** package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.
- The **javax.servlet.http** package contains interfaces and classes that are responsible for http requests only.

Interfaces in javax.servlet package

There are many interfaces in javax.servlet package. few are as follows:

- Servlet
- ServletRequest
- ServletResponse
- RequestDispatcher
- ServletConfig
- ServletContext

Classes in javax.servlet package

There are many classes in javax.servlet package. few are as follows:

- GenericServlet
- ServletInputStream
- ServletOutputStream
- ServletRequestWrapper
- ServletResponseWrapper
- ServletRequestEvent
- ServletContextEvent
- ServletException
- UnavailableException

Interfaces in javax.servlet.http package

There are many interfaces in javax.servlet.http package. few are as follows:

- HttpServletRequest
- HttpServletResponse
- HttpSession
- HttpSessionContext

Classes in javax.servlet.http package

There are many classes in javax.servlet.http package. They are as follows:

- HttpServlet
- Cookie
- HttpServletRequestWrapper
- HttpServletResponseWrapper
- HttpSessionEvent
- HttpSessionBindingEvent
- HttpUtils

Servlet Interface

- **Servlet interface provides** common behavior to all the servlets. Servlet interface defines methods that all servlets must implement.
- Servlet interface needs to be implemented for creating any servlet (either directly or indirectly). It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

Methods of Servlet interface

- There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig, which contains initialization and startup parameters for this servlet. The ServletConfig object returned is the one passed to the init method.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

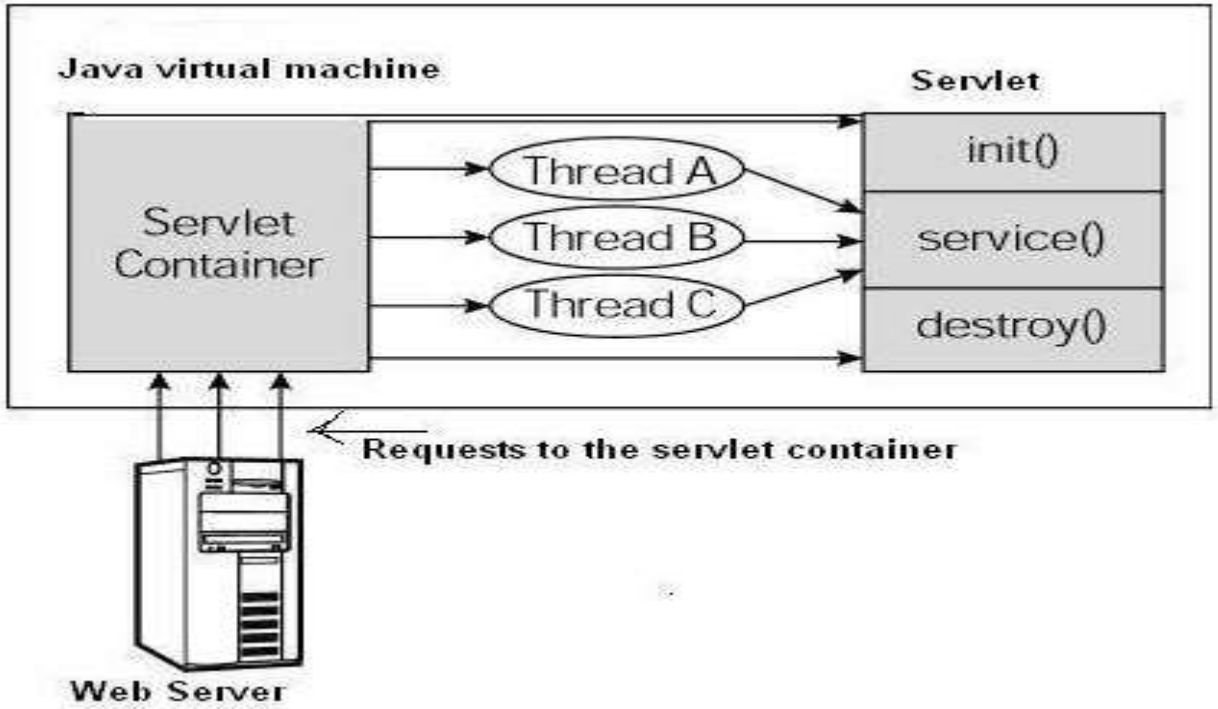
GenericServlet class

- **GenericServlet** classimplements **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 request so it is protocol-independent.

Servlet Life Cycle

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

- The servlet is initialized by calling the **init()** method.
- The servlet calls **service()** method to process a client's request.
- The servlet is terminated by calling the **destroy()** method.
- Finally, servlet is garbage collected by the garbage collector of the JVM.


Java virtual machine
Servlet
Servlet Container
Thread A
Thread B
Thread C
init()
service()
destroy()
Requests to the servlet container
Web Server

The init() Method

- The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards.
- The init() method simply creates or loads some data that will be used throughout the life of the servlet.

```
public void init() throws ServletException {
   // Initialization code...
}
```

The service() Method

- The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.
- Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

```
public void service(ServletRequest request, ServletResponse response)
   throws ServletException, IOException {
}
```

The doGet() Method

A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
   // Servlet code
}
```

The doPost() Method

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
   // Servlet code
}
```

The destroy() Method

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives our servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

```
public void destroy() {
   // Finalization code...
}
```

Example

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

public class First implements Servlet{
ServletConfig config=null;

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

}

public void service(ServletRequest req,Ser
    vletResponse res)
throws IOException,ServletException{
res.setContentType("text/html");

PrintWriter out=res.getWriter();
out.print("<html><body>");
out.print("<b>hello simple 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 2007-1010";
}

}
```

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

public class First extends GenericServlet{
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 generic servlet</b>");
out.print("</body></html>");

}
}
```