3.1 SERVLETS: JAVA SERVLET ARCHITECTURE

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 web server, process the request, produce the response, and then send a response back to the web server.

The properties of Servlets are as follows:

- Servlets work on the server side.
- Servlets are capable of handling complex requests obtained from the web server.

Servlets are grouped under the Advanced Java tree that is used to create dynamic web applications. Servlets are *robust*, *well scalable*, and are primarily used in developing *server-side applications*. If we go a little back in time, we would be able to witness that before the introduction of servlets, *CGI (Common Gateway Interface)* was used. Among several indigenous tasks that a servlet is capable of doing, dynamically performing client requests and responses are most common. Other tasks that a servlet can do effectively are:

- Can easily manage/control the application flow.
- Suitable to implement business logic.
- Can effectively balance the load on the server side.
- Easily generate dynamic web content.
- Handle HTTP Request and Response
- Also act as an interceptor or filter for a specific group of requests.

Servlet Architecture can be depicted from the image itself as provided below as follows:

Types of Servlet

- **Generic Servlets**: These are those servlets that provide functionality for implementing a servlet. It is a generic class from which all the customizable servlets are derived. It is protocol-independent and provides support for HTTP, FTP, and SMTP protocols. The class used is '**javax.servlet.Servlet'** and it only has 2 methods – init() to initialize & allocate memory to the servlet and destroy() to deallocate the servlet.

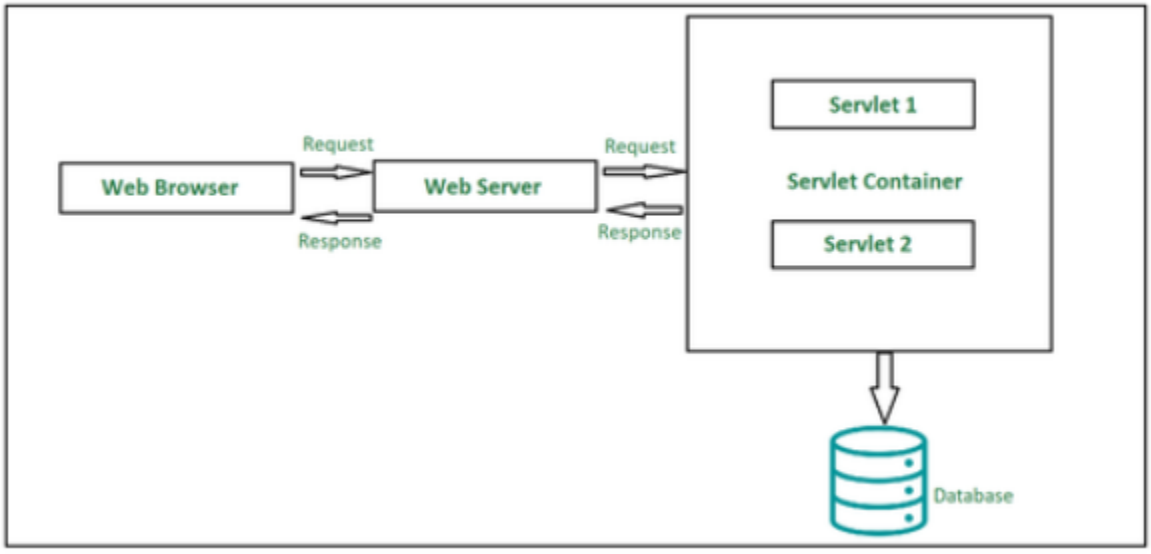
- **HTTP Servlets**: These are protocol dependent servlets, that provides support for HTTP request and response. It is typically used to create web apps. And has two of the most used methods – doGET() and doPOST() each serving their own purpose.

There are three potential ways in which we can employ to create a servlet:

- Implementing Servlet Interface
- Extending Generic Servlet
- Extending HTTP Servlet

Components of Servlet Architecture

Servlet Architecture contains the business logic to process all the requests made by client. Below is the high-level architecture diagram of servlet. Let's see in brief, how does each component add to the working of a servlet.



1. Client

The client shown in the architecture above is the web browser and it primarily works as a medium that sends out HTTP requests over to the web server and the web server generates a response based on some processing in the servlet and the client further processes the response.

2. Web Server

Primary job of a web server is to process the requests and responses that a user sends over time and maintain how a web user would be able to access the files that has been hosted over the server. The server we are talking about here is a software which manages access to a centralized resource or service in a network. There are precisely two types of webservers:

- Static web server
- Dynamic web server

3. Web Container

Web container is another typical component in servlet architecture which is responsible for communicating with the servlets. Two prime tasks of a web container are:

- Managing the servlet lifecycle
- URL mapping

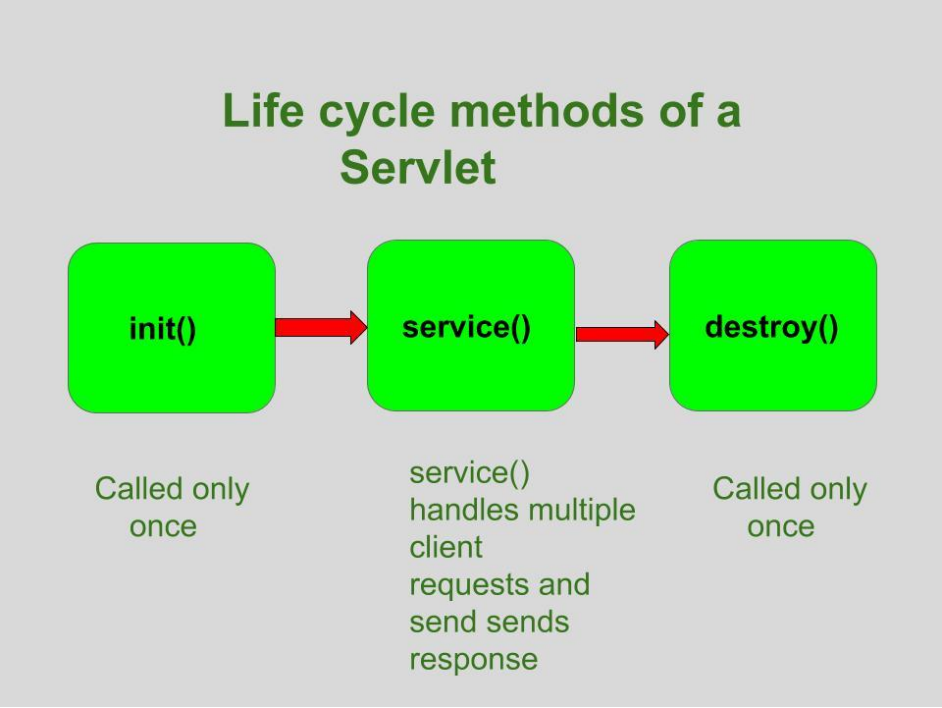
Web container sits at the server-side managing and handling all the requests that are coming in either from the servlets or from some JSP pages or potentially any other file system.

How does a Servlet Request flow?

Every servlet should override the following 3 methods namely:

1. init(): To initalize/instantiate the servlet container.
2. service(): This method acts like an intermediatory between the HTTP request and the business logic to serve that particular request.
3. destroy(): This method is used to deallocate the memory allocated to the servlet.

These methods are used to process the request from the user.



Following are the steps in which a request flows through a servlet which can be observed in the architecture diagram:

- The client sends over a request.
- The request is accepted by the web server and forwarded to the web container.
- In order to obtain the servlet's address, the web container traces *web.xml* file corresponding to the request *URL pattern.*
- By the time above process takes place, the servlet should have been instantiated and initialized. The *init()* method is invoked to initialize the servlet.
- By passing *ServletRequest* and *Response object, public service()* method is called by the container.
- In the next step, the *ServletRequest* and *ServletResponse* objects are type casted to *HttpServletRequest* and *HttpServletResponse* objects by the *public service()* method.
- Now *protected service()* method is called by the *public service()* method.
- The *protected service()* method dispatches the request to the correct handler method based on the type of request.
- When servlet container shuts down, it unloads all the servlets and calls *destroy()* method for each initialized servlet.

Advantages

- Prime functionality of a servlet is that they are independent of server configuration, and they are pretty much compatible with any of the web servers.
- Servlets are also protocol-independent supporting *FTP*, *HTTP*, *SMTP*, etc. protocols to the fullest.
- Until destroyed manually, servlets can be retained in the memory helping process several requests over time. Also, once a database connection is established, it can facilitate process several requests for a database in the very same database session.
- Servlets inherit Java's property of portability and hence are compatible with nearly any web server.
- Servlets are first converted into byte codes and then executed, which helps in increasing the processing time.

Disadvantages

- Designing a servlet can be pretty laborious.
- Exceptions need to be handled while designing a servlet since they are not thread safe.
- Developers may need additional skills to program a servlet.