Java Servlet

Definition for Java Servlet

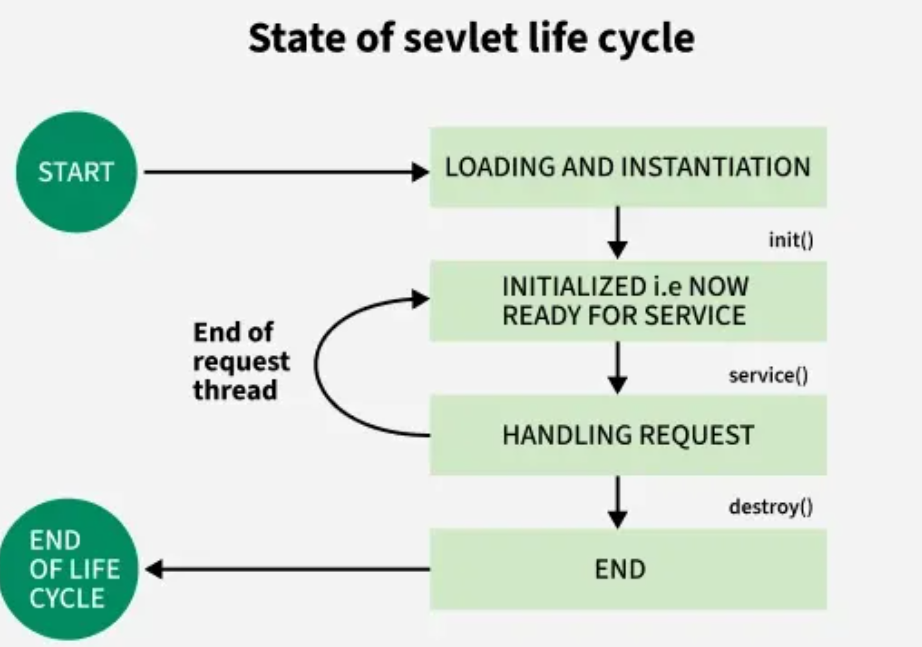
* Java Servlet is a Java program that runs on a Java-enabled web server or application server.
* It handles client requests, processes them and generates responses dynamically.
* Servlets are the backbone of many server-side Java applications due to their efficiency and scalability.

**Benefits of Java Servlets:**

* Faster execution as Servlets do not create new processes for each request.
* Write-once, run-anywhere feature of Java.
* Single instance handles multiple requests.
* Easily integrates with databases using JDBC.
* It inherits robust security features from web servers.

Life Cycle of a Servlet (or) **Java Servlets Architecture**

* understand the life cycle of the Servlet object, which is actually understanding how the Servlet container manages the Servlet object.



**Java servlets container play a very important role.**

* It is responsible for handling important tasks like load balancing, session management and resource allocation, it make sure that all the requests are process efficiently under high traffic.
* The container distribute requests accross multiple instances, which helps improve the system performance.

**Servlet Life Cycle Methods**

There are three life cycle methods of a Servlet:

* init()
* service()
* destroy()

A Java servlet consists of at least four methods, each of which is called during the life cycle of the Java servlet. These methods are init(), service(), the appropriate request method(doget,post…), and destroy().

init():

* The init() method is called automatically when the Java servlet is created
* This method is called by the Servlet container to indicate that this Servlet instance is instantiated successfully and is about to put into the service.
* A Servlet life begins from this method. This method is called only once to load the servlet,
* The init() method is used to initialize variables and objects that are used throughout the Java servlet.
* The init() method does not take an argument, returns a void, and throws a servletException.
* In the real world, the init() method is used to read server-specific initialization parameters such as performance parameters and database settings.
* **override the init() method**
* We can override the init() method to include statements that are executed once during the life cycle of the Java servlet and not for each request to use the Java servlet.
* The below code shows how to override the init() method to read server-specific initialization parameters.
* The servletConfig object contains the methods necessary to interact with server-specific initialization parameters.
* In this example, an instance of the servletConfig object is created and is called con.
* Next, the getInitParameter() method is called to retrieve the first initialization parameters, which is returned as a String object called p1.
* Server-specific initialization parameters are used with the deployment descriptor to control the behavior of the Java servlet and Java Server Pages,

public void init() throws servletException {

servletConfig con = getServletConfig();

String p1 = con.getInitParameter("parameter1");

}

**2. service() Method**

 The **service() method** runs every time a client makes a request to the servlet.

 It checks the type of request (GET, POST, etc.) and forwards it to the correct method (**doGet()**, **doPost()**, etc.).

 Since multiple clients can send requests at the same time, **multiple threads** may run together.

 This can cause **data conflicts** if shared resources are not properly synchronized, especially with **static variables**.

 To avoid these issues, the **SingleThreadModel** can be used. This ensures:

* Only **one request is processed at a time**.
* Other requests wait in a **queue** until the current one is finished.

**Important Key Points about service() Method:**

* This method provides the connection between client and server
* The web server calls the service() method to handle requests coming from the client( web browsers) and to send response back to the client
* This method determines the type of Http request (GET, POST, PUT, DELETE)
* This method also calls various other methods such as doGet(), doPost(), doPut(), doDelete()
* This method accepts two parameters.
  + req is the ServletRequest object which encapsulates the connection from client to server
  + res is the ServletResponse object which encapsulates the connection from server back to the client

**3. destroy() Method**

This method runs only once during the lifetime of a Servlet and signals the end of the Servlet instance

* The destroy() method is called only once.
* It is called at the end of the life cycle of the servlet.
* This method performs various tasks such as closing connection with the database, releasing memory allocated to the servlet, releasing resources that are allocated to the servlet and other cleanup activities.

**Execution of Java Servlets**

* Servlet is created
* Servlet is initialized
* Servlet is ready to service
* Servlet is servicing
* Servlet is not ready to service
* Servlet is destroyed

Execution of Servlets basically involves Six basic steps:

* The Clients send the request to the Web Server.
* The Web Server receives the request.
* The Web Server passes the request to the corresponding servlet.
* The Servlet processes the request and generates the response in the form of output.
* The Servlet sends the response back to the webserver.
* The Web Server sends the response back to the client and the client browser displays it on the screen.

**What is CGI (Common Gateway Interface)?**

In CGI ,server has to create and destroy the process for every request

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.

So, in CGI server has to create and destroy the process for every request.

It's easy to understand that this approach is applicable for handling few clients but as the number of clients increases, the workload on the server increases and so the time is taken to process requests increases.

**Difference Between Java Servlets and CGI**

The following table explains the difference between the servlet and CGI:

| **Basis** | **Servlet** | **CGI** |
| --- | --- | --- |
| Approach | It is thread based i.e. for every new request new thread is created. | It is process-based i.e. for every new request new process is created. |
| Language Used | The codes are written in JAVA programming language. | The codes are written any programming language. |
| Object-Oriented | Since codes are written in Java, it is object oriented and the user will get the benefits of OOPs | Since codes are written in any language, all the languages are not object-oriented thread-based. So, the user will not get the benefits of OOPs |
| Portability | It is portable. | It is not portable. |
| Persistence | It remains in the memory until it is not explicitly destroyed. | It is removed from the memory after the completion of the process-basedrequest. |
| Server Independent | It can use any of the web-server. | It can use the web-server that supports it. |
| Data Sharing | Data sharing is possible. | Data sharing is not possible. |
| Link | It links directly to the server. | It does not link the web server directly to the server. |
| HTTP server | It can read and set HTTP servers. | It can neither read nor set HTTP servers. |
| Cost | Construction and destruction of new threads are not costly. | Construction and destruction of the new processes are costly. |
| Speed | Its can speed is slower. | It can speed is faster. |
| Platform dependency | It can be Platform Independent | It can be Platform dependent. |

**Key Classes and Interfaces**

Various classes and interfaces present in these packages are:

| **Component** | **Type** | **Package** |
| --- | --- | --- |
| Servlet | Interface | jakarta.servlet.\* |
| ServletRequest | Interface | jakarta.servlet.\* |
| ServletResponse | Interface | jakarta.servlet.\* |
| GenericServlet | Class | jakarta.servlet.\* |
| HttpServlet | Class | jakarta.servlet.http.\* |
| HttpServletRequest | Interface | jakarta.servlet.http.\* |
| HttpServletResponse | Interface | jakarta.servlet.http.\* |
| Filter | Interface | jakarta.servlet.\* |
| ServletConfig | Interface | jakarta.servlet.\* |

**Servlet Container**

Servlet container, also known as Servlet engine, is an integrated set of objects that provide a run time environment for Java Servlet components. It is a system that manages Java Servlet components on top of the Web server to handle the Web client requests.   
  
Services provided by the Servlet container:

* **Network Services:** Loads a Servlet class. The loading may be from a local file system, a remote file system or other network services. The Servlet container provides the network services over which the request and response are sent.
* **Decode and Encode MIME-based messages**: Provides the service of decoding and encoding MIME-based messages.
* **Manage Servlet container:**Manages the lifecycle of a Servlet.
* **Resource management**: Manages the static and dynamic resources, such as HTML files, Servlets and JSP pages.
* **Security Service:** Handles authorization and authentication of resource access.
* **Session Management:** Maintains a session by appending a session ID to the URL path

**Real-World Use Cases of Java Servlets**

* **E-Commerce Platforms**: Dynamic catalog generation and order processing.
* **Banking Applications**: Secure user sessions and real-time transaction processing.
* **Content Management Systems**: Handling file uploads and dynamic content delivery.