**Introduction to Java Servlets:--**

Today, we all are aware of the need to create **dynamic web pages** i.e. the ones that can change the site contents according to the time or can generate the content according to the request received from the client. If you like coding in Java, then you will be happy to know that using Java there also exists a way to generate dynamic web pages and that way is **Java Servlet**. But before we move forward with our topic let’s first understand the need for server-side extensions.

**What is Java Servlet?**

Java 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.

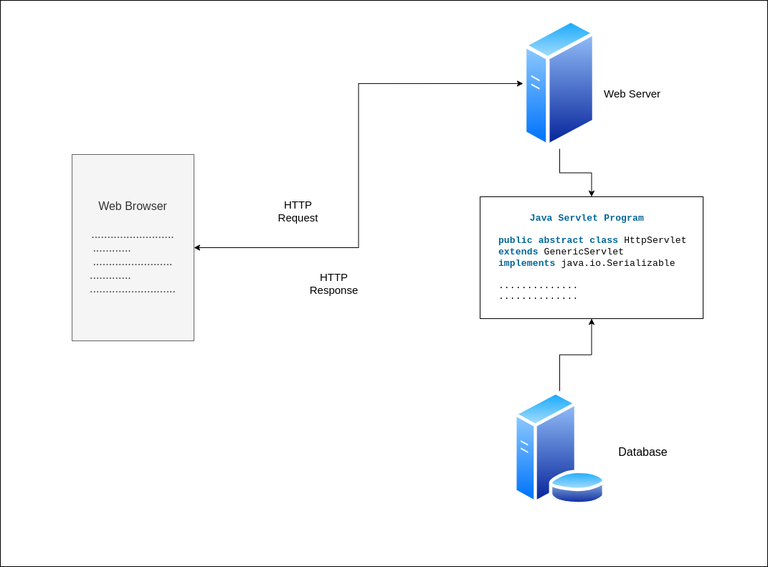
**Properties of Java Servlet**

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.

**Java Servlets Architecture**

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



**Execution of Java Servlets**

Execution of Servlets basicallyinvolves Six basic steps:

1. The Clients send the request to the Web Server.
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.

Now let us do discuss eccentric point that why do we need For Server-Side extensions?

**Need of Server-Side Extensions**

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 (nowdays known as – ‘Jakarta EE’)** which sets standards for creating dynamic Web applications in Java.

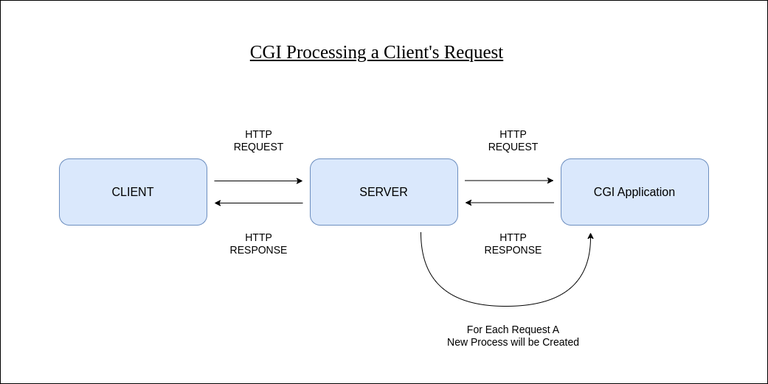
Before learning about something, it’s important to know the need for that something, it’s not like that this is the only technology available for creating Dynamic Web Pages. The Servlet technology is similar to other Web server extensions such as **Common Gateway Interface (**CGI) scripts and **Hypertext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

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

**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**

| **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. |

**Servlets APIs**

Servlets are built from two packages:

* javax.servlet(Basic)
* javax.servlet.http(Advance)

Various classes and interfaces present in these packages are:

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

**Advantages of a Java Servlet**

* Servlet is **faster** than CGI as it doesn’t involve the creation of a new process for every new request received.
* Servlets, as written in Java, are platform independent.
* Removes the overhead of creating a **new process** for each request as Servlet doesn’t run in a separate process. There is only a single instance that handles all requests concurrently. This also saves the memory and allows a Servlet to easily manage the client state.
* It is a server-side component, so Servlet inherits the **security** provided by the Web server.
* The **API** designed for Java Servlet automatically acquires the advantages of the Java platforms such as platform-independent and portability. In addition, it obviously can use the wide range of APIs created on Java platforms such as **JDBC** to access the database.
* Many Web servers that are suitable for personal use or low-traffic websites are offered for free or at extremely **cheap costs** eg. Java servlet. However, the majority of commercial-grade Web servers are rather expensive, with the notable exception of Apache, which is free.

**The 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. In simple words, 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.

**Conclusion**

**Java Servlets** are crucial components for defining business logic and handling complex web requests. These components promote the dynamic development of a web site and has a lot of potential to change the application dynamics. Here are some of the key features we learn in this article:

* Java Servlets aids in the development of server-side application development and provide a way to deal with dynamic content to build robust and interactive websites.
* As soon as the request reaches the web server, a servlet instance is initialized using **init()**method, and the HTTP request depending on its type can be handled by doGet() or doPost() methods.
* Servlets make our web application more responsive and efficient as it doesn’t create and destroy a process every time a request is received.

**FAQs on Java Servlet**

**Q1. What is the Lifecycle of Java Servlet?**

*Java Servlet has a pre-defined lifecycle starting from initializer (memory allocation) until the object is destructed (memory deallocated). It takes place in following steps :*

1. *init() – is called as soon as the request is received by the web server and a new servlet instance is initialized.*
2. *service() – to handle client request and redirect the request to an appropriate doGet() or doPost()*
3. *destroy() – called when the request is handled, response sent back to the client and finally the memory allocated to the servlet is deallocated.*

**Q2. How to configure a Java Servlet?**

*Configuration of Java Servlet is defined inside a* ***deployment descriptor*** *file –* ***web.xml.*** *Although, it is legacy now as most servlet based applications nowadays used Annotation based configuration to configure a Java Servlet using @WebServlet Annotation.*

*Example : @WebServlet(name = “MyServlet”, urlPatterns = “/myServlet”)*

**Q3. What is a Servlet Filter?**

*A Servlet Filter is an additional component for performing pre-post processing work on the web requests like logging, monitoring, debugging and troubleshooting.*