Pre-Spring *Hanoon🍓*

CGI (Common Gateway Interface) :

## ** Client Request: The client (browser) sends an HTTP request to the web server.**

## ** Server Processing: The web server receives and identifies the request as a CGI request.**

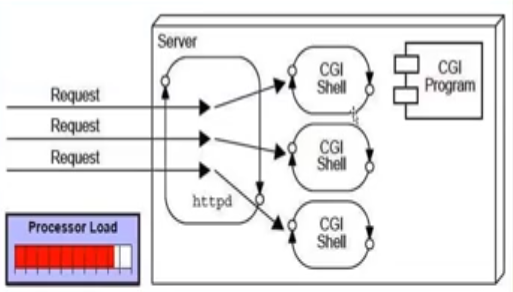
## ** CGI Script Execution: The server executes the CGI script, which can be written in languages like Perl, Python, or C (dependent language).**

## ** Data Processing: The CGI script processes the request, interacts with databases or other resources, and generates the required output.**

(by starting a new process 🡪 processor load)

## ** Response Generation: The CGI script sends the output, usually in the form of HTML, back to the server.**

## ** Client Response: The web server sends the generated HTML back to the client as the HTTP response. (more users 🡪 load on the server (slow)).**

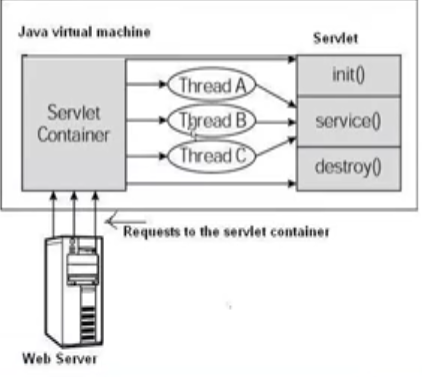


Servlet:

* **Servlets are server-side Java programs designed to handle HTTP requests and generate dynamic web content efficiently.’**
* **Rely on the JVM (Java Virtual Machine) and benefit from multithreading—where each client request triggers the creation of a new thread.**
* **secured, platform-independent, and can run on any web container (like Tomcat, JBoss, or Glassfish) across various operating systems (Windows, Linux, Solaris, Mac, etc.)**

**Servlet Workflow:**

1. **Client Request: A client (browser) sends an HTTP request to the web server.**
2. **Server Processing: The server identifies the request as a Servlet request.**
3. **Servlet Loading: If the servlet isn’t already loaded, the server loads and initializes it.**
4. **Request Handling:**
   * **The server creates instances of HttpServletRequest and HttpServletResponse (wrapper classes).**
   * **Developers can extend these classes to customize request/response objects by overriding specific methods.**
5. **Servlet Execution: The server calls the service() method of the servlet, passing in the request and response objects.**
6. **Data Processing: The servlet interacts with databases or other resources to generate the desired output.  
   - Multithreading allows handling many requests at once, reducing processor load.**
7. **Response Generation: The servlet sends the HTML response back to the server.**
8. **Client Response: The web server sends the HTTP response to the browser.**

****

Web Server: Handles HTTP requests from client browsers and responds with HTML content or other types of data, Use HTTP Protocol🡪 Apache HTTP Server.

Servlet/ Web Container:

- Communication Support: Servlet Container provides a seamless way of communication between **web clients (browsers)** and web components like **Servlets** and **JSPs**, Thanks to the container, developers don’t need to manually create **server sockets** to listen for requests, parse them, or generate responses.

**. Lifecycle and Resource Management**

* The container **manages the servlet’s lifecycle**, from loading it into memory, initializing it, handling requests, and finally destroying it when no longer needed.
* It also offers tools like **JNDI** for **resource pooling and management**, optimizing performance.

- Multithreading Support:

- The container creates **a new thread for each request**, providing independent **request and response** objects for processing.

**- Servlets are loaded only once** and reused for multiple requests, which improves **performance and reduces memory usage**.

- JSP Support: JSPs don’t look like normal Java classes but every JSP in the application is compiled by a container and converted to a Servlet and then the container manages them like other servlets.

- Miscellaneous Task:

The Servlet Container handles:

* **Resource pooling** and **memory optimization**.
* Running the **garbage collector** to free up unused resources.
* Managing **security configurations**.
* Supporting **multiple applications** and **hot deployment** (updating apps without stopping the server).

**Web Application?**

A web application is a software application that runs on a web server rather than being installed on the user's local computer. It is accessed via a web browser over a network, such as the Internet or an intranet.

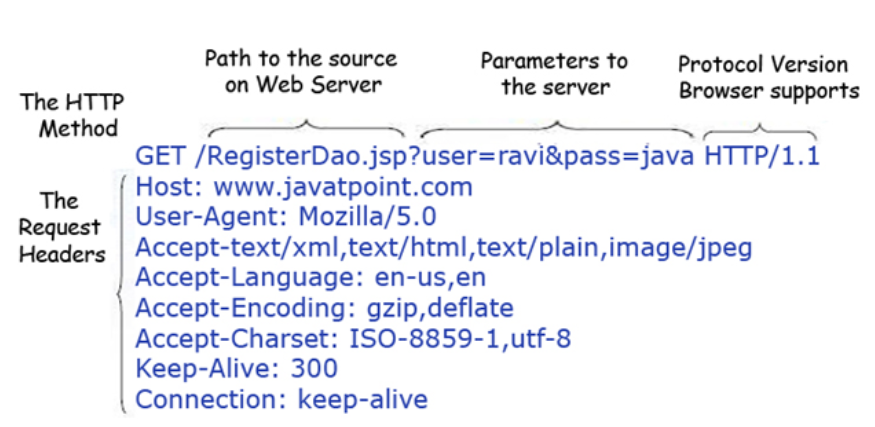
* They use a mix of **server-side scripts** (like Java Servlets) and **client-side scripts** (like JavaScript) to offer dynamic, interactive user experiences.
* Web applications provide a complete environment for developing **enterprise-level solutions**, with features such as:
* **Advanced security management** for **authentication** and **authorization**.
* **Integration with databases** and **legacy systems**.

Idempotent (save) HTTP methods: (return the same result every time)

* + GET, PUT, DELETE, HEAD, OPTIONS.

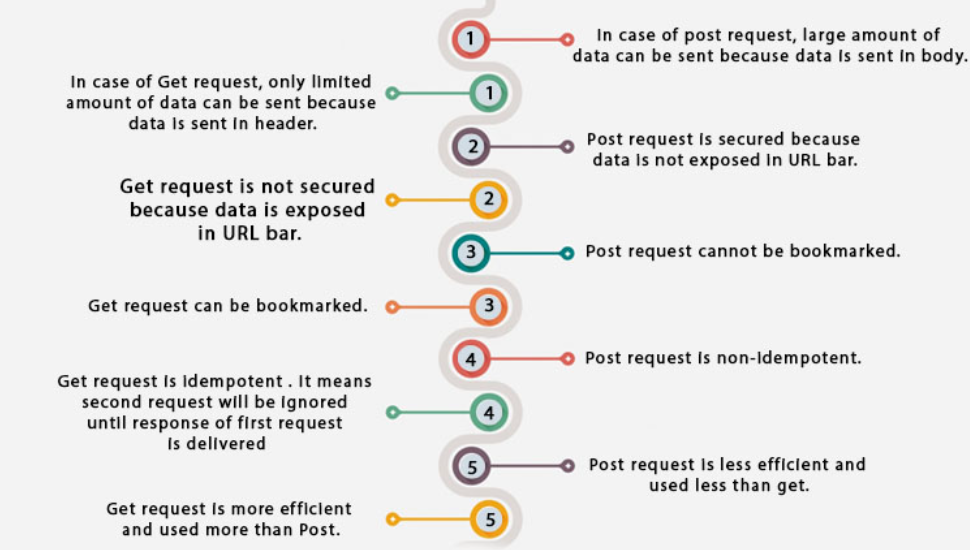
**Difference Between GET and POST Requests**

* **GET Request:**
  + **Purpose:** Retrieve data from the server.
  + **Parameters:** Parameters are appended to the URL as query strings.
  + **Idempotent:** Multiple identical GET requests should have the same effect as a single request.
  + **Caching:** GET requests can be cached by the browser.
  + **Size Limit:** URL length limits apply (typically around 2000 characters).



* **POST Request:**
  + **Purpose:** Send data to the server, often to create or update resources.
  + **Parameters:** Parameters are sent in the body of the request, not the URL (more secure).
  + **Non-idempotent:** Multiple identical POST requests may create duplicate entries or affect the server state.
  + **Caching:** POST requests are generally not cached.
  + **Size Limit:** No practical limit on the size of data sent.

****



When a servlet request is made, the web server receives:

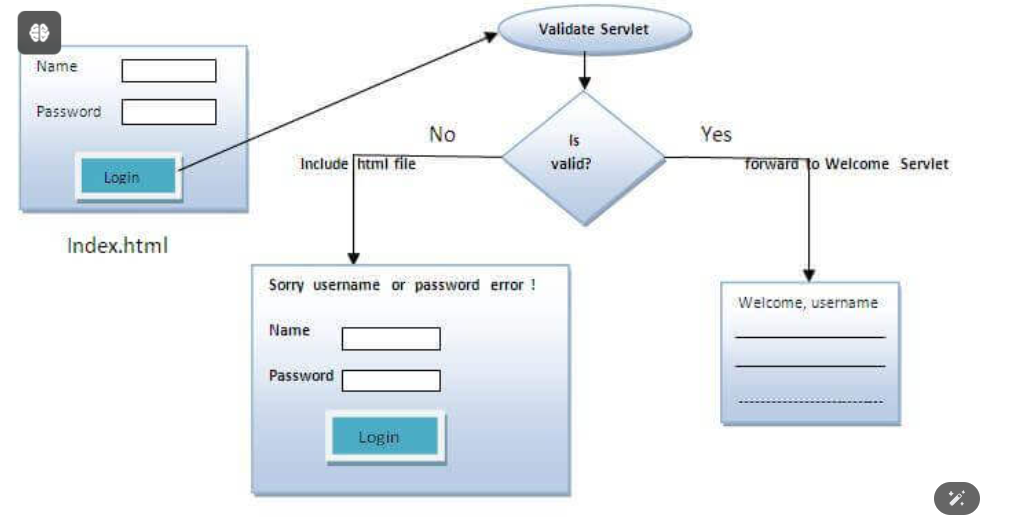
* HTTP Method: GET, POST, etc.
* Request URL: The URL used to make the request.
* Headers: Information about the client, content type, etc.
* Parameters: Data sent with the request (query parameters or form data).

Body: The body contains the data being sent for POST requests.

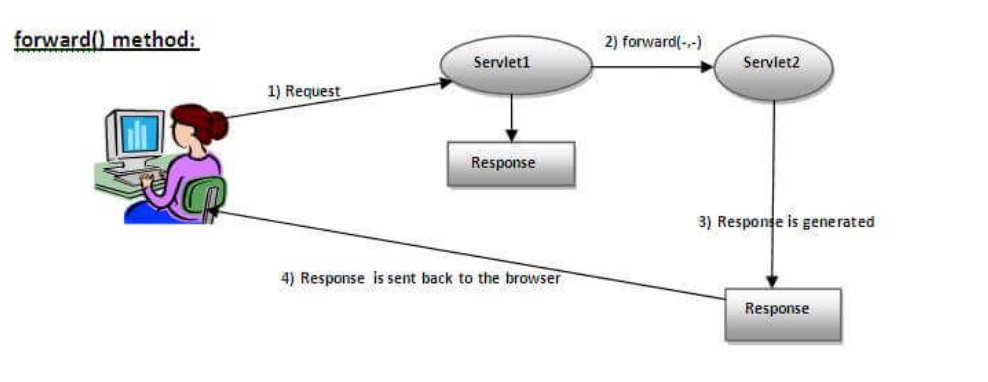
RequestDispatcher and sendRedirect() :

Servlet Collaboration:

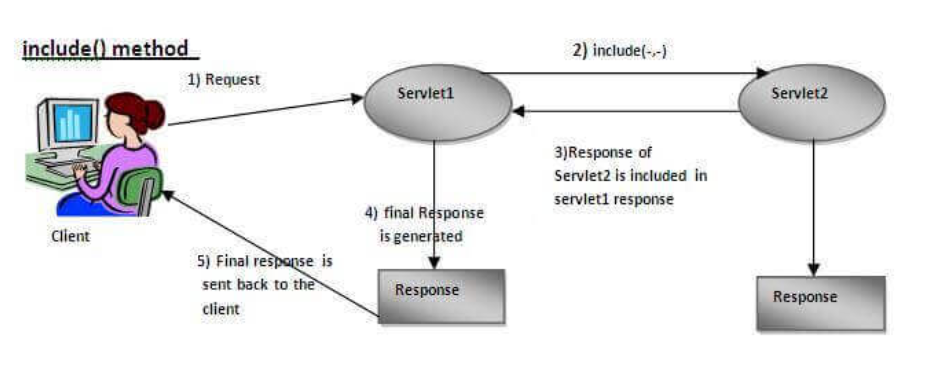
* RequestDispatcher: Used to forward requests and responses between servlets or JSPs within the same server. It doesn’t change the URL in the browser.

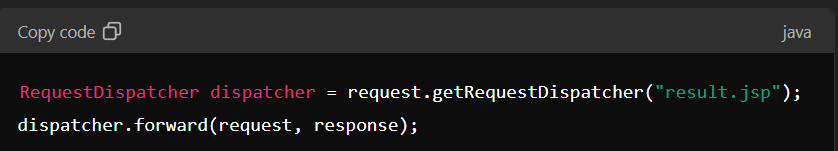


* forward(request, response): forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

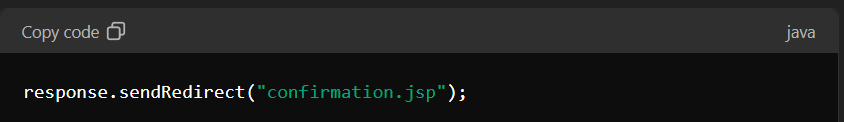


* include(request, response): includes content of resource in the response.

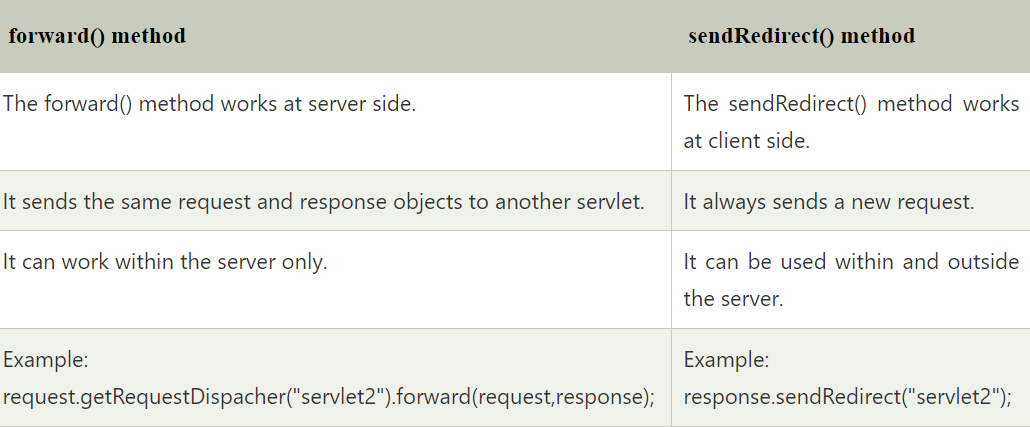




* sendRedirect(): Sends a new request to the client, causing the browser to make a new request to a different URL. The URL in the browser changes.



**Differences:**



Difference Between ServletConfig and ServletContext

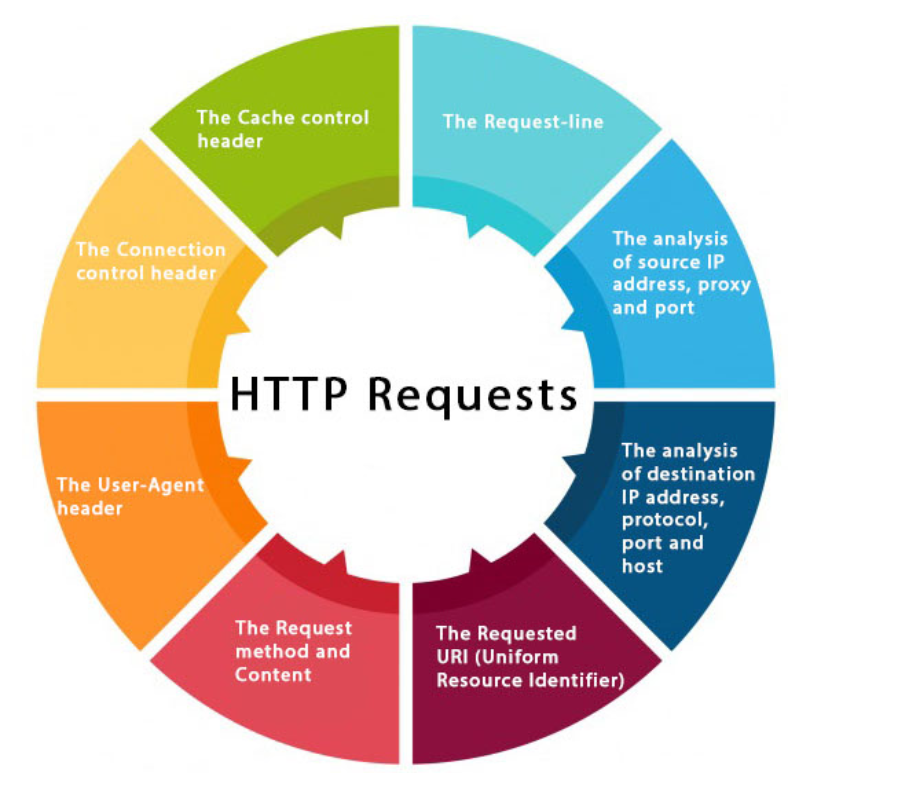
* **ServletConfig:** Provides servlet-specific initialization parameters(init). Each servlet has its own ServletConfig🡪 DatabaseURL.
* **ServletContext:** Provides application-wide information and parameters(init for all servlets). It is shared across all servlets in the web application(unique for complete application)🡪Application name.

Ways to Maintain User State :

* Cookies: Small pieces of data stored on the client’s browser.
* Session: Server-side storage of user-specific data that persists across multiple requests.
* Hidden Form Fields: Data embedded within HTML forms.
* URL Rewriting: Appending session data to URLs.

\*Most Used Approach: Sessions are commonly used due to their flexibility and security.

HTTP Requests:



Servlet Containers: part of the web server which can be run in a separate process. We can classify the servlet container states into three types:

* Standalone: It is typical Java-based server in which the servlet container and the web servers are the integral part of a single program. For example:- Tomcat running by itself.
* In-process: It is separated from the web server, because a different program runs within the address space of the main server as a plug-in. For example, Tomcat is running inside the JBoss.
* Out-of-process: The web server and servlet container are different programs which are run in a different process. For performing the communications between them, web server uses the plug-in provided by the servlet container, Apache HTTP Server و Tomcat كعمليات منفصلة

Servlet Interfaces:

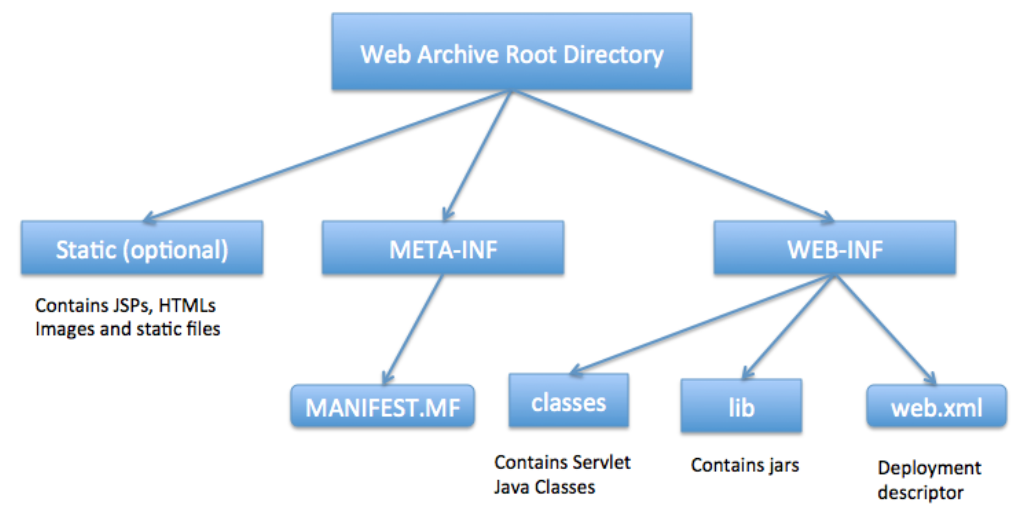
* provides common behavior to all the servlets.Servlet interface defines methods that all servlets must implement.
* Provides 3 life cycle methods (to init() , to service() the requests, to destroy() the servlet ), and 2 non-life cycle methods.
* init(): invoked by the web container only once and initializes the servlet.
* service(): respond to requests invoked by the web container.
* destroy(): invoked only once and destroy servlet.
* getServletConfig(): returns the object of ServletConfig.
* getServletInfo(): info like writer, copyright, version…

War File:Contains file of web project (may have servlet, xml, jsp, image, css, js … files.

* Saves time: It takes less time to transfer files from client to server.

MIME: “Content-Type” response header, the server sends MIME type to the client to let them know the kind of data is sending, MIME types usually used is “text/html, text/xml, application/xml”

\*Web Applications are modules that run on the server to provide both static and dynamic content to the client browser. Apache webserver supports PHP and we can create a web application using PHP. Java provides web application support through Servlets and JSPs that can run in a Servlet container and provide dynamic content to the client browser. Java Web Applications are packaged as Web Archive (WAR)



Difference between PrintWriter and ServletOutputStream?

* Different two ways to write data to the HTTP response.
* PrintWriter 🡪 for writing character data (text-based content) such as HTML, XML, JSON.
* ServletOutputStream🡪 Best for writing binary data (e.g., images, files).

Servlet Filters: objects perform filtering tasks on either the request to a resource, the response from a resource, or both, used to process requests before they reach a servlet and to process responses before they are sent to the client. Filters are defined in the web.xml configuration file or via annotations in modern Java EE applications.

* Authentication and Authorization: verify if a user has a valid session before allowing access to secure areas of a website
* **Data Compression**: compress response data to reduce the amount of data sent over the network, improving performance and reducing load times.
* **Logging and Auditing**: used to log request and response details, such as the time taken to process a request, the request URI, and the response status.
* **Input Validation**: validate input data from a request to ensure it meets certain criteria before passing it to the servlet. This can help prevent invalid or malicious data from reaching the application logic.
* **Request/ Response Modification**: Filters can modify the request before it reaches the servlet. For example, they can add headers or parameters to the request & modify the response before it is sent to the client. For instance, they can add headers, change content types, or manipulate the response body.
* Cross-Site Request Forgery (CSRF) Protection: Filters can help protect against CSRF attacks by verifying that requests are coming from trusted sources.

Servlet Listeners: Java classes that listen to specific events in the lifecycle of a web application. They allow you to perform custom actions when these events occur. Listeners are registered in the web.xml configuration file or using annotations.

* ServletContextListener: Listens to changes in the servlet context, such as when the web application starts or stops.
* HttpSessionListener: Listens to HTTP session lifecycle events, such as when a session is created or destroyed.
* ServletRequestListener: Listens to HTTP request lifecycle events, such as when a request is received or completed
* ServletRequestAttributeListener: Listens to changes to requests' attributes.

Servlet3:

* File Uploads - Adding Web Components dynamically
* Annotations-Based Configuration
* Asynchronous Processing - Improved Servlet API
* Declarative Security - Enhanced Request and Response APIs

[**Ways for servlet authentication?**](https://www.digitalocean.com/community/tutorials/servlet-interview-questions-and-answers?utm_medium=affiliates&utm_source=impact&utm_campaign=123201&utm_content=237721&irgwc=1&irclickid=xf4R2C0llxyPRPNwv9wC4xpkUkCz%3A1ztMxeFS40#what-are-different-ways-for-servlet-authentication)

* **HTTP Basic Authentication:** The client sends an HTTP request with an Authorization header containing a base64-encoded **username** and password.
* **HTTP Digest Authentication:** he client and server use a challenge-response mechanism. The password is not sent in plain text; instead, a hashed value is sent, which includes a nonce (a random value) to protect against replay attacks.
* **HTTPS Authentication:** Uses HTTPS (HTTP over SSL/TLS) to encrypt the communication between the client and server. It ensures that data, including credentials, is transmitted securely over the network.
* **Form Based Login:** Users authenticate by submitting a form with their username and password. The form posts to a login servlet or endpoint that validates the credentials**.**

**\***[**To get the IP address of client in servlet**](https://www.digitalocean.com/community/tutorials/servlet-interview-questions-and-answers?utm_medium=affiliates&utm_source=impact&utm_campaign=123201&utm_content=237721&irgwc=1&irclickid=xf4R2C0llxyPRPNwv9wC4xpkUkCz%3A1ztMxeFS40#how-to-get-the-ip-address-of-client-in-servlet)**:** request.getRemoteAddr()

### [How to upload file on server](https://www.digitalocean.com/community/tutorials/servlet-interview-questions-and-answers?utm_medium=affiliates&utm_source=impact&utm_campaign=123201&utm_content=237721&irgwc=1&irclickid=xf4R2C0llxyPRPNwv9wC4xpkUkCz%3A1ztMxeFS40#write-a-servlet-to-upload-file-on-server)

* Servlet: The FileUploadServlet handles the file upload process by receiving the file, saving it to the server, and sending a response.
* HTML Form: The upload.html file provides a user interface for selecting and uploading files.
* Configuration: Ensure the servlet is correctly mapped in web.xml or through annotations.

[**How to get the server information in a servlet**](https://www.digitalocean.com/community/tutorials/servlet-interview-questions-and-answers?utm_medium=affiliates&utm_source=impact&utm_campaign=123201&utm_content=237721&irgwc=1&irclickid=xf4R2C0llxyPRPNwv9wC4xpkUkCz%3A1ztMxeFS40#how-to-get-the-server-information-in-a-servlet)?

* To get the servlet information in a servlet through servlet context object. `getServletContext().getServerInfo()`.

[**How to get the actual path of servlet in server?**](https://www.digitalocean.com/community/tutorials/servlet-interview-questions-and-answers?utm_medium=affiliates&utm_source=impact&utm_campaign=123201&utm_content=237721&irgwc=1&irclickid=xf4R2C0llxyPRPNwv9wC4xpkUkCz%3A1ztMxeFS40#how-to-get-the-actual-path-of-servlet-in-server)

- getServletContext().getRealPath(request.getServletPath()).

Deployment descriptor(web.xml):

The deployment descriptor is a configuration file for the web application and its name is web.xml, it resides in the WEB-INF directory. Servlet container uses this file to configure web application servlets, servlet config params, context init params, filters, listeners, welcome pages and error handlers. With servlet 3.0 annotations, we can remove a lot of clutter from web.xml by configuring servlets, filters, and listeners using annotations.

EncodeURL & encodeRedirectURL:

* encodeURL: encode creating URLs that will be used in HTML or JavaScript within the same response.
* encodeRedirectURL: typically used when redirecting a client to a different URL.

URL Rewriting: a technique used to ensure that session information is included in URLs, especially where cookies are not available or are disabled. It is a method of session tracking that appends a session identifier to URLs to maintain state across requests.

50 interview Questions🥳

https://www.digitalocean.com/community/tutorials/servlet-interview-questions-and-answers?utm\_medium=affiliates&utm\_source=impact&utm\_campaign=123201&utm\_content=237721&irgwc=1&irclickid=xf4R2C0llxyPRPNwv9wC4xpkUkCz%3A1ztMxeFS40#why-do-we-have-servlet-filters