Pre-Spring

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.**

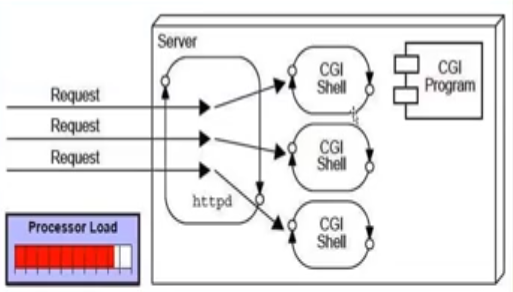
## ** CGI Script Execution: The server executes the CGI script, which can be written in languages like Perl, Python, or C (dependent language).**

## ** 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:

* **Server-side Java programs (java language only) that run on the server to generate dynamic web content efficiently.**
* **Depends on JVM.**
* **Using the benfits of multithreading & for each request, a new thread is created**
* **Secured.**
* **the garbage collector of JVM collects garbage.**

- Servlets and platform and system independent, the web application developed with Servlet can be run on any standard web container such as Tomcat, JBoss, Glassfish servers and on operating

systems such as Windows, Linux, Unix, Solaris, Mac, etc.

 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 servlet request.

 Servlet Loading: If the servlet is not already loaded, the server loads the servlet and initializes it.

 Request Handling: The server creates an instance of HttpServletRequest and HttpServletResponse.

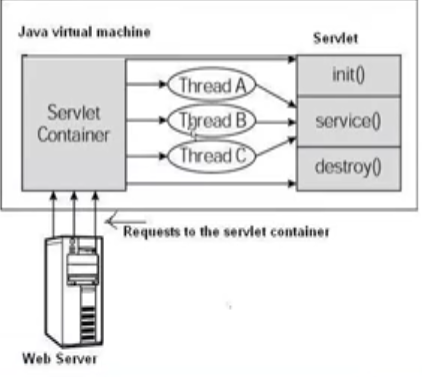
 Servlet Execution: The server calls the servlet's service() method, passing the request and response objects.

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

\*Can receive many requests & use Multithread to process (less processor load).

 Response Generation: The servlet 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.

****

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 easy way of communication between web clients (Browsers) and the servlets and JSPs. Because of the container, we don’t need to build a server socket to listen for any request from the web client, parse the request and generate a response.

- Lifecycle and Resource Management: Servlet Container takes care of managing the life cycle of servlet. From the loading of servlets into memory, initializing servlets, invoking servlet methods and to destroy them. The container also provides utility like JNDI for resource pooling and management.

- Multithreading Support: The container creates a new thread for every request to the servlet and provides their request and response objects to the processing. So servlets are not initialized for each request and save time and memory.

- 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: Servlet container manages the resource pool, perform memory optimizations, execute garbage collector, provides security configurations, support for multiple applications, hot deployment, and several other tasks behind the scene that makes a developer life easier.

**Web Application?**

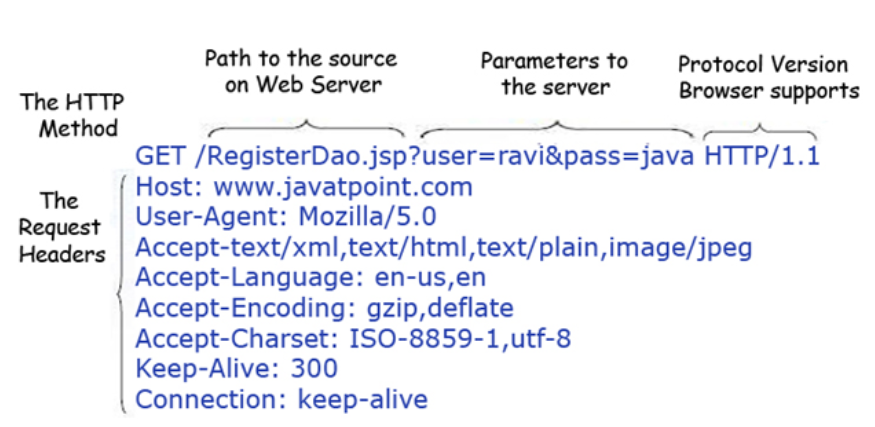
A web application is a software application that runs on a web server rather than being installed on the user's local computer. It is accessed via a web browser over a network, such as the Internet or an intranet. Web applications often use a combination of server-side scripts (Java servlets) and client-side scripts (JavaScript) to provide a dynamic user experience, provides a more comprehensive environment for developing and deploying enterprise applications. It includes everything a servlet container does, plus additional enterprise-level features like Security Management for authentication and authorization, Integration with database and legacy systems.

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

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

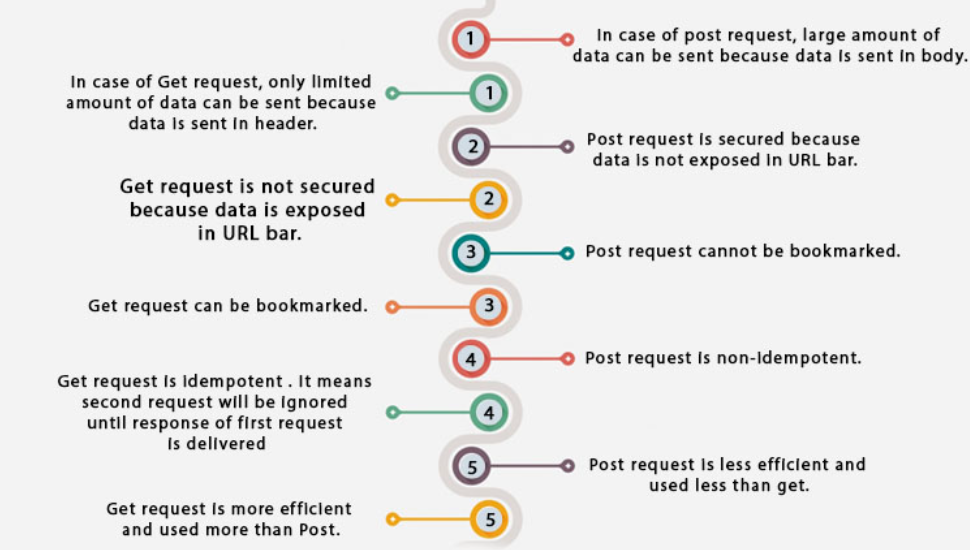
**Difference Between GET and POST Requests**

* **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:

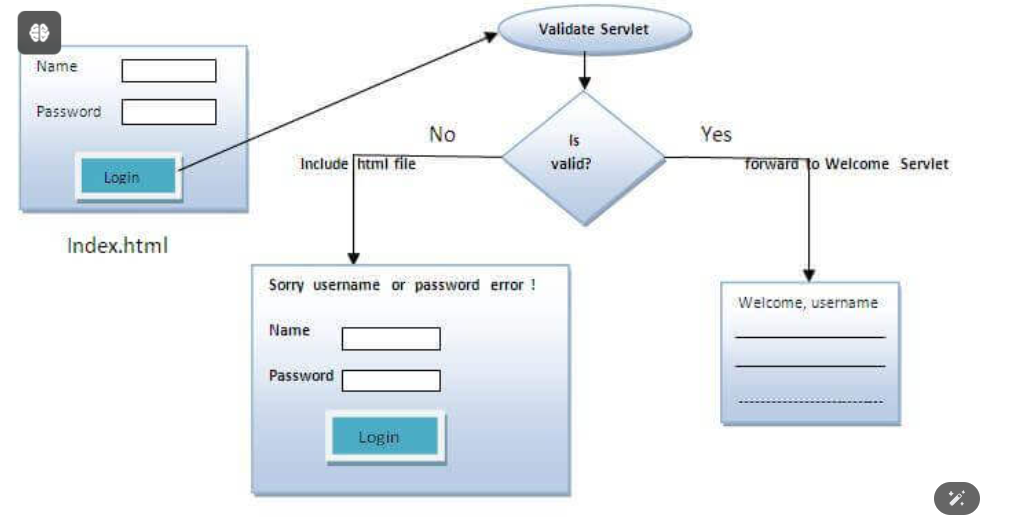
* HTTP Method: GET, POST, etc.
* Request URL: The URL used to make the request.
* Headers: Information about the client, content type, etc.
* Parameters: Data sent with the request (query parameters or form data).

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

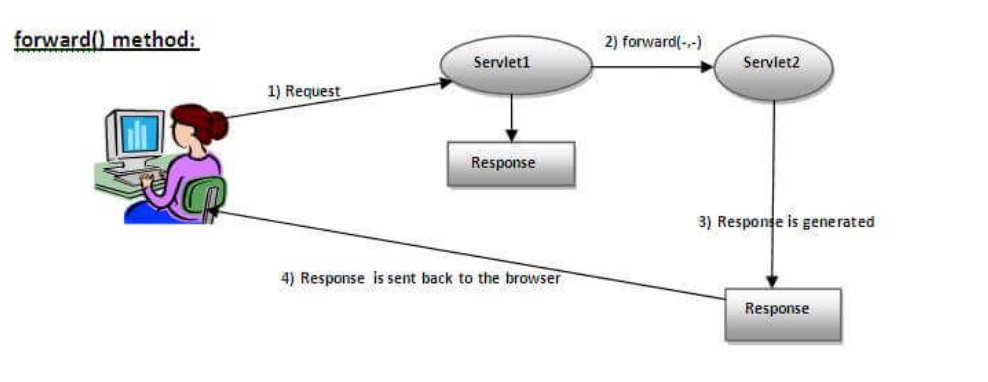
RequestDispatcher and sendRedirect() :

Servlet Collaboration:

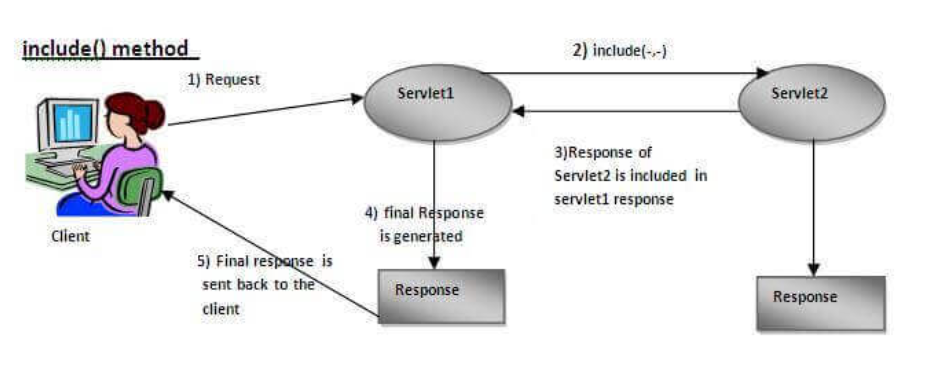
* RequestDispatcher: Used to forward requests and responses between servlets or JSPs within the same server. It doesn’t change the URL in the browser.

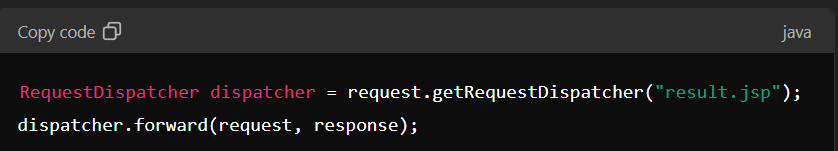


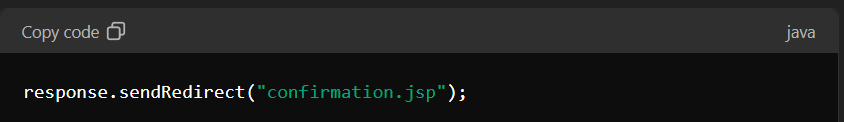
* forward(request, response): forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.



* 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:**

* **RequestDispatcher:** Forwarding happens server-side. The URL remains unchanged, and data can be shared using request attributes.
* **sendRedirect():** The browser is redirected to a new URL. The original request is discarded, and data cannot be shared using request attributes.

Difference Between ServletConfig and ServletContext

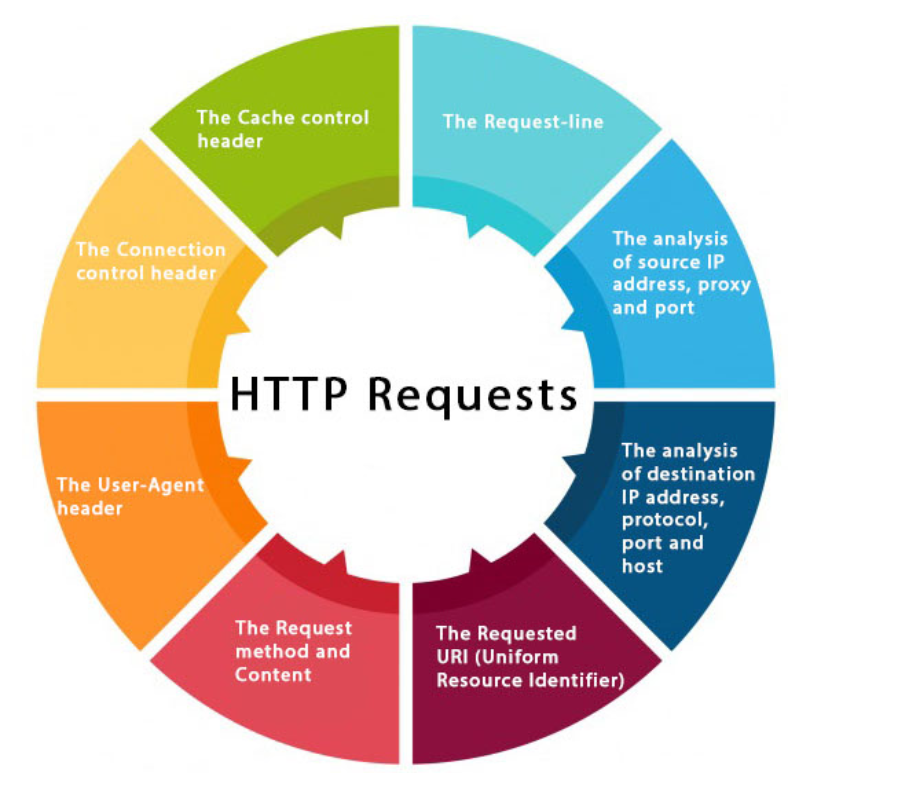
* **ServletConfig:** Provides servlet-specific initialization parameters(init). Each servlet has its own ServletConfig🡪 DatabaseURL.
* **ServletContext:** Provides application-wide information and parameters(init for all servlets). It is shared across all servlets in the web application(unique for complete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

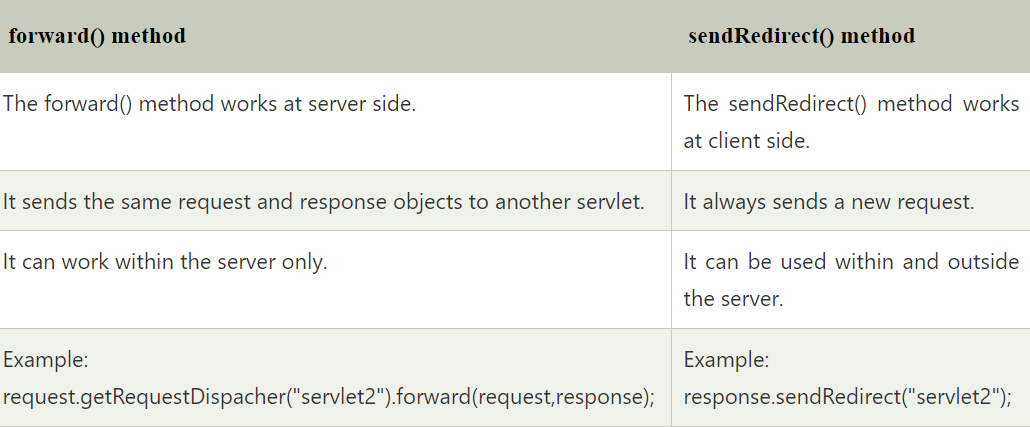
* provides common behavior to all the servlets.Servlet interface defines methods that all servlets must implement.
* Provides 3 life cycle methods (to init() , to service() the requests, to destroy() the servlet ), and 2 non-life cycle methods.
* init(): invoked by the web container only once and initializes 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.

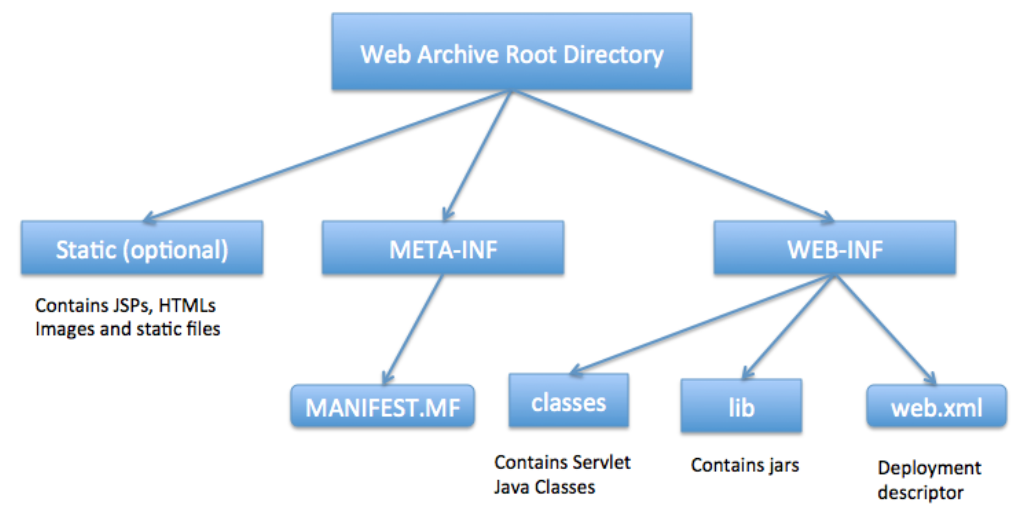
sendRedirect(): redirect response to another resource, it may be servlet, jsp or html file.

Forward() & sendRedirect()



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.