**Chapter 2: SERVLET PROGRAMMING**

**Topic – 1: Introduction**

**Introduction To Web**

* Uses use **web browsers** to interact with **websites**.
* Sometimes servers return response in form of **HTML pages**.
* User makes request using **URL** & server returns requested webpages as **response**.

**Web Application**

* **Website:** Collection of **static pages** & resources like **image**, **video** etc.
* **Web application:** Website with **dynamic** functionality.

**Topic – 2: Hypertext Transfer Protocol**

**Introduction To HTTP**

* **HTTP** is used during communication between **client** & **server**.
* It is a **stateless** protocol.
* **Stateless protocol:** Supporting just **one request** per connection.
* **SMTP** and **FTP** are **stateful** protocol.
* **HTTP** works on **datagram sockets**.

**HTTP Methods**

* Most common methods used are **Get** and **Post**.

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| **GET** | **URL request to server.** |
| **POST** | **Developer request to server for storing HTML file.** |
| **PUT** | **Same as POST but can also be used to update webpage.** |
| **DELETE** | **Request to delete resources from server.** |
| **OPTION** | **Request to show options in communication methods.** |
| **HEAD** | **Similar to GET but only webpage header is returned.** |

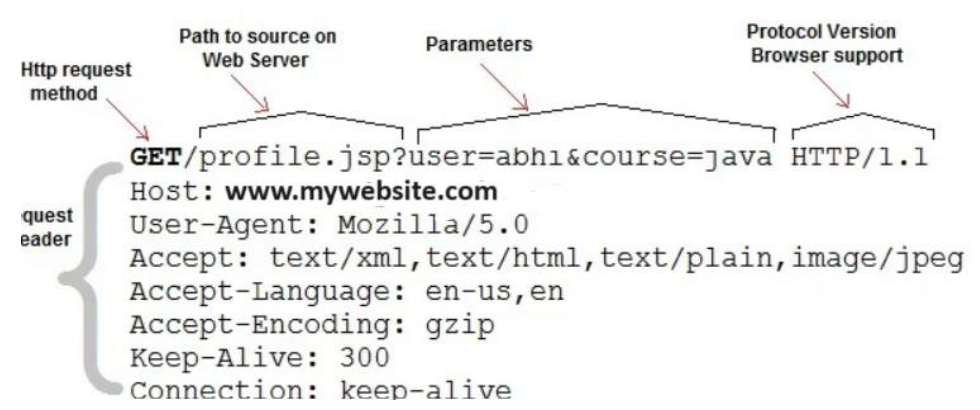
**GET v/s POST**

|  |  |
| --- | --- |
| **GET Method** | **POST Method** |
| **Data while being sent, is fit in the header.** | **Data while being sent, is fit in the request body.** |
| **Can send limited data.** | **Can send unlimited data.** |
| **Data is exposed, so not secured.** | **Data isn’t exposed, so secured.** |
| **Can be bookmarked.** | **Can’t be bookmarked.** |

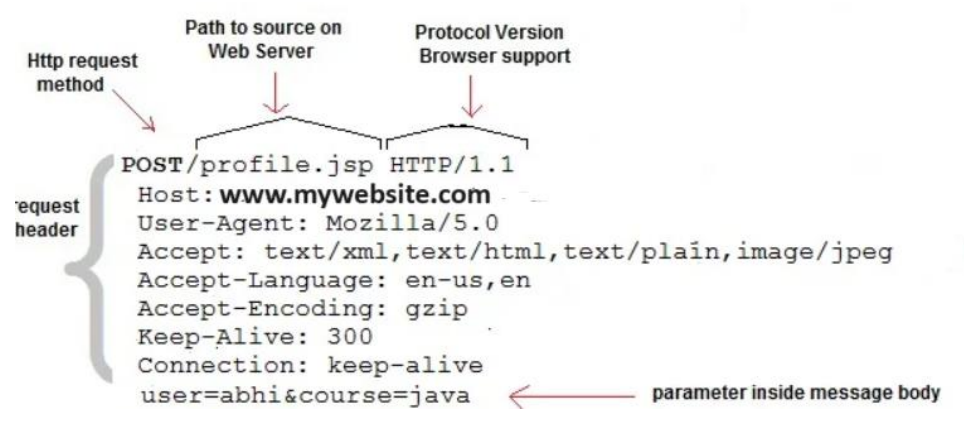
**POST v/s PUT**

|  |  |
| --- | --- |
| **POST Method** | **PUT Method** |
| **POST applied on an URL creates child resource.** | **PUT applied on an URL replaces the existing resource.** |
| **So, POST to /books will come under it as /books/1.** | **So, PUT to /books will replace it as /1.** |

**Anatomy Of HTTP GET Request**



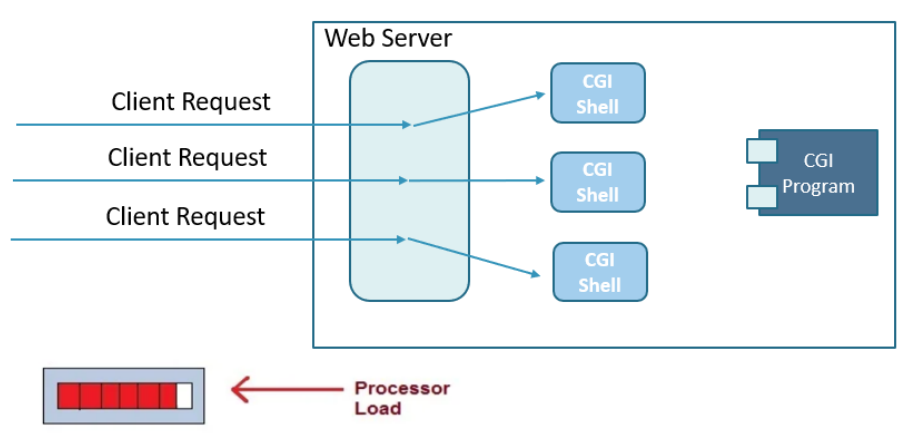
**Anatomy Of HTTP POST Request**



**Topic – 3: CGI (Common Gateway Interface)**

**Introduction**

* When Servlet **didn’t** exist, **CGI** was used.
* It embedded **links** that were links to **dynamic** pages, **not** static.
* The way to execute **CGI program** for each OS was **different**.
* So, the browser ran it in different **OS shells** to know the compatibility.
* Then the request is sent accordingly to the **web server**.
* Then the **HTTP file** is set as per the compatibility found in request.



**Advantages**

* Can be written in many programming languages.
* If program has bugs, it **won’t** crash.
* **No** concurrency issues in the database.

**Disadvantages**

* Takes a **lot of time** to response.
* **Doesn’t scale** smoothly.
* **Lacks OOP**, making use of encapsulation not possible.
* **CGI code** & **embedded HTML** code **aren’t** good combination in terms of logic.

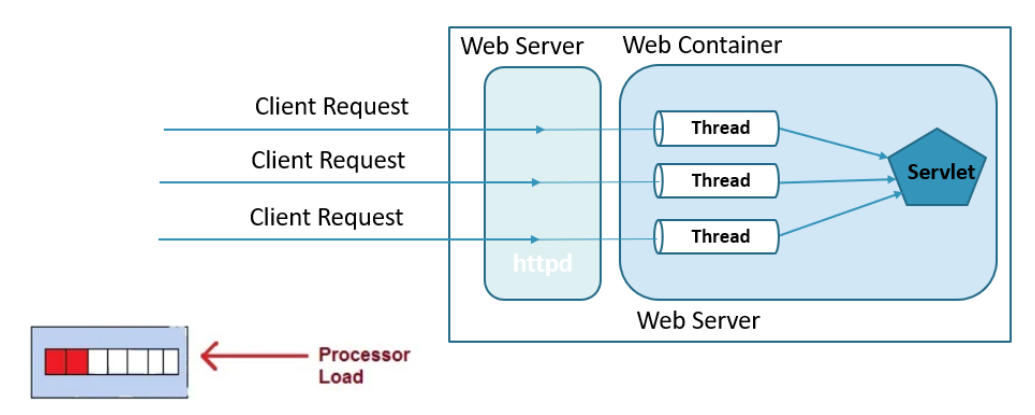
**Topic – 4: Servlet**

**Introduction**

* **Servlet** is a web technology by **Sun Microsystems** itself.
* It is defined by user with a **Java class**.
* Managed by ***Servlet container*** (**Servlet engine**).
* Implemented using ***javax.servlet.Servlet***.

**Servlet Properties**

* Webserver’s functionality enhancing **dynamic web resource**.
* Writes server-side program.



**Advantages**

* **Highly portable** as they are written in Java.
* **More features** than CGI.
* **Efficient** for using just one object instance.
* **Very safe** as Java has automatic garbage collector.
* Can be **easily integrated** with the server components.
* **API is extensible** in terms of network protocols features.

**Servlet Container**

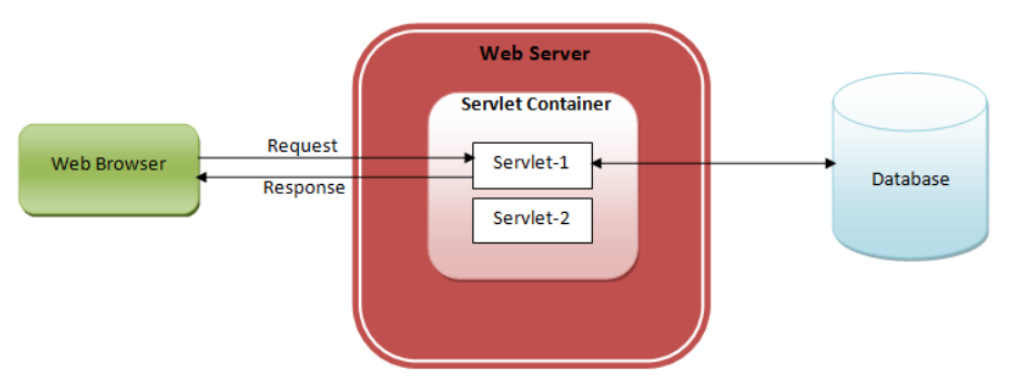
* Also known as **Servlet Engine**.
* An executable program meant to **load**, **initialize** & **execute** servlets.
* Container handles the **multiple thread requests** by creating & destroying one thread at a time.

**Application Of Servlet**

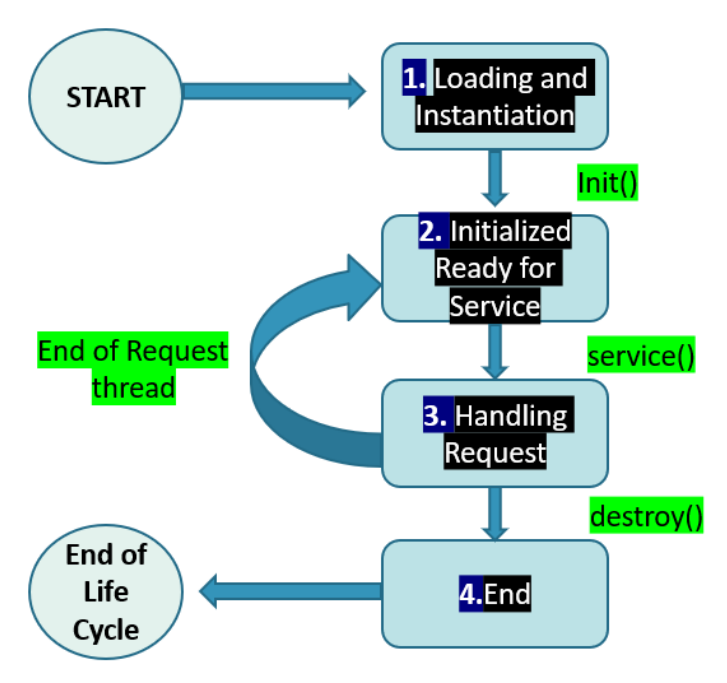
* Reads data sent by users through systems like **HTML forms**.
* These data can be read using **Java variables** & responded accordingly.
* Data can be sent to users in form of **HTML**, **XML**, **Excel** etc.
* Can return various information to browser & clients.

**Servlet Architecture**

* ***javax.servlet.Servlet*** isn’t used directly, but extended with ***javax.servlet.GenericServlet*** or ***javax.servlet.http.HttpServlet***.



**Servlet Life Cycle**



**init() Method**

* User has to **manually call** it just **once** for the **first thread**.
* Or it can also be called **automatically** when the Servlet servers **fire up**.

***public void init() throws ServletException {***

***// init() initialization***

***}***

**service() Method**

* Takes requests like **GET**, **PUT**, **UPDATE**, **DELETE** etc.
* Generates a **new thread** when the request comes.
* Implements operations through methods like ***doGet()***, ***doPut()***, ***doUpdate()***, ***doDelete()*** etc.

**destroy() Method**

* When called, checks for any data to **recover** from website & **halts** all threads.

***public void destroy() {***

***// destroy() finalizing***

***}***

**Topic – 5: Deployment Descriptor File (DDF)**

**Introduction**

* **DDF** links **http** requests with **servlets**.
* Its file must always be named as **web.xml**.
* This file resides in **WEB-INF/WAR** directory.
* ***<servlet>*** is used to declare a **servlet element** & has to be declared **separately** for each Servlet.
* Then <***servlet-mapping>*** is used to define a **URL to Servlet mapping**.

**Syntax**

***<servlet>***

***<servlet-name> Practice1 </servlet-name>***

***<servlet-class>***

***com.parul.controller.servlet***

***</servlet-class>***

***</servlet>***

***<servlet-mapping>***

***<servlet-name> Practice1 </servlet-name>***

***<url-pattern> / </url-pattern>***

***<servlet-mapping>***

**Topic – 6: Servlet API**

**Servlet Interface Methods**

|  |  |
| --- | --- |
| **Interfaces** | **Classes** |
| **public void init(ServletConfiguration)** | **-** |
| **public void service(ServletRequestreq, ServletResponse res)** | **Is invoked multiple times in a servlet’s life cycle, unlike init() & destroy().** |
| **public void destroy()** | **-** |
| **public String ConfiggetServletConfig()** | **Gets object of ServletConfig.** |
| **public String getServletInfo()** | **Gets copyright info about Servlet.** |

**HTTP Servlets**

* Methods in **HttpServlet** are also **abstract classes**.
* All are ***protected void*** except a few.

|  |  |
| --- | --- |
| **Method** | **Description** |
| **service(HttpServletRequest req, ServletResponse res)** | **Same service method.** |
| **doGet(HttpServletRequest req, ServletResponse res)** | **Handles GET request.** |
| **doPost(HttpServletRequest req, ServletResponse res)** | **-** |
| **doHead(HttpServletRequest req, ServletResponse res)** | **-** |
| **doOptions(HttpServletRequest req, ServletResponse res)** | **-** |
| **doPut(HttpServletRequest req, ServletResponse res)** | **-** |
| **doTrace(HttpServletRequest req, ServletResponse res)** | **-** |
| **doDelete(HttpServletRequest req, ServletResponse res)** | **-** |
| **getLastModified(HttpServletRequest req)** | **Gets time for last modified HttpServletRequest.** |

**Servlet Config**

* ServletConfig is an object.