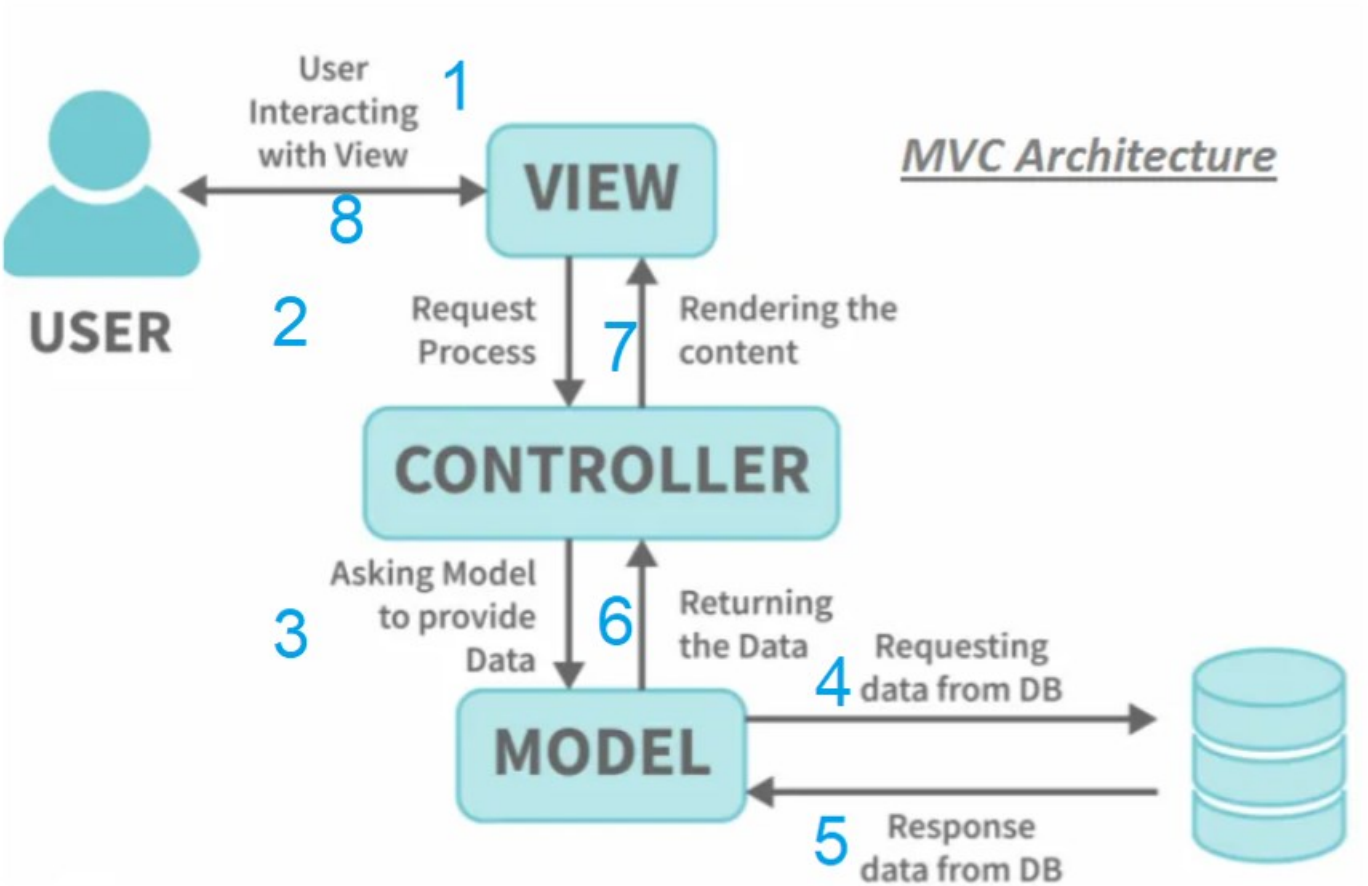
Java Servlets - Introduction

Presented by

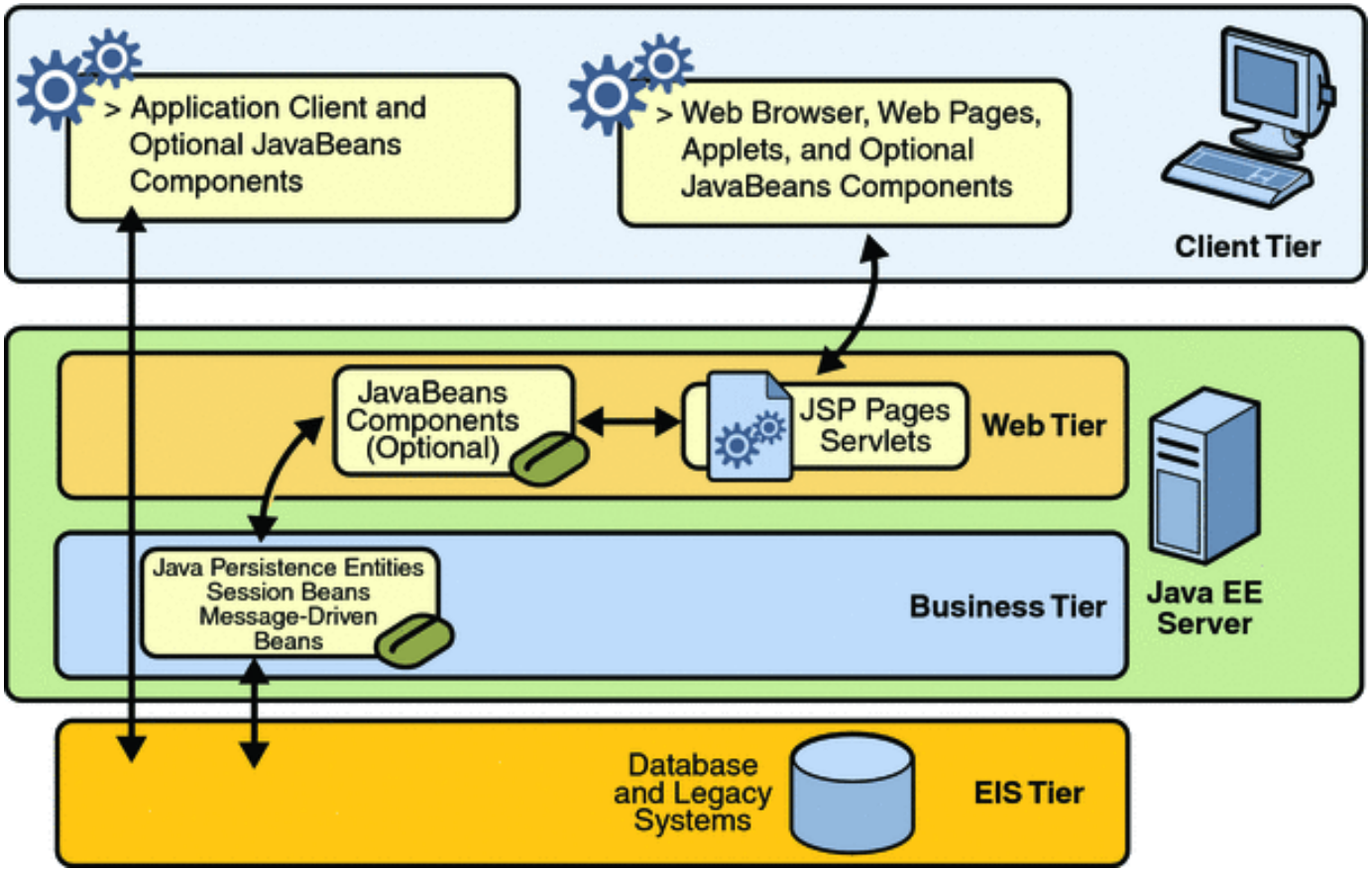
Sagar

Java Consultant

MVC Design Pattern

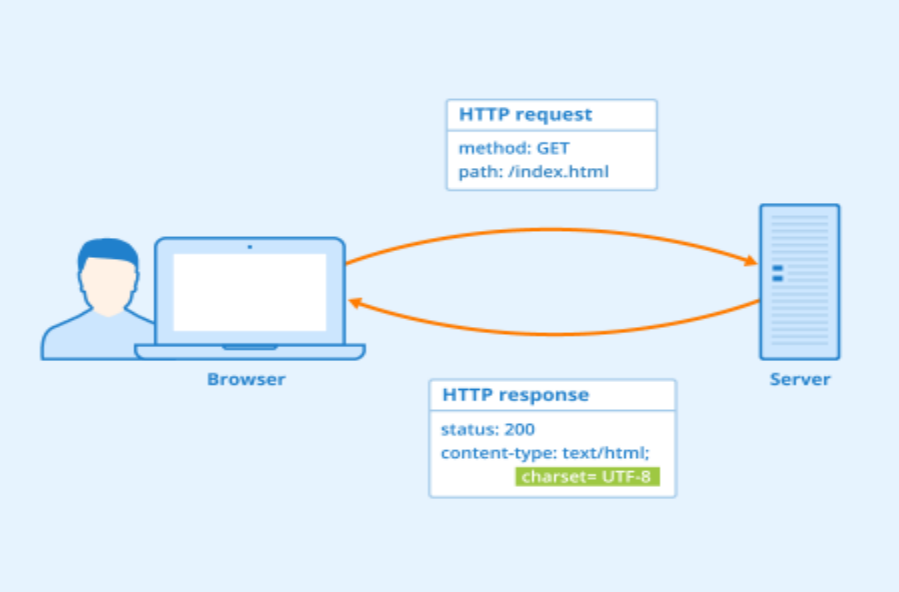


JEE Architecture



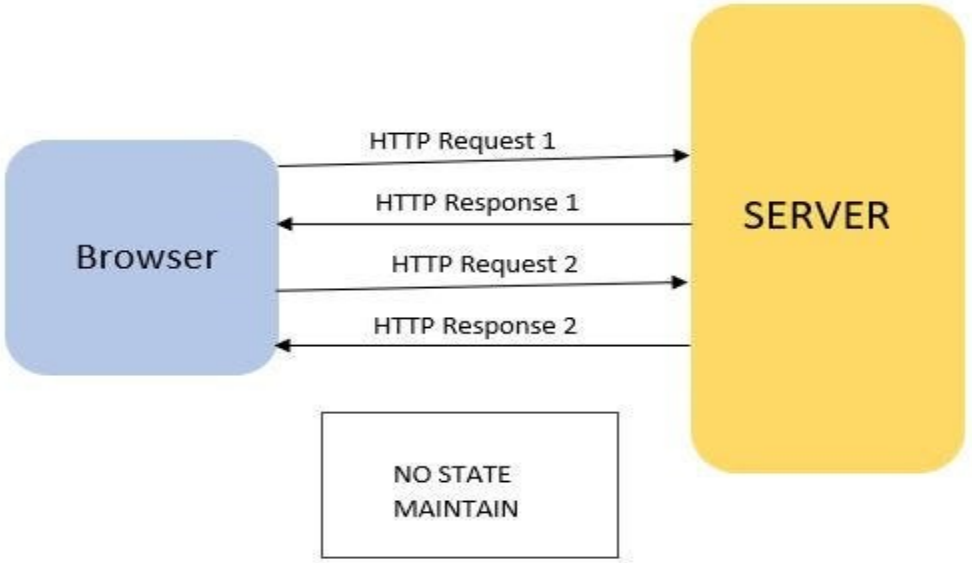
HTTP Headers

- The HTTP header is part of the Hypertext Transfer Protocol (HTTP) .
- It transmits additional information during HTTP requests or responses.
- In addition to the data that is delivered to a browser by the web server of the called website, server and browser exchange meta information about the document via the HTTP header.

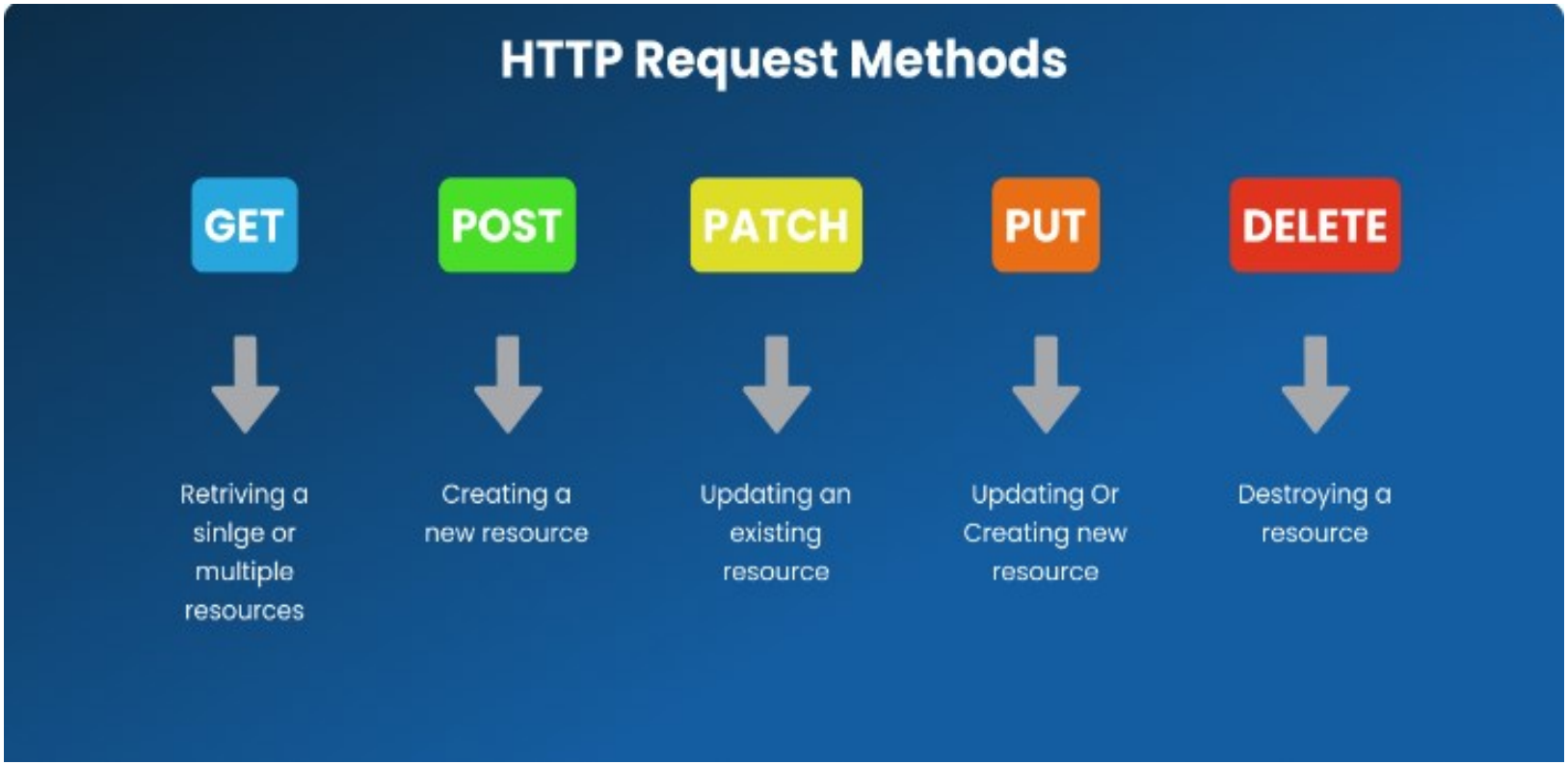


HTTP Protocol

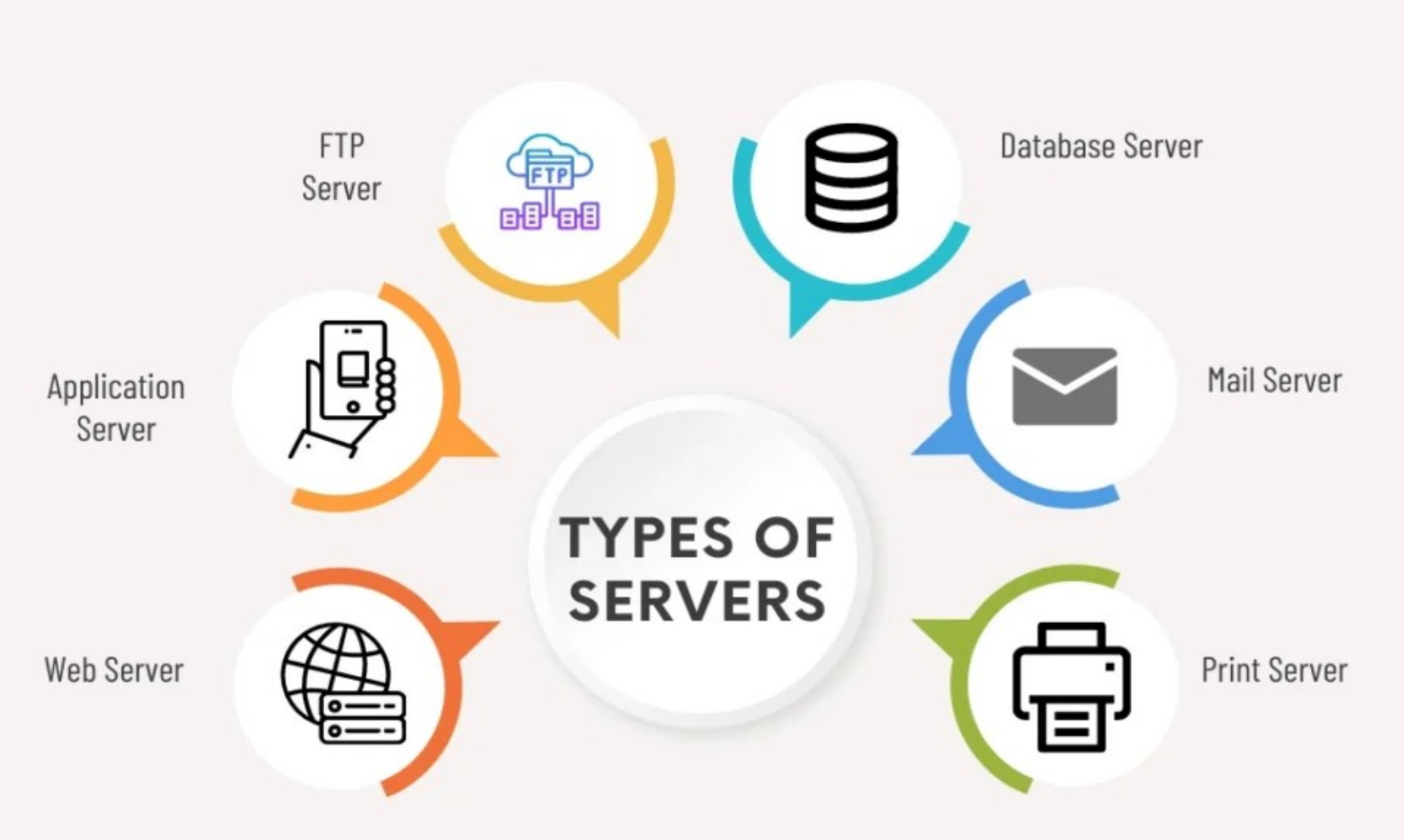
- Stateless Protocols is a network protocols in which Client send request to the server and server response back to current state of user
- HTTP is stateless protocol.
- Stateless Protocol does not remember previous request or response.



HTTP Methods



Types of Servers



Web Server and application Server

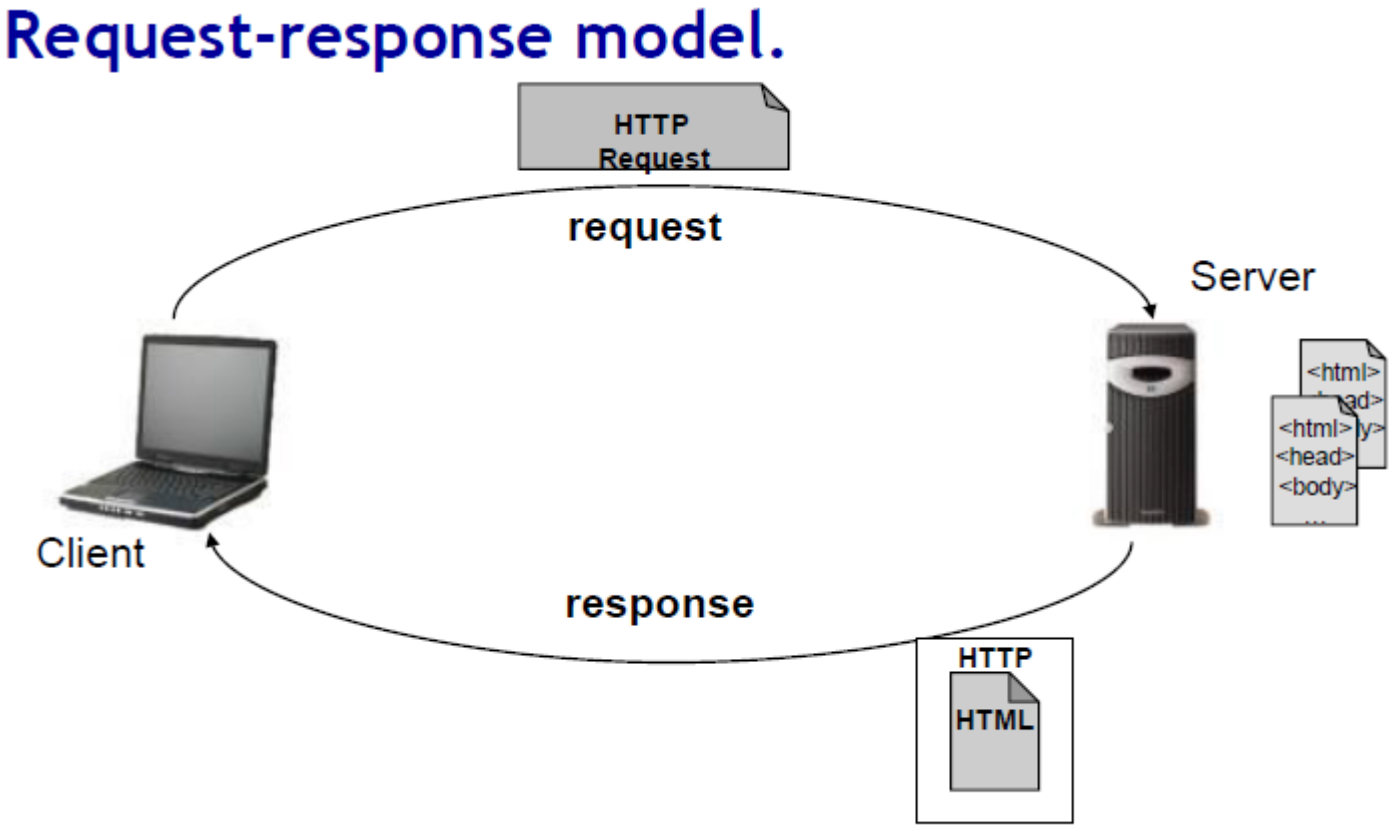
Web server

- Information for the internet is stored on web servers.
- Information is retrieved using “HTTP” headers by requesting the files stored on online servers before being transferred to your web browser.

Application server

- Application servers are the best choice for corporations because they can efficiently host big amounts of application data to numerous users at once.
- Through virtual server connections, these servers link clients to software programs.
- This enables users to access applications without downloading data to their own hardware.

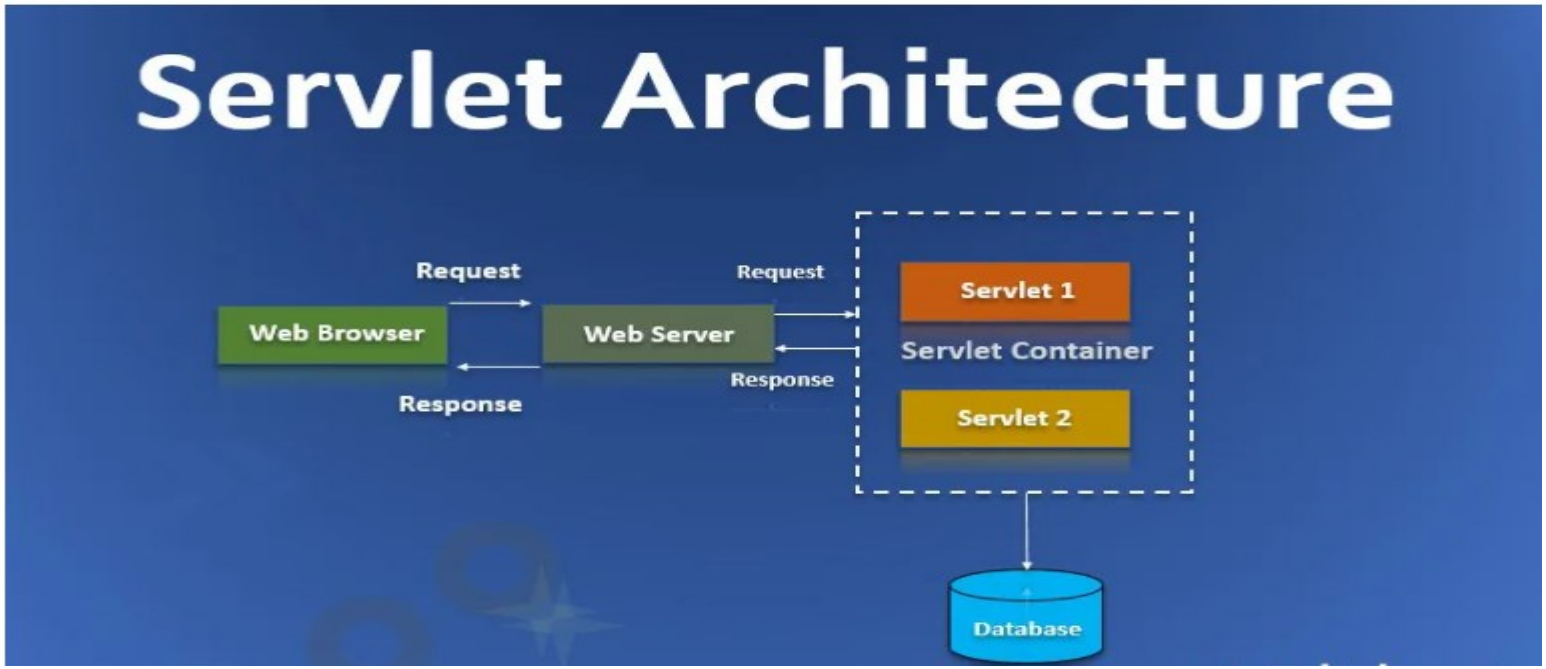
Request and Response



Servlet – What ?

Servlet is a server side dynamic component.

Servlet architecture comes under a java programming language to create **dynamic web applications**.

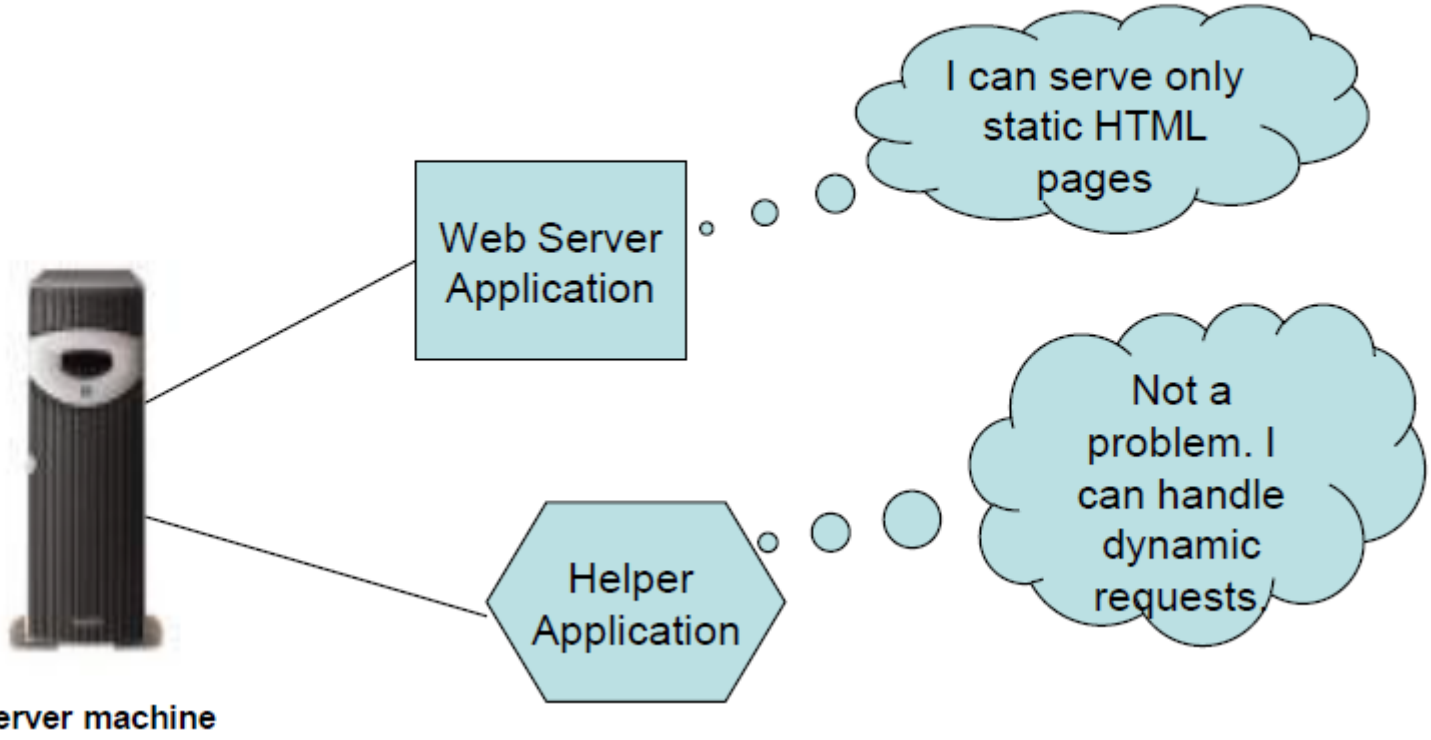


Servlet – What ?

- Servlet performs the below services:
    - •Control the flow of the application.
    - •Generate dynamic web content.
    - •Server-side load balancing.
    - •Implement business logic.
- Servlets are used to develop server-side applications

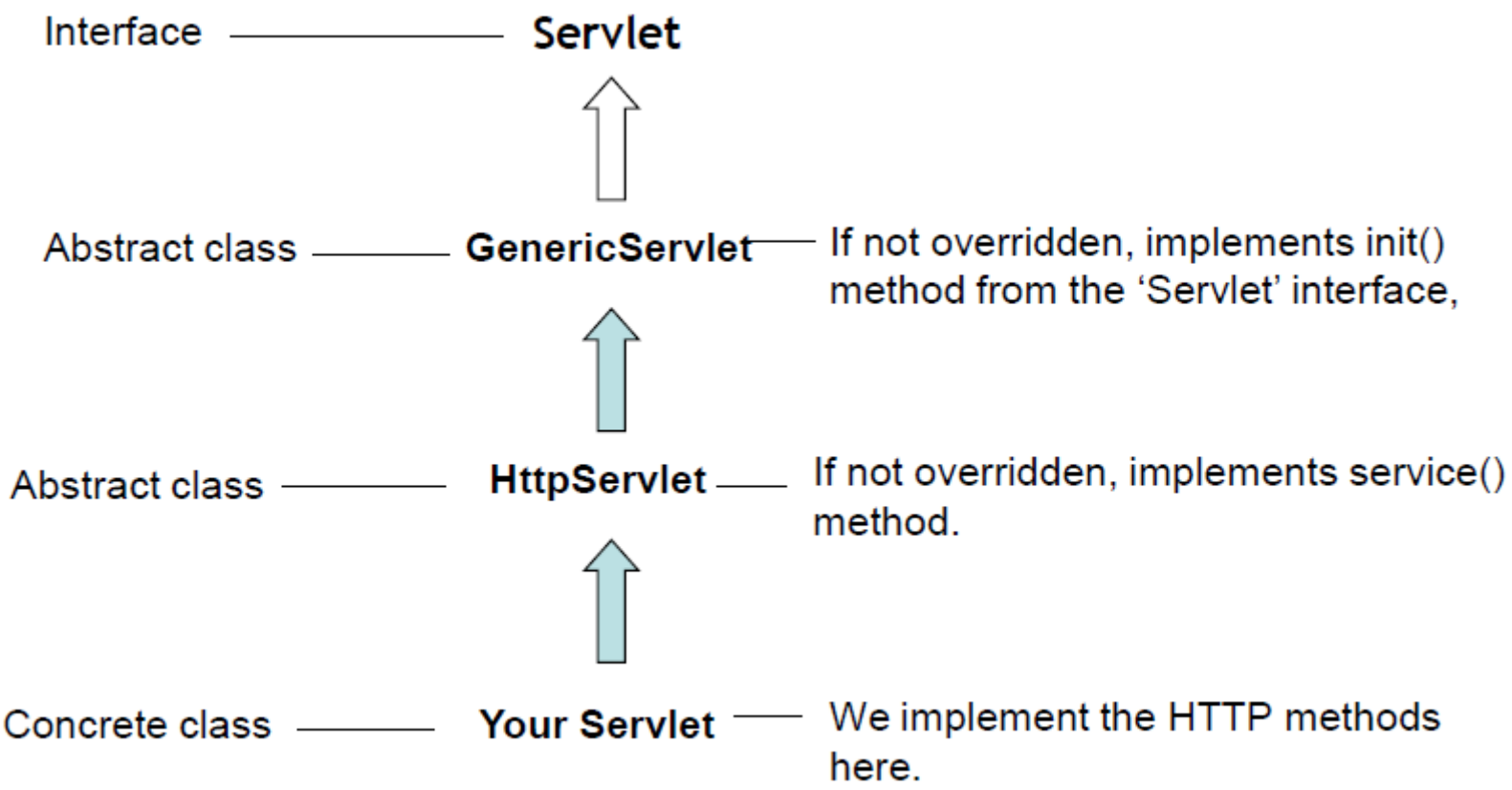
Servlet – What ?

Where does Servlet come into the picture?



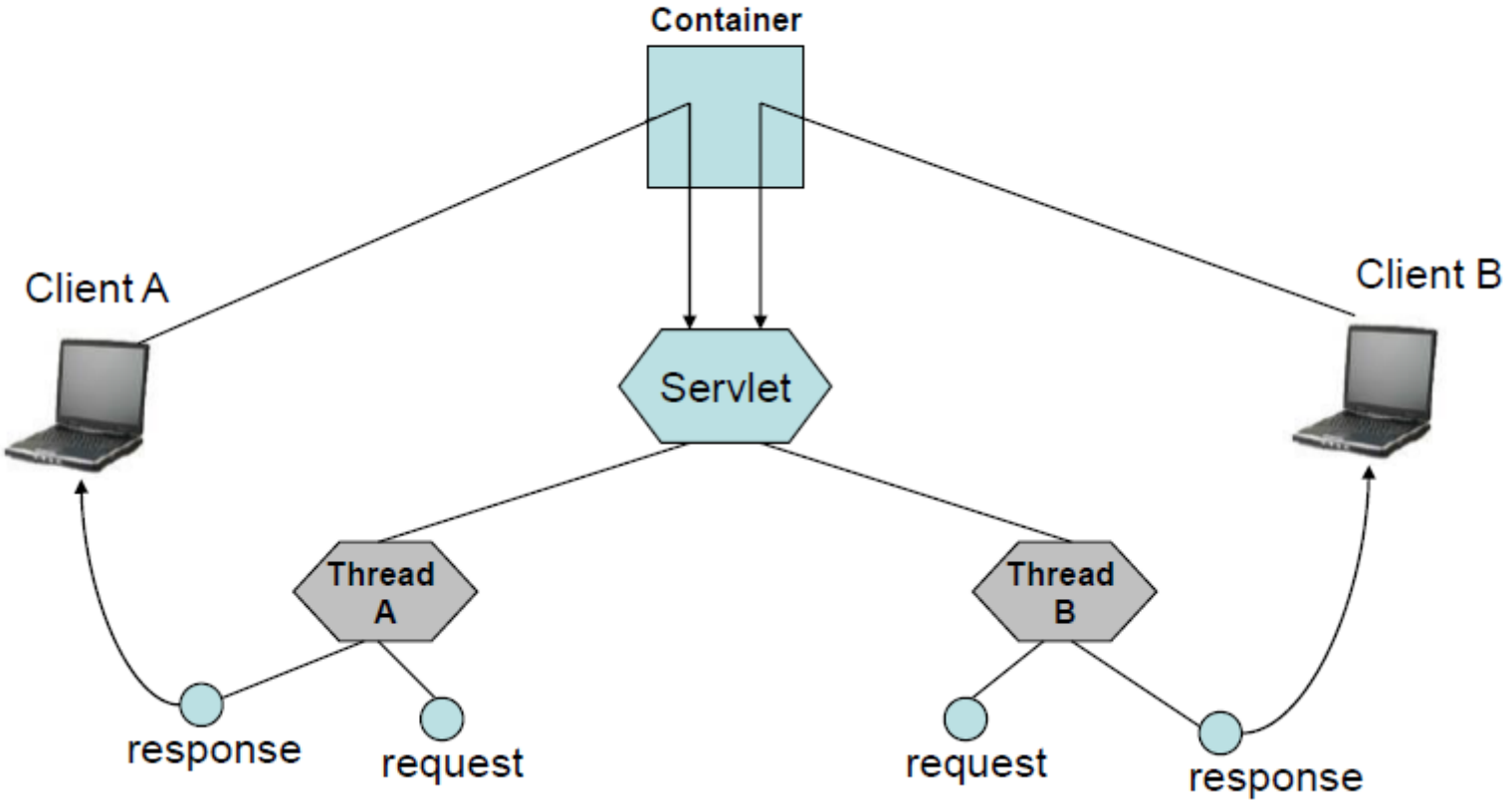
"The Helper Application is a **SERVLET**"

Servlet Hierarchy

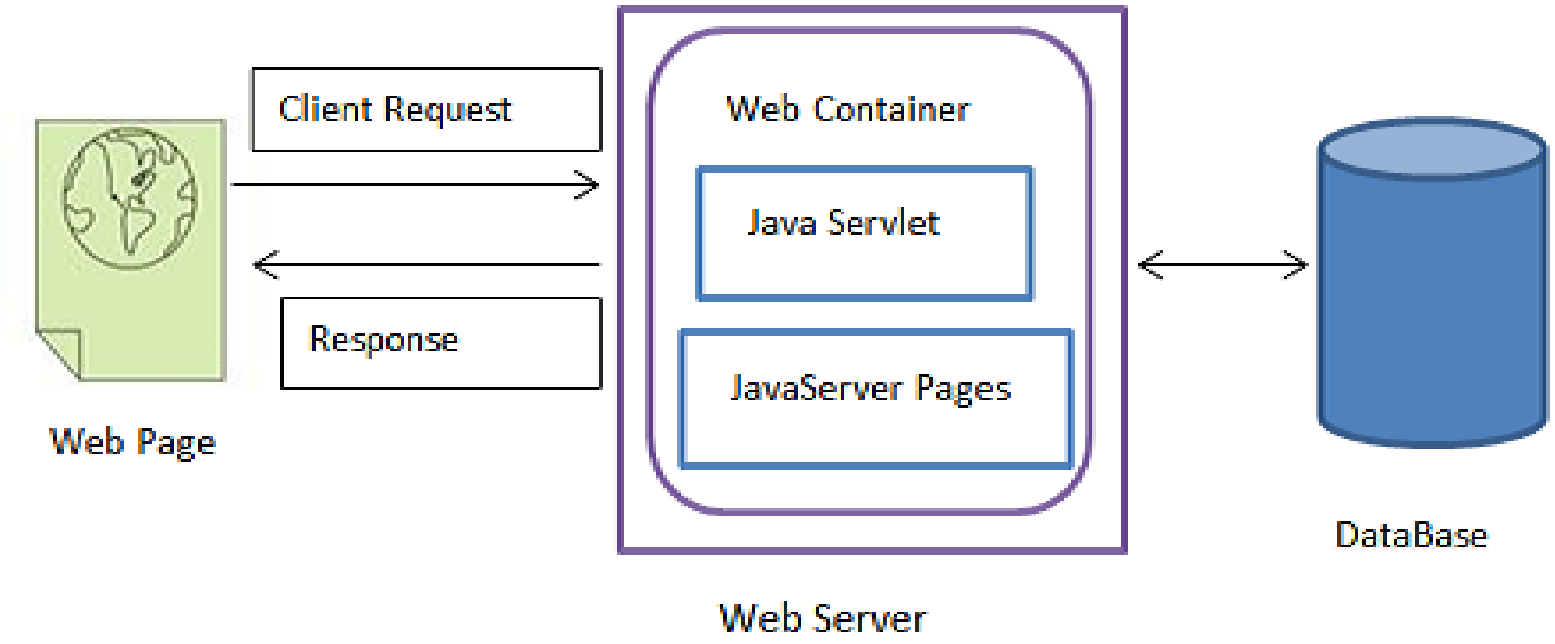


Servlet Request Thread Model

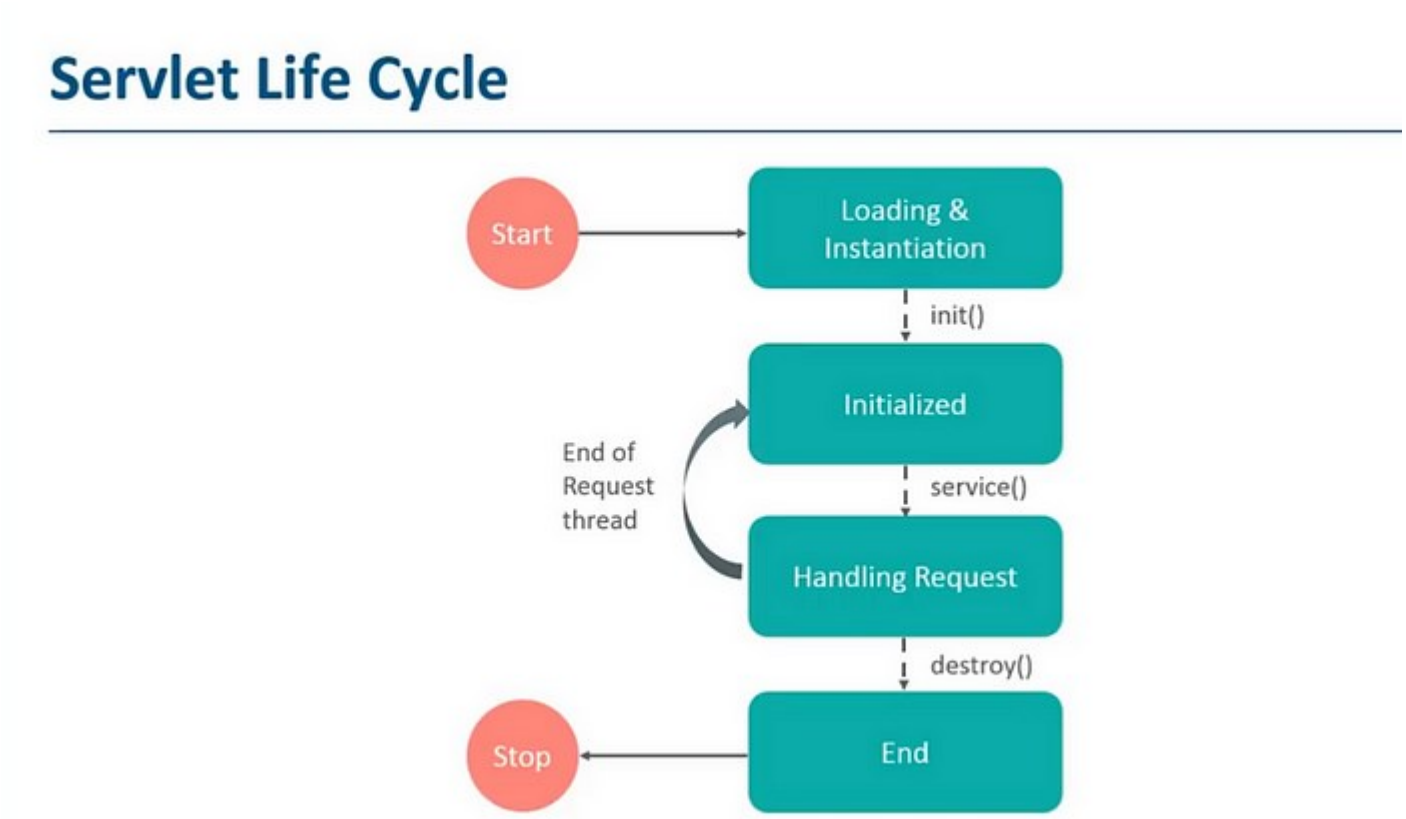
The Container runs multiple threads to process multiple requests to a single servlet.



Web Container



Servlet Life Cycle Methods



Deployment Descriptor

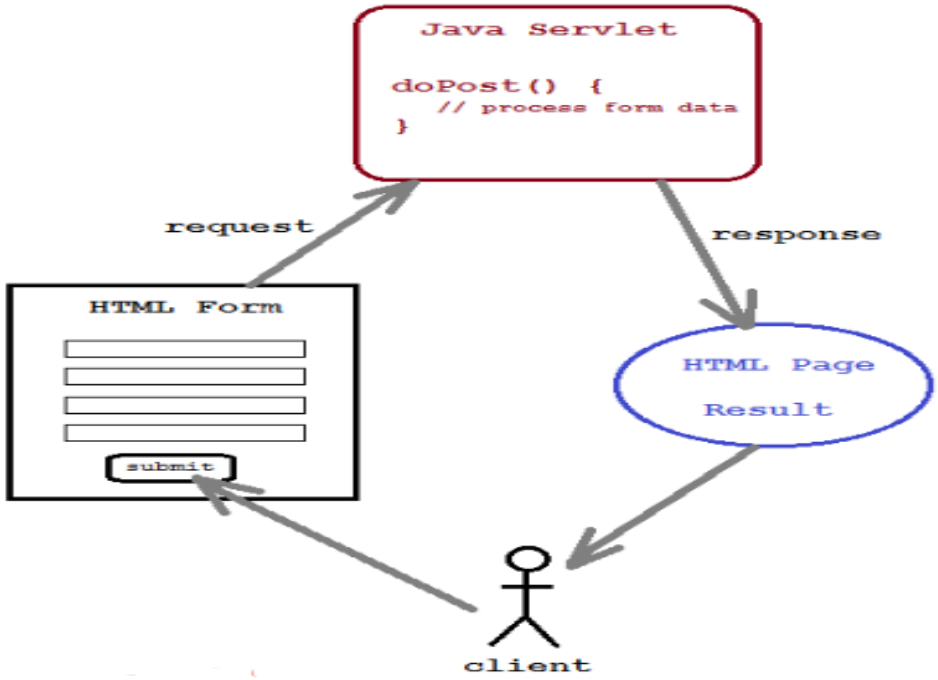
How does the Container know which Servlet the client has requested for?

web.xml

```
<web-app>
  .........
  <servlet>
    <servlet-name>LoginServlet</servlet-name>
     <servlet-class>LoginServlet</servlet-class>
   </servlet>

  <servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/LoginServlet</url-pattern>
  </servlet-mapping>
  ..........
  ..........
</web-app>
```

Servlet Handling Request



Servlet Form Parameters

Login Form: login.html

```
<form action="LoginServlet">
      <input type="text" name="name">
            <br><br>
      <input type="submit" value="Submit">
</form>
```

In the Servlet

```
@Override public void service(HttpServletRequest request, HttpServletResponse
response) throws IOException {
            String name = request.getParameter("name");
            response.getWriter().println("<h1>Hello " + name + "!</h1>");
}
```

**Demo

Inter Servlet Communication – sendRedirect()

- Navigating from one servlet to another / another response
- Used through ***sendRedirect() method and RequestDispatcher object***
- **sendRedirect()** example :

Inter Servlet Communication-RequestDispatcher

Request Dispatcher allows a servlet to communicate with another servlet / response page

It has ***forward()*** and ***include()*** methods

a. **forward()** – forwards to a response
can not return to the request point

b. **include()** – includes / inserts a response
continues after including the response

Ex:

```
RequestDispatcher dis = request.getRequestDispatcher("ServletTwo");
dis.forward(request, response);

RequestDispatcher dis = request.getRequestDispatcher("login.html");
dis.include(request, response);
```

**Demos

Servlet Config and Context Parameters

ServletConfig & ServletContext are basically the configuration objects which are used by the servlet container to initialize the various parameter of a web application.

ServletConfig is an interface in the Servlet API and is used to initialize a single servlet in a web application by the servlet container.

Inside deployment descriptor(web.xml), servlet initialization parameter are defined related to that servlet.

Servlet Config an InitParameters

The particulars are declared in the **<init-param />** tag in a name-value pair

Ex:

```
<servlet>
    <description></description>
    <display-name>UserDetailsServlet</display-name>
    <servlet-name>UserDetailsServlet</servlet-name>
    <servlet-class>UserDetailsServlet</servlet-class>
    <init-param>
        <param-name>username</param-name>
        <param-value>Krishna</param-value>

    </init-param>
    <init-param>
        <param-name>city</param-name>
        <param-value>Hyderabad</param-value>
    </init-param>
    <init-param>
        <param-name>technology</param-name>
        <param-value>Java</param-value>
    </init-param>
</servlet>
```

**Demo

Servlet Context

ServletContext

ServletContext is a shared memory segment for Web applications.

When an object is placed in the ServletContext, it exists for the life of a Web application, unless it is explicitly removed or replaced.

There is one ServletContext per web application and all servlet share it.

It can be retrieved via getServletContext() method.

Servlet Context…

ServletContext has an "application" scope, and can also be used to pass information between servlets within the same application.

The method getInitParameter(“parametername”) will get the value of the parameter

**Demo

Thank You