Unit 2
Servlet and JSP

Web development using Servlets

Syllabus…

- **Servlets:** Introduction
- Web application Architecture
- Http Protocol & Http Methods
- Web Server & Web Container
- Servlet Interface
- GenericServlet
- HttpServlet
- Servlet Life Cycle
- ServletConfig
- ServletContext
- Servlet Communication
- Session Tracking Mechanisms

jsp

- **JSP:** Introduction
- JSP LifeCycle
- JSP Implicit Objects & Scopes
- JSP Directives
- JSP Scripting Elements
- JSP Actions: Standard actions and customized actions

Servlet

- **Servlet** technology is used to create web application (resides at server side and generates dynamic web page).

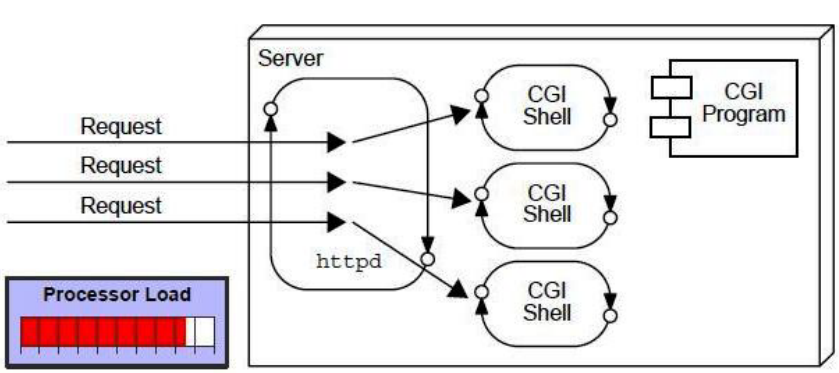
- **Servlet** technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was popular as a server-side programming language. But there was many disadvantages of this technology.

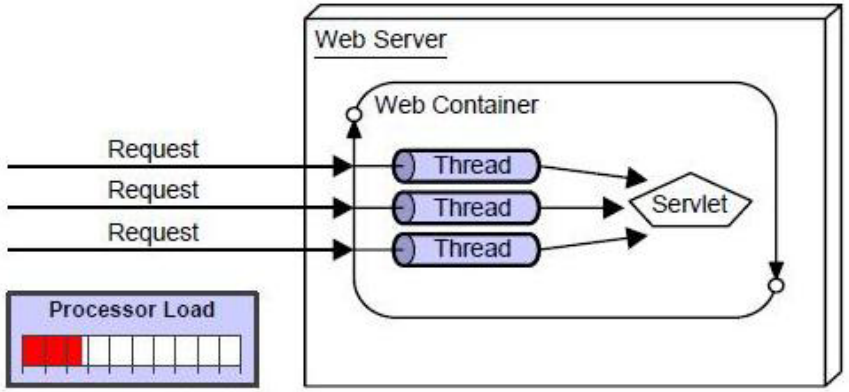
What is a Servlet?

- Servlet is an API that provides many interfaces and classes including documentations.
- Servlet is an interface that must be implemented for creating any servlet.
- Servlet is a class that extend the capabilities of the servers and respond to the incoming request. It can respond to any type of requests.
- Servlet is a web component that is deployed on the server to create dynamic web page.

Servlet vs CGI

- CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request. For each request, it starts a new process.
-





Disadvantages of CGI

- If number of clients increases, it takes more time for sending response.
- For each request, it starts a process and Web server is limited to start processes.
- It uses platform dependent language e.g. C, C++, perl.

Benefits of servlets

- **Better performance:** because it creates a thread for each request not process.
- **Portability:** because it uses java language.
- **Robust:** Servlets are managed by JVM so we don't need to worry about memory leak, garbage collection etc.
- **Secure:** because it uses java language..

Introduction - what is a request and response

HTTP Request

Key elements of a "**request**" stream:

- HTTP method (action to be performed).
- The page to access (a URL).
- Form parameters.

HTTP Response

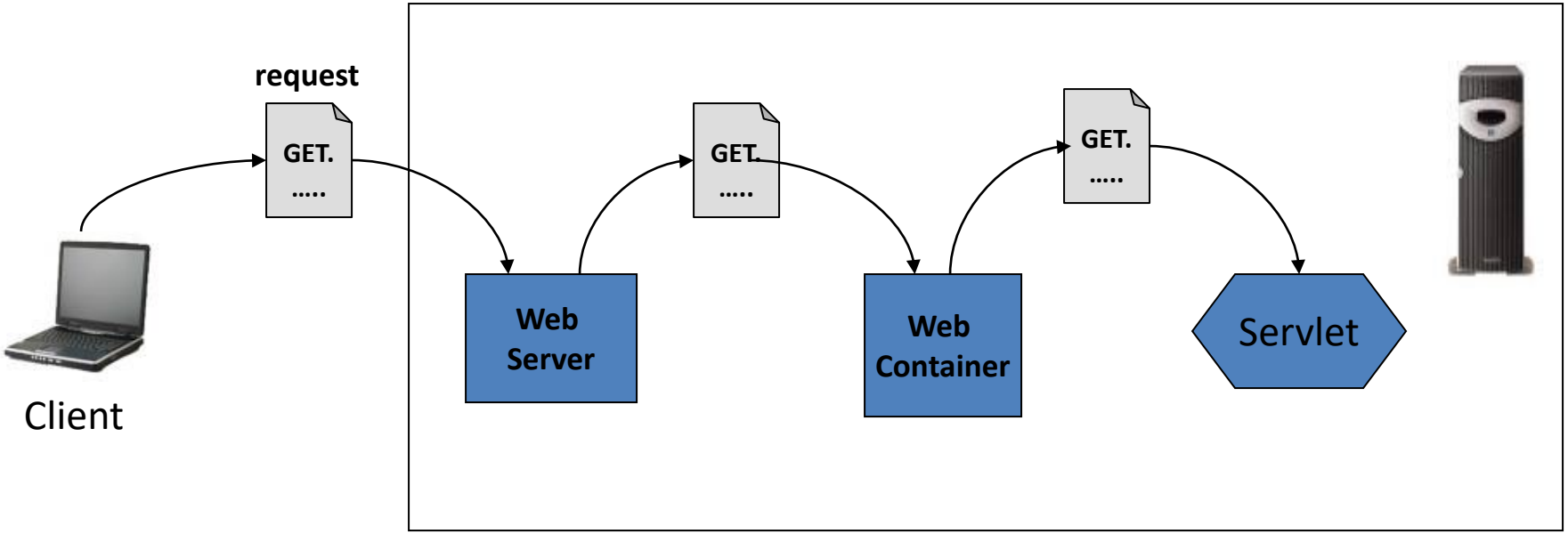
Key elements of a "**response**" stream:

- A status code (for whether the request was successful).
- Content-type (text, picture, html, etc…).
- The content (the actual content).

Servlet Architecture -Web Container

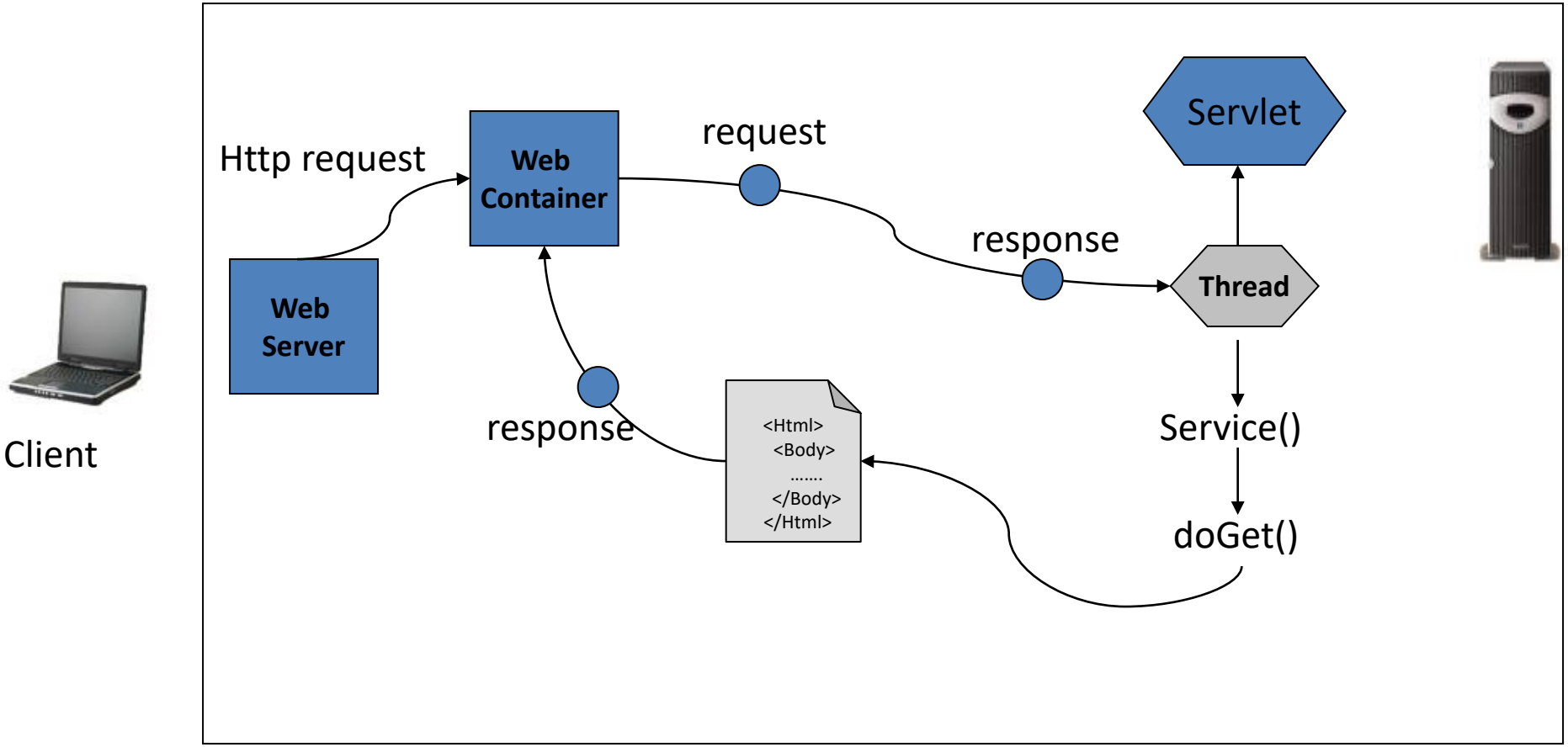
- **What is a Web Container?**

- Servlets don't have a main method.
  - They are under the control of another Java application called the "**Container**" **(e.g. Tomcat)**

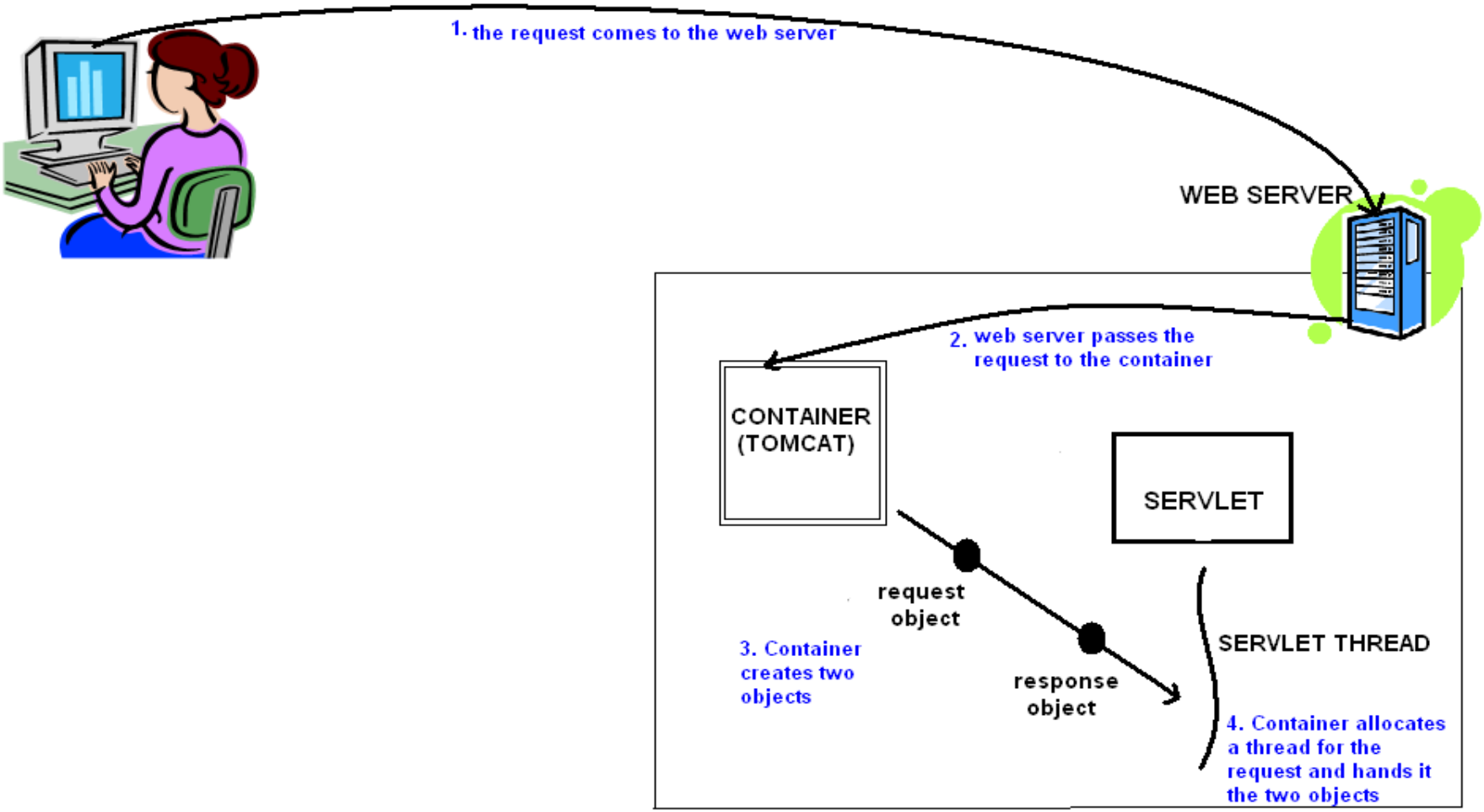


Servlet Architecture - Web Container

- **How does the Container handle a request?**



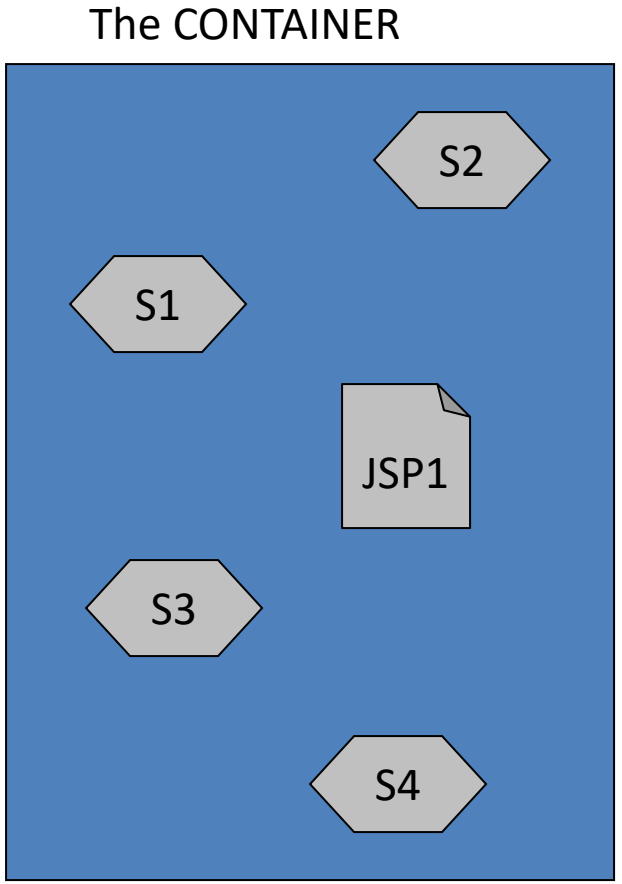
How it works??


1. the request comes to the web server
WEB SERVER
2. web server passes the request to the container
CONTAINER (TOMCAT)
SERVLET
request object
response object
3. Container creates two objects
SERVLET THREAD
4. Container allocates a thread for the request and hands it the two objects

Servlet Architecture - Web Container

What is the role of Web Container ?

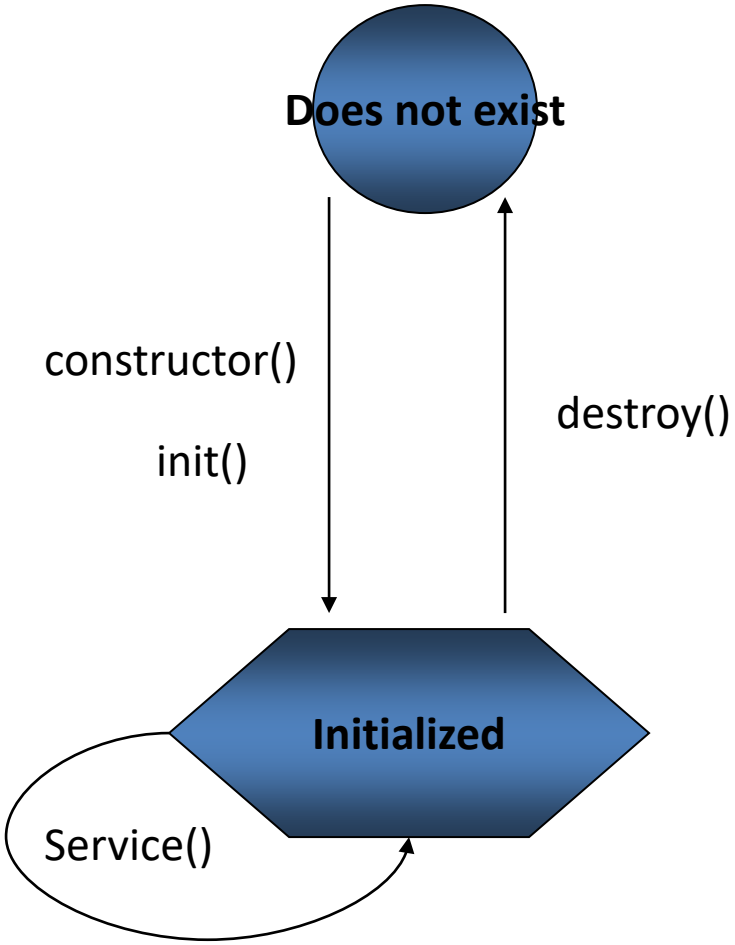
- Communication Support
- Lifecycle Management
- Multi-threading support
- Security
- JSP Support



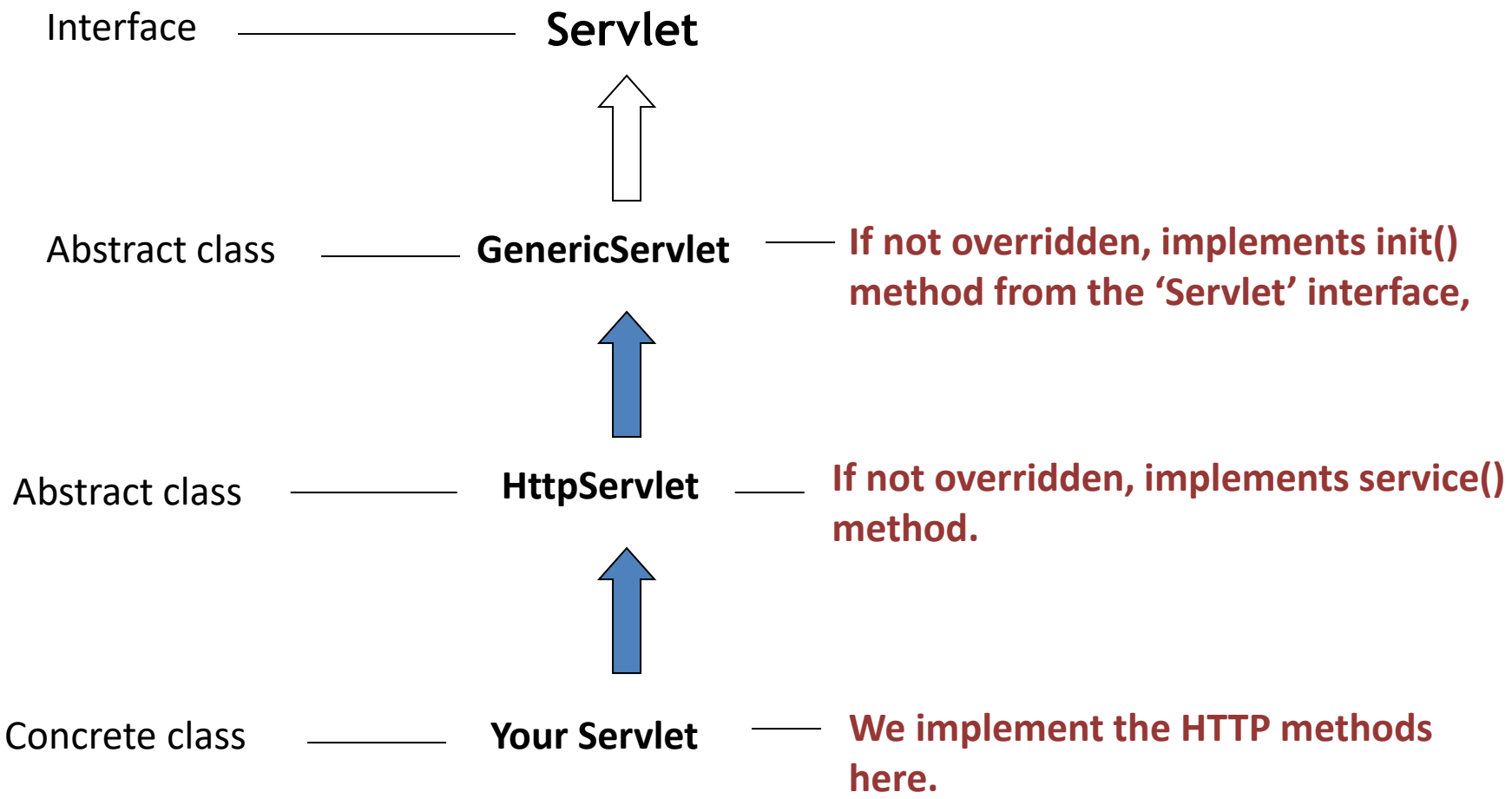
The container can contain multiple Servlets & JSPs within it

Servlet Lifecycle

- **The Servlet lifecycle is simple, there is only one main state – "Initialized".**



Servlet Lifecycle - Hierarchy

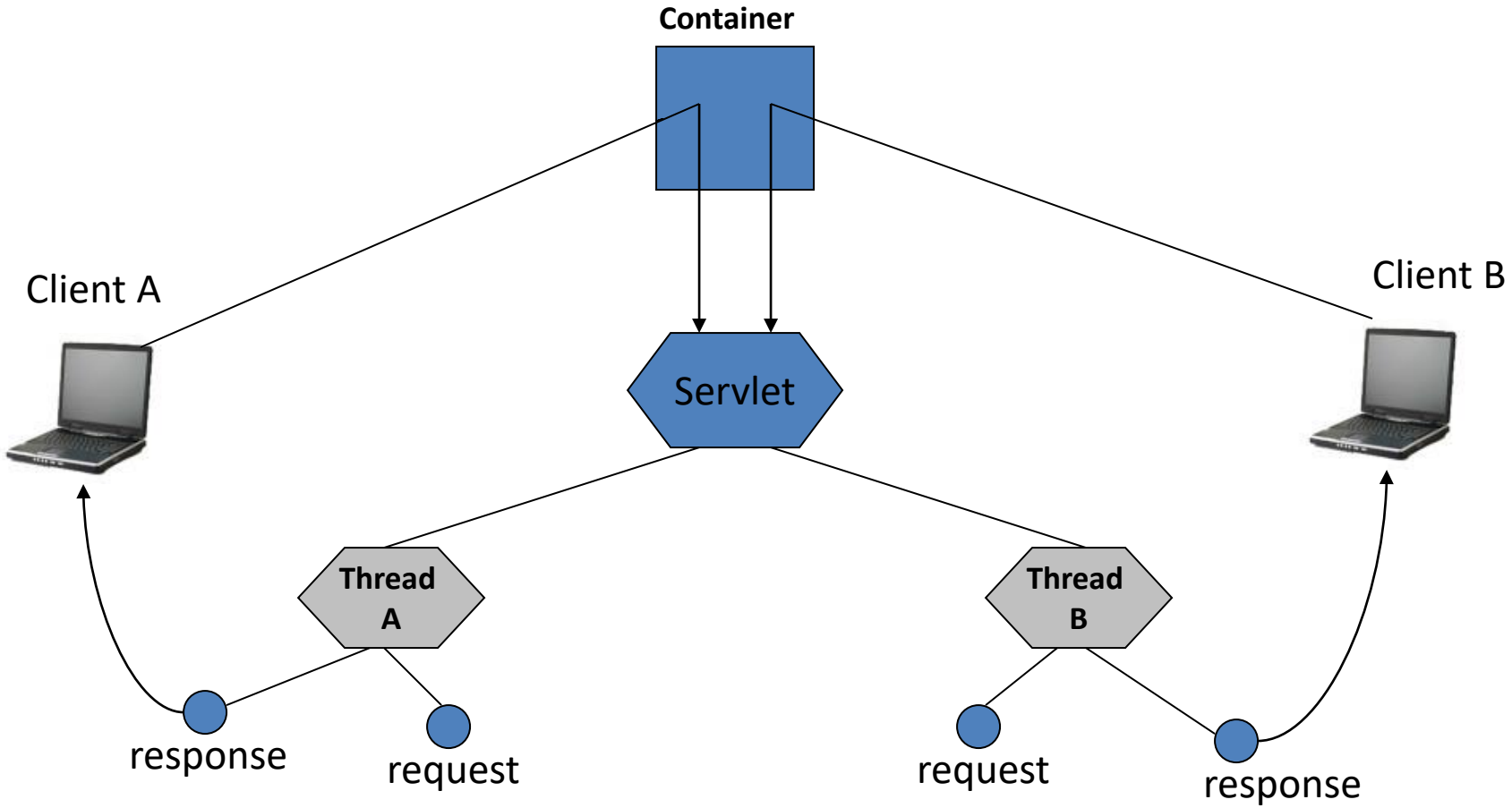


Servlet Lifecycle – 3 big moments

	When is it called	What it's for	Do you override it
init()	The container calls the init() before the servlet can service any client requests.	To initialize your servlet before handling any client requests.	Possibly
service()	When a new request for that servlet comes in.	To determine which HTTP method should be called.	No. Very unlikely
doGet() or doPost()	The service() method invokes it based on the HTTP method from the request.	To handle the business logic.	Always

Servlet Lifecycle - Thread handling

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

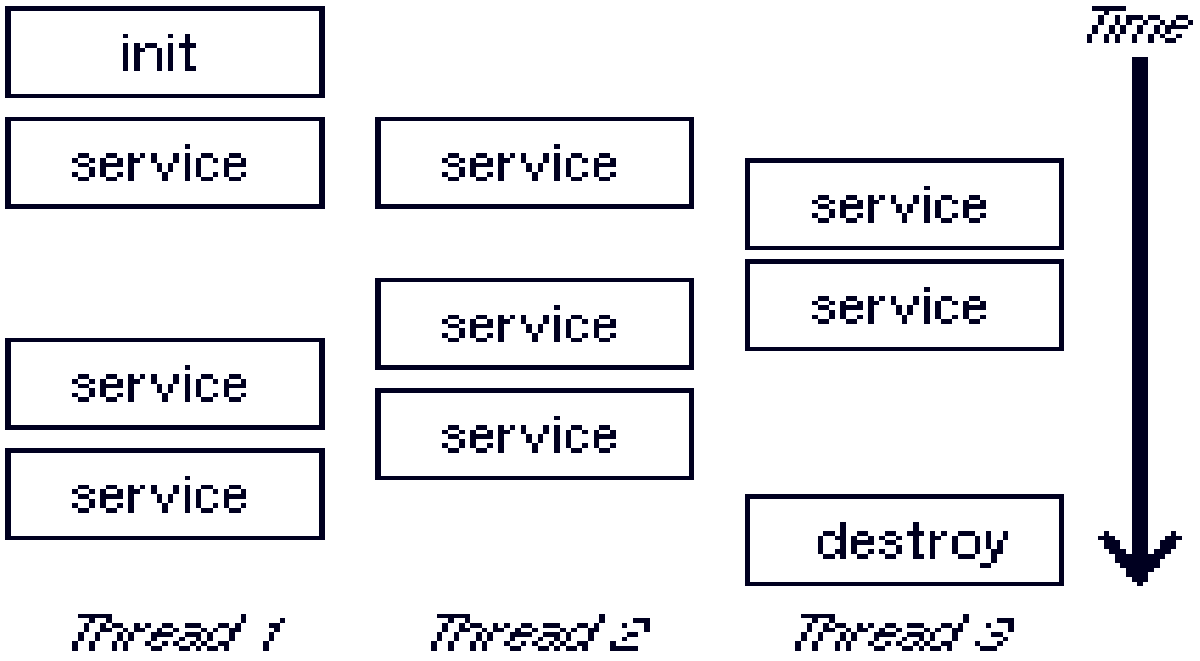


Request and Response – GET v/s POST

- **The HTTP request method determines whether doGet() or doPost() runs.**

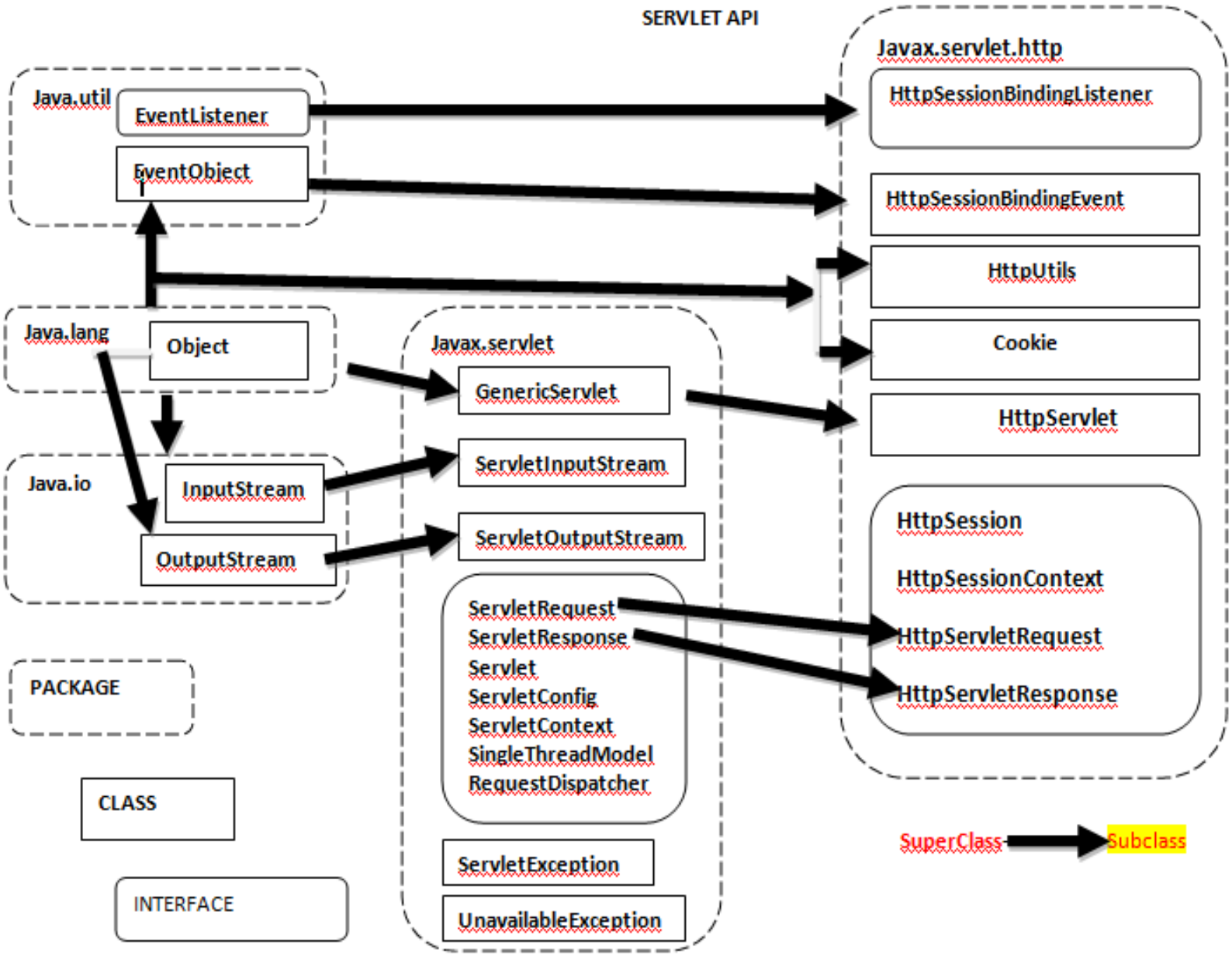
	GET (doGet())	POST (doPost())
HTTP Request	The request contains only the request line and HTTP header.	Along with request line and header it also contains HTTP body.
Parameter passing	The form elements are passed to the server by appending at the end of the URL.	The form elements are passed in the body of the HTTP request.
Size	The parameter data is limited (the limit depends on the container)	Can send huge amount of data to the server.
Idempotency	GET is Idempotent	POST is not idempotent
Usage	Generally used to fetch some information from the host.	Generally used to process the sent data.

A Typical Servlet Lifecycle



Servlet Life Cycle Steps

1. Server loads the servlet
2. Creates 1 or more instances of class
3. Calls the init()
4. Servlet request is received
5. Calls the service method
6. Service() processes the request
7. Waits for next request or unloads it self
8. Use destroy() to unload.


SERVLET API
Javax.servlet.http
Java.util
EventListener
EventObject
HttpSessionBindingListener
HttpSessionBindingEvent
HttpUtils
Cookie
Java.lang
Object
Javax.servlet
GenericServlet
HttpServlet
Java.io
InputStream
OutputStream
ServletInputStream
ServletOutputStream
HttpSession
HttpSessionContext
HttpServletRequest
HttpServletResponse
ServletRequest
ServletResponse
Servlet
ServletConfig
ServletContext
SingleThreadModel
RequestDispatcher
PACKAGE
CLASS
INTERFACE
ServletException
UnavailableException
SuperClass
Subclass

RequestDispatcher Interface

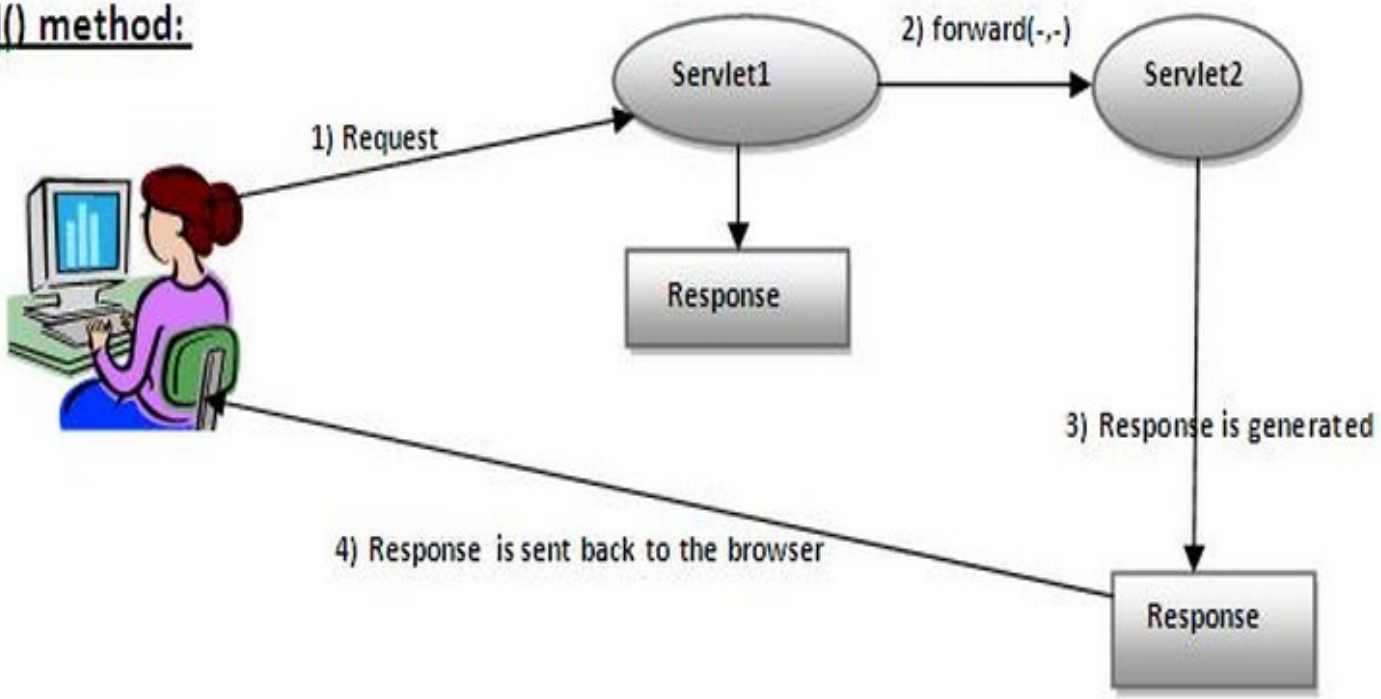
The RequestDispacher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp.This interface can also be used to include the content of antoher resource also. It is one of the way of servlet collaboration.

Methods of RequestDispatcher interface

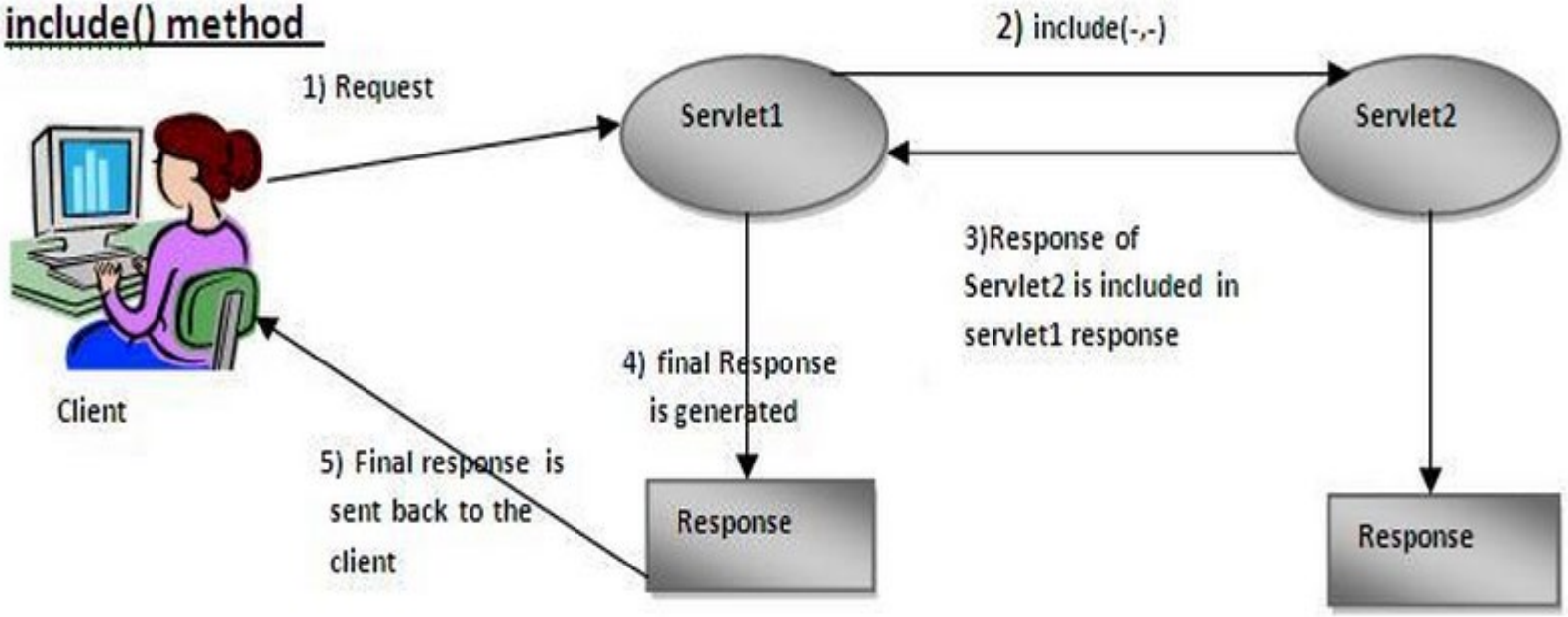
The RequestDispatcher interface provides two methods. They are:

1. **public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
2. **public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Includes the content of a resource (servlet, JSP page, or HTML file) in the response.

forward() method:



include() method



How to get the object of RequestDispatcher

The getRequestDispatcher() method of ServletRequest interface returns the object of RequestDispatcher.
Syntax:

Syntax of getRequestDispatcher method

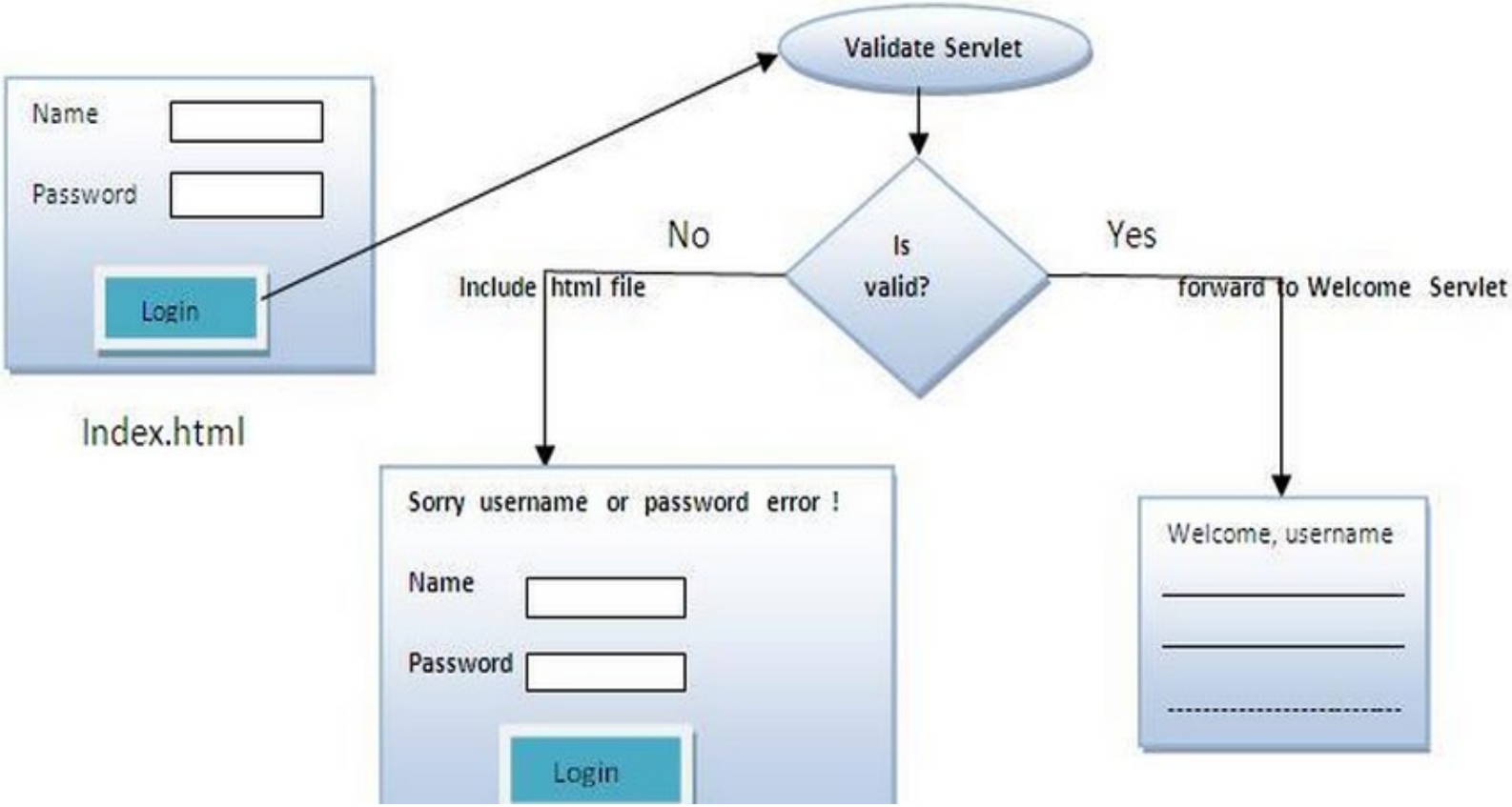
```
public RequestDispatcher getRequestDispatcher(String resource);
```

Example of using getRequestDispatcher method

```
RequestDispatcher rd=request.getRequestDispatcher("servlet2");
//servlet2 is the url-pattern of the second servlet

rd.forward(request, response);//method may be include or forward
```

- **index.html file:** for getting input from the user.
- **Login.java file:** a servlet class for processing the response. If password is servet, it will forward the request to the welcome servlet.
- **WelcomeServlet.java file:** a servlet class for displaying the welcome message.
- **web.xml file:** a deployment descriptor file that contains the information about the servlet.


Validate Servlet
Name
Password
Login
Index.html
No
Is
valid?
Yes
Include html file
forward to Welcome Servlet
Sorry username or password error !
Name
Password
Login
Welcome, username

Index.html

index.html

```
<form action="servlet1" method="post">
Name:<input type="text" name="userName"/><br/>
Password:<input type="password" name="userPass"/><br/>
<input type="submit" value="login"/>
</form>
```

```
public class Login extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String n=request.getParameter("userName");
        String p=request.getParameter("userPass");

        if(p.equals("servlet")){
                RequestDispatcher rd=request.getRequestDispatcher("servlet2");
                rd.forward(request, response);
        }
        else{
                out.print("Sorry UserName or Password Error!");
                RequestDispatcher rd=request.getRequestDispatcher("/index.html");
                rd.include(request, response);

                }
        }
```

WelcomeServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class WelcomeServlet extends HttpServlet {

        public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String n=request.getParameter("userName");
        out.print("Welcome "+n);
        }

}
```

Web.xml

```
<web-app>
 <servlet>
    <servlet-name>Login</servlet-name>
    <servlet-class>Login</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>WelcomeServlet</servlet-name>
    <servlet-class>WelcomeServlet</servlet-class>
  </servlet>


  <servlet-mapping>
    <servlet-name>Login</servlet-name>
    <url-pattern>/servlet1</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>WelcomeServlet</servlet-name>
    <url-pattern>/servlet2</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
   <welcome-file>index.html</welcome-file>
  </welcome-file-list>
```