Parul®
University

Advance Java

Prof. Gaurav Kumar, Assistant Professor
IT and Computer Science

Unit -2

Java Servlets

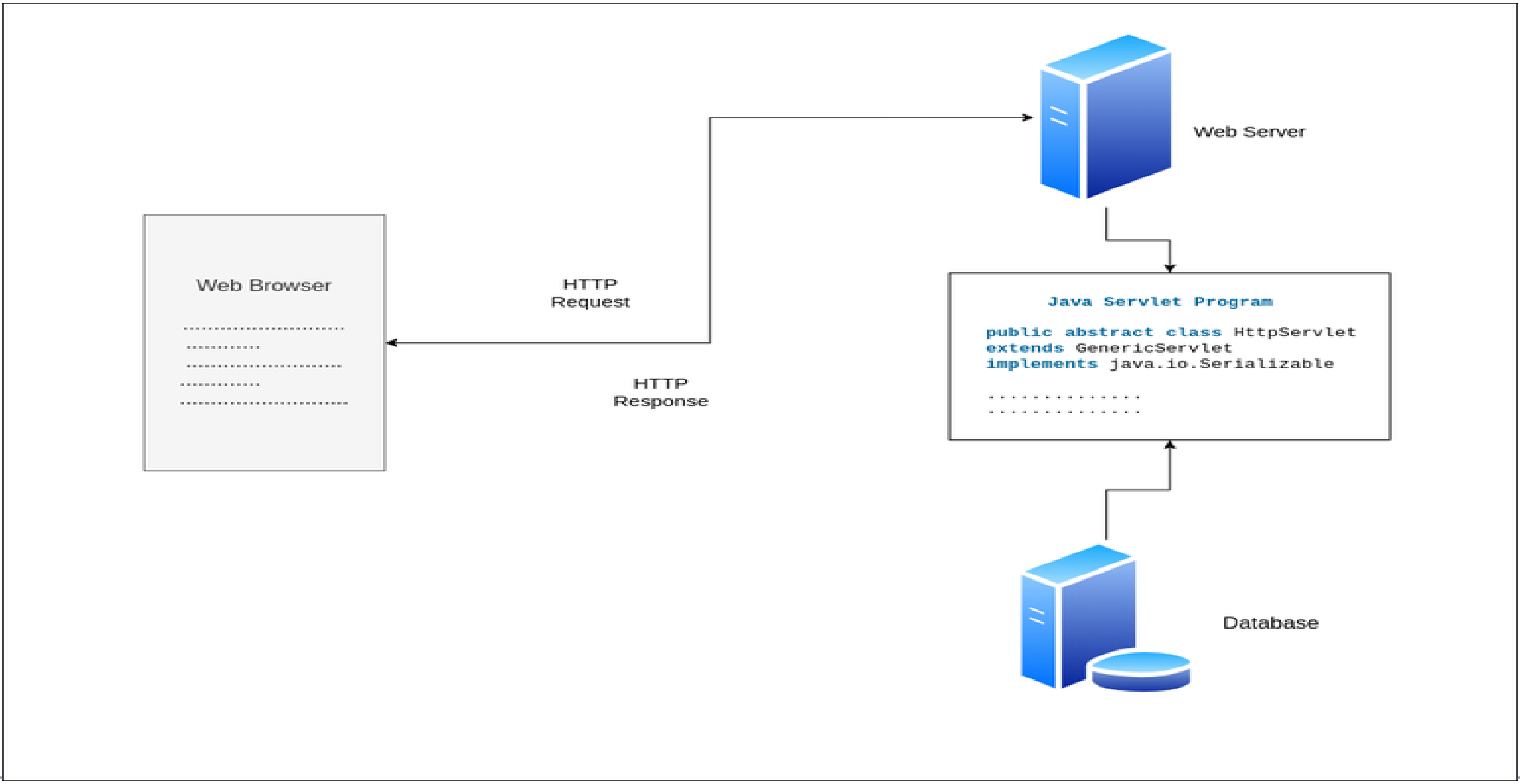
Introduction to Servlets

- Java Servlets are the Java programs that run on the Java-enabled web server or application server.
- They are used to handle the request obtained from the web server, process the request, produce the response, and then send a response back to the web server.
- Servlets are written in Java and are

Parul® University

DIGITAL LEARNING CONTENT

Java Servlets Architecture



```
public abstract class HttpServlet
extends GenericServlet
implements java.io.Serializable

. . . . . . . . . . . . . .
. . . . . . . . . . . . . .
```

Execution of Java Servlets

Execution of Servlets basically involves Six basic steps:

1. The Clients send the request to the Web Server.
2. The Web Server receives the request.
3. The Web Server passes the request to the corresponding servlet.
4. The Servlet processes the request and generates the response in the form of output.
5. The Servlet sends the response back to the webserver.
6. The Web Server sends the response back to the

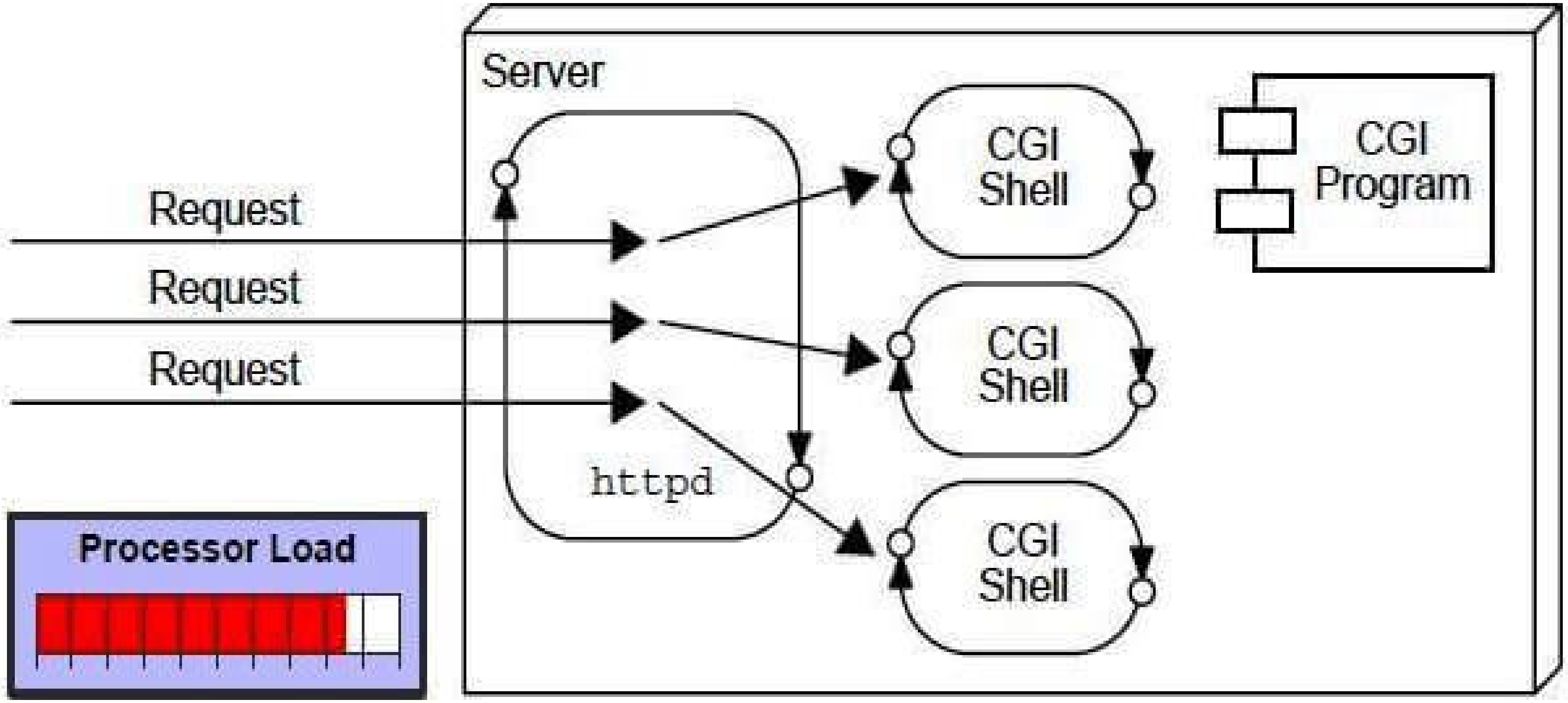
CGI(Common Gateway Interface)

CGI is actually an external application that is written by using any of the programming languages like **C** or **C++** and this is responsible for processing client requests and generating dynamic content.

In CGI application, when a client makes a request to access dynamic Web pages, the Web server performs the following operations:

- It first locates the requested web page *i.e* the required CGI application using URL.
- It then creates a new process to service the client's request.
- Invokes the CGI application within the process and passes the request information to the application.
- Collects the response from the CGI application

CGI(Common Gateway Interface)



Disadvantages of CGI

There are many problems in CGI technology:

- If the number of clients increases, it takes more time for sending the response.
- For each request, it starts a process, and the web server is limited to start processes.

Advantages of Servlet

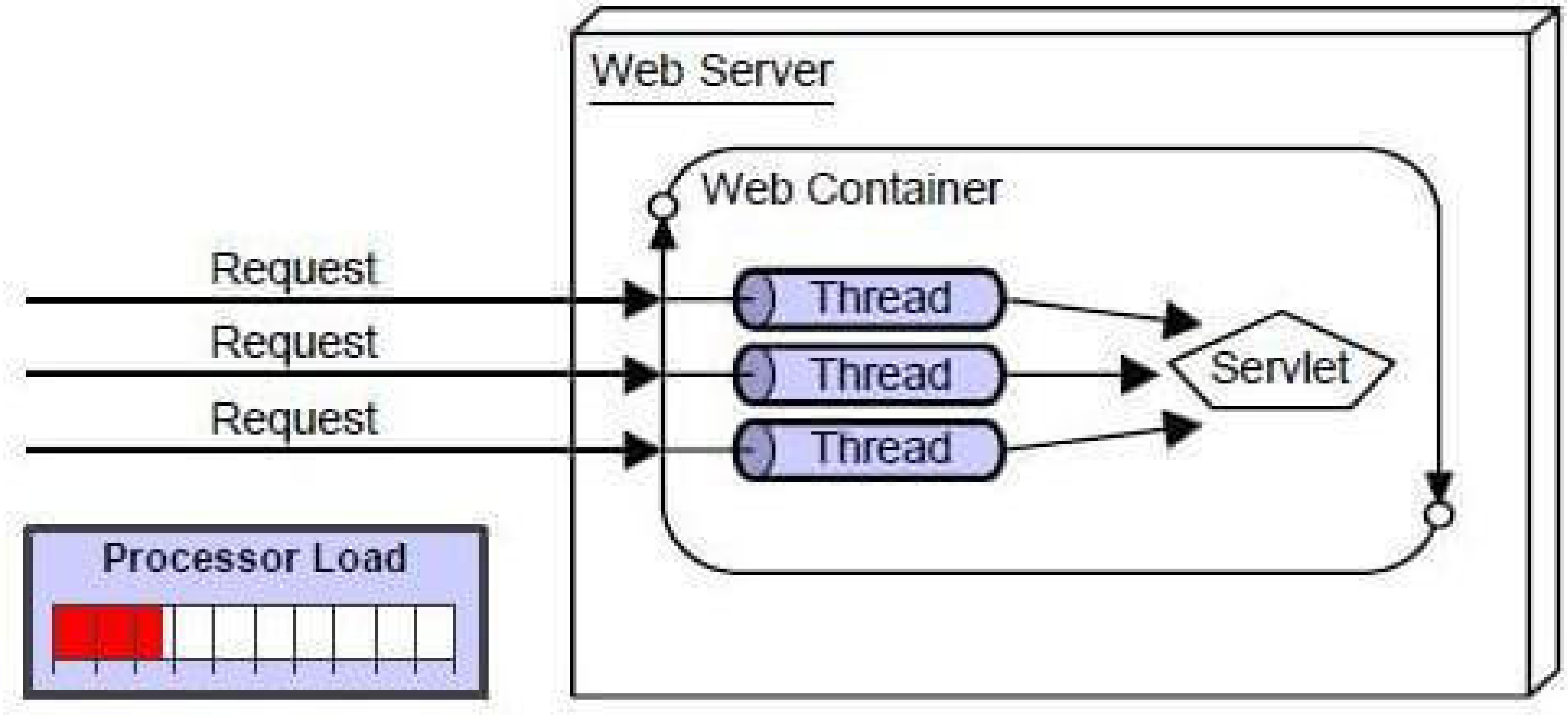
The advantages of Servlet are as follows:

Better performance: because it creates a thread for each request, not process.

Robust: JVM manages Servlets, so we don't need to worry about the memory leak, GarbageCollection, etc.

Secure & platform independent : because it uses java language.

Advantages of Servlet



Web Terminology

Website is a collection of related web pages that may contain text, images, audio and video.

A website can be of two types:

- Static Website is the basic type of website in which the codes are fixed for each page so the information contained in the page does not change and it looks like a printed page.
- Dynamic Website is a collection of dynamic web pages whose content changes dynamically. It accesses content from a database or Content Management System (CMS). Therefore, when you

Server

- Server is a device or a computer program that accepts and responds to the request made by other program, known as client.
- It is used to manage the network resources and for running the program or software that provides services.
- There are two types of servers:

1. Web Server
2. Application Server

Web Server

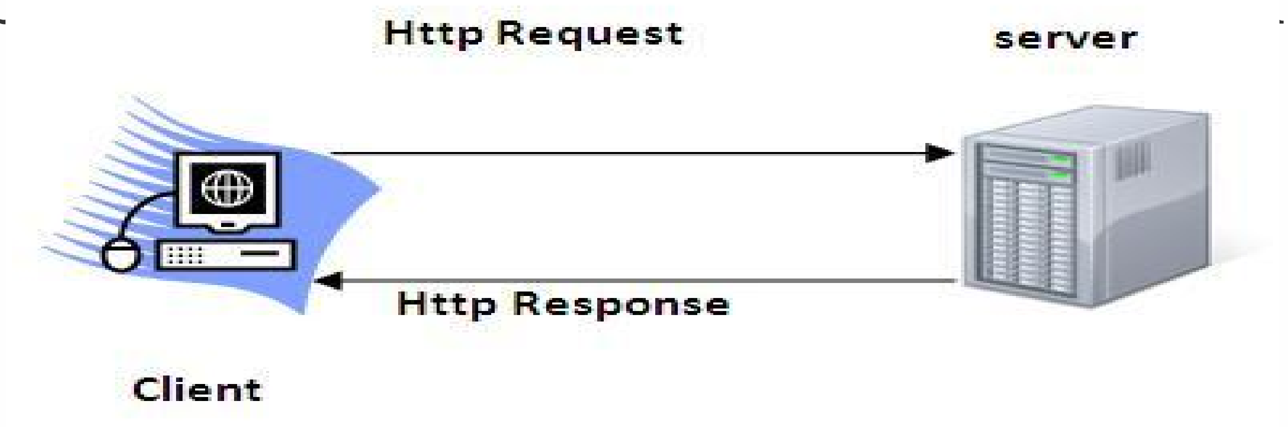
- Web server contains only web or servlet container. It can be used for servlet, jsp, struts, jsf etc. It can't be used for EJB.
- It is a computer where the web content can be stored. In general web server can be used to host the web sites but there also used some other web servers also such as FTP, email, storage, gaming etc
- Examples of Web Servers are: **Apache Tomcat** and **Resin**.

Application Server

- Application server contains Web and EJB containers. It can be used for servlet, jsp, struts, jsf, ejb etc. It is a component based product that lies in the middle-tier of a server centric architecture.
- It provides the middleware services for state maintenance and security, along with persistence and data access. It is a type of server designed to install, operate and host associated services and applications for the IT services, end users and organizations.
- Examples of Application Servers are: Jboss, Glassfish, Weblogic etc.

HTTP

- HTTP is TCP/IP based communication protocol, which is used to deliver the data like image files, query results, HTML files etc on the World Wide Web with the default port is TCP 80.
- It is the data communication protocol used to estal d server.



HTTP Request

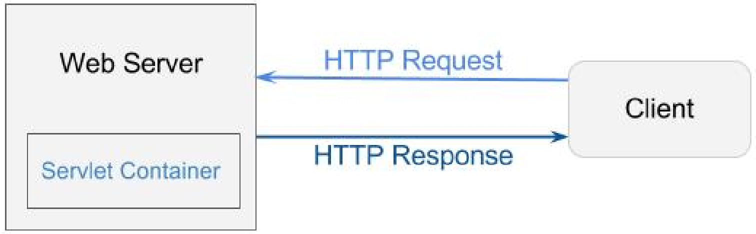
- The request sent by the computer to a web server, contains all sorts of potentially interesting information; it is known as HTTP requests.
- The HTTP request method indicates the method to be performed on the resource identified by the **Requested URI (Uniform Resource Identifier)**. This method is case-sensitive and should be used in uppercase.
- The HTTP request methods are:
- **GET**:- Asks to get the resource at the requested URL.

HTTP Request

- **POST**:-Asks the server to accept the body info attached. It is like GET request with extra info sent with the request.
- **PUT**:- Says to put the enclosed info (the body) at the requested URL.
- **DELETE:-** Says to delete the resource at the requested URL.

Servlet Container

- It provides the runtime environment for JavaEE (j2ee) applications.
- It manages the lifecycle of a Servlet.
- It handles authorization and authentication of resource access.
- It provides the service of decoding and encoding MIME-based messages.



Servlet Life Cycle

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. **Servlet class is loaded**

   The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

2. **Servlet instance is created**

   The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

Servlet Life Cycle

3. **init method is invoked**

The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet. Servlet interface. Syntax of the init method is given below:

```
public void init(ServletConfig config) throws ServletException
```

Servlet Life Cycle

4. **service method is invoked**

The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

```
public void service(ServletRequest request,
 ServletResponse response)
throws ServletException, IOException
```

Servlet Life Cycle

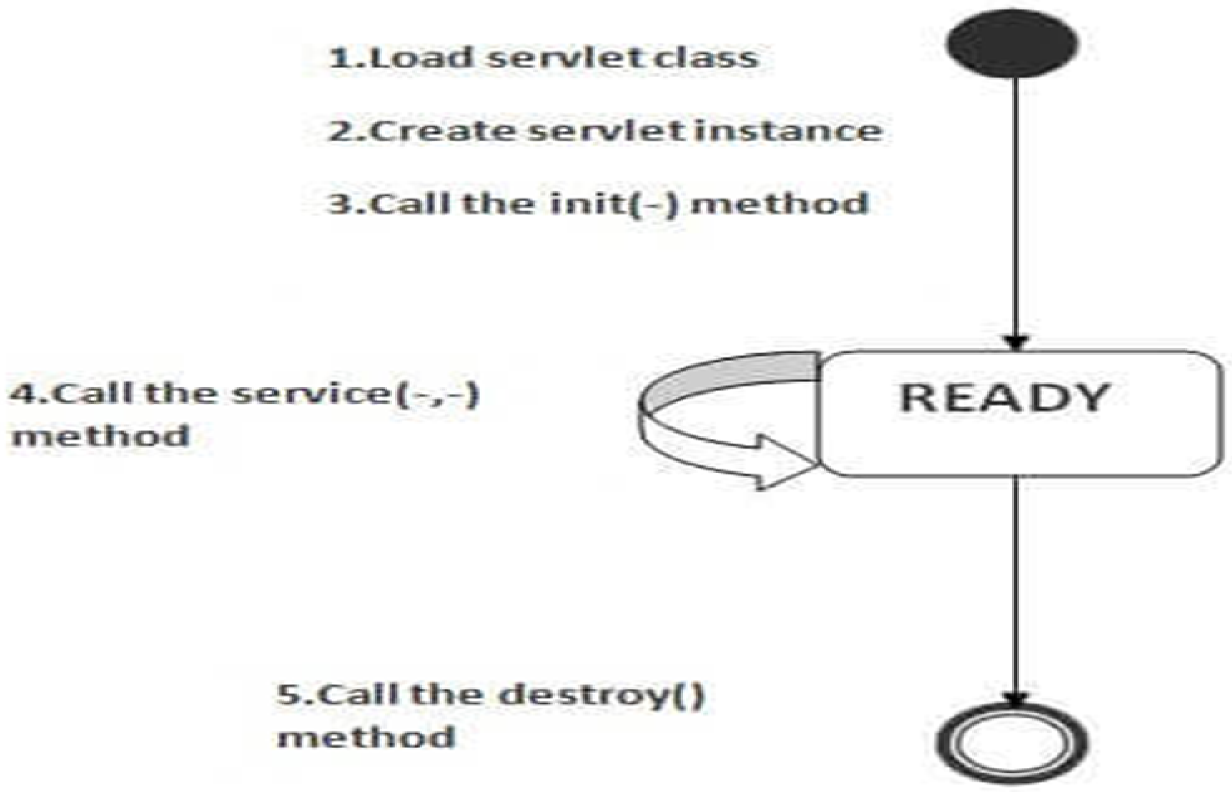
5. **destroy method is invoked**
The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

```
public void destroy()
```

Parul® University

DIGITAL LEARNING CONTENT

Servlet Life Cycle


1.Load servlet class
2.Create servlet instance
3.Call the init(-) method
4.Call the service(-,-) method
READY
5.Call the destroy() method

Servlet API

- The javax.servlet and javax.servlet.http packages represent interfaces and classes for servlet api.(after tomcat 10 and later jakarta.servlet and jakarta.servlet.http).
- The **jakarta.servlet** package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.
- The **jakarta.servlet.http** package contains interfaces and classes that are responsible for http requests only.

Servlet Interface

- Servlet interface provides common behavior to all the servlets.Servlet interface defines methods that all servlets must implement.
- Servlet interface needs to be implemented for creating any servlet (either directly or indirectly).
- It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

Methods of Servlet Interface

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request, ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

Parul® University

DIGITAL LEARNING CONTENT

Example of Servlet Interface

```
import jarkarta.servlet.*;
public class First implements Servlet{
ServletConfig config=null;
public void init(ServletConfig config){
this.config=config;
System.out.println("servlet is initialized");  }
public void service(ServletRequest req,ServletResponse res)
throws IOException,ServletException{
res.setContentType("text/html");  }
public void destroy(){System.out.println("servlet is destroyed");}
public ServletConfig getServletConfig(){return config;}
public String getServletInfo(){return "copyright 2007-1010";}
```

GenericServlet class

- GenericServlet class implements Servlet, ServletConfig and Serializable interfaces.
- It provides the implementation of all the methods of these interfaces except the service method.
- GenericServlet class can handle any type of request so it is protocol-independent.
- You may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

Example of GenericServlet class

```
import java.io.*;
import jakarta.servlet.*;
public class First extends GenericServlet{
public void service(ServletRequest req,
ServletResponse res)
throws IOException,ServletException{
res.setContentType("text/html");
PrintWriter out=res.getWriter();
out.print("<html><body>");
out.print("<b>hello generic servlet</b>");
out.print("</body></html>");  } }
```

HttpServlet class

- The HttpServlet class extends the GenericServlet class and implements Serializable interface.
- HTTP Servlets are designed to seamlessly work within the HTTP protocol framework, enabling robust and efficient web applications.
- Key Methods in HTTP Servlet

doGet(HttpServletRequest req, HttpServletResponse res): Handles GET requests.

doPost(HttpServletRequest req, HttpServletResponse res): Handles POST requests.

doPut, doDelete: Handle respective HTTP methods.

Example of HttpServlet class

```
import java.io.*;
import jakarta.servlet.*;

@WebServlet("/example")
public class ExampleServlet extends HttpServlet {
protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException {
response.getWriter().println("Hello, Servlet!");
}
}
```

ServletConfig Interface

- An object of ServletConfig is created by the web container for each servlet. This object can be used to get configuration information from web.xml file.
- The core advantage of ServletConfig is that you don't need to edit the servlet file if information is modified from the web.xml file.

Syntax:-

public ServletConfig getServletConfig();

ServletContext Interface

- An object of ServletContext is created by the web container at time of deploying the project. This object can be used to get configuration information from web. xml file. There is only one ServletContext object per web application.
- It is easy to maintain if any information is shared to all the servlet, it is better to make it available for all the servlet. We provide this information from the web.xml file, so if the information is changed, we don't need to modify the servlet.
- **getServletContext() method** of ServletConfig interface returns the object of ServletContext.

RequestDispatcher Interface

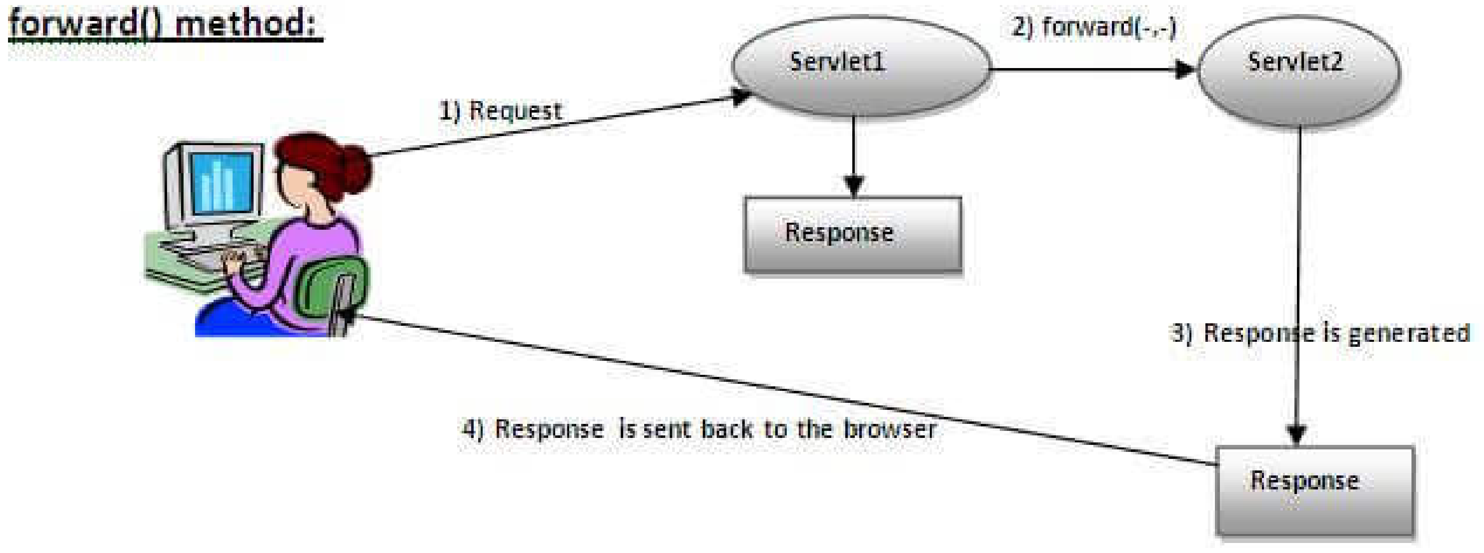
- The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp.
- This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.

There are two methods defined in the RequestDispatcher interface.

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.

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.

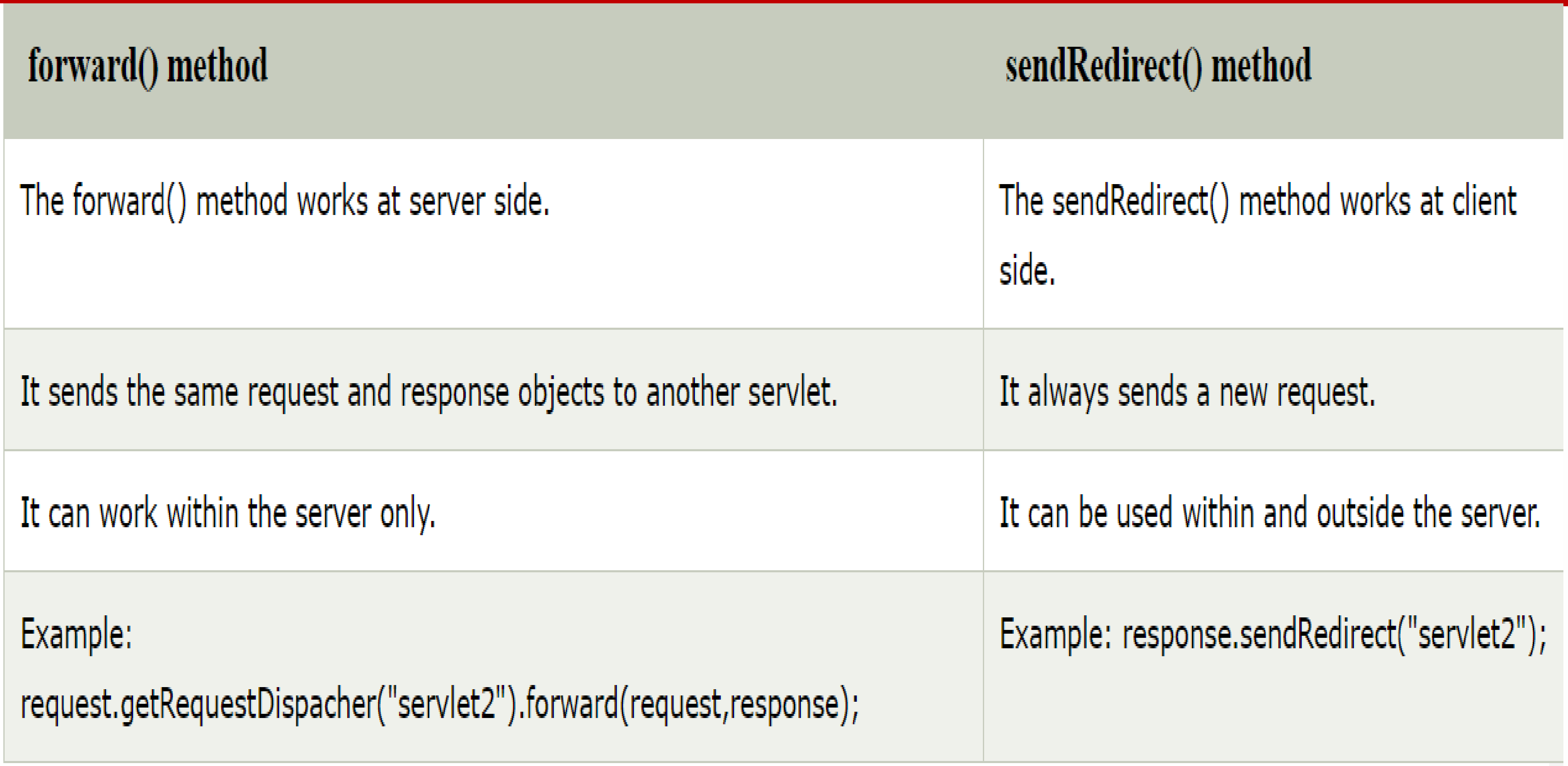
RequestDispatcher Interface



sendRedirect

- The sendRedirect() method of HttpServletResponse interface can be used to redirect response to another resource, it may be servlet, jsp or html file.
- It accepts relative as well as absolute URL.
- It works at client side because it uses the url bar of the browser to make another request. So, it can work inside and outside the server.

sendRedirect() VS forward()



forward() method	sendRedirect() method
The forward() method works at server side.	The sendRedirect() method works at client side.
It sends the same request and response objects to another servlet.	It always sends a new request.
It can work within the server only.	It can be used within and outside the server.
Example: request.getRequestDispacher("servlet2").forward(request,response);	Example: response.sendRedirect("servlet2");

Session Tracking in Servlets

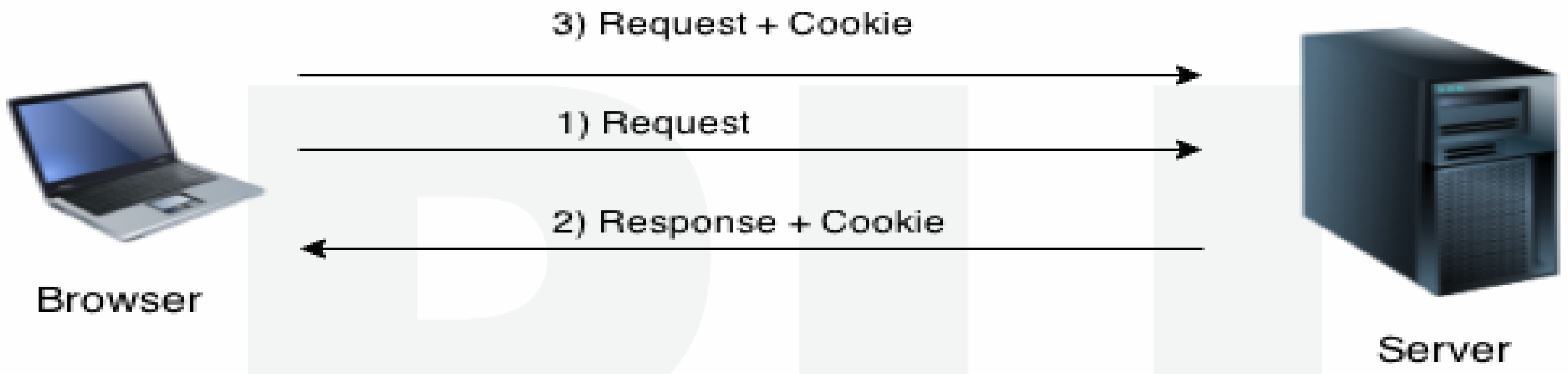
- Session simply means a particular interval of time.
- Session Tracking is a way to maintain state (data) of an user. It is also known as session management in servlet.
- Http protocol is a stateless so we need to maintain state using session tracking techniques. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.
- There are four techniques used in Session tracking:

1. Cookies
2. Hidden Form Field
3. URL Rewriting
4. HttpSession

Cookies

- A **cookie** is a small piece of information that is persisted between the multiple client requests.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.
- By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.

Cookies



There are 2 types of cookies in servlets.
Non-persistent cookie:- It is **valid for single session** only. It is removed each time when user closes the browser.
Persistent cookie:It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Cookies

Advantage of Cookies

Simplest technique of maintaining the state.

Cookies are maintained at client side.

Disadvantage of Cookies

It will not work if cookie is disabled from the browser.

Only textual information can be set in Cookie object.

Cookie Class

jakarta.servlet.http.Cookie class provides the functionality of using cookies. It provides a lot of useful methods for cookies.
For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:
public void addCookie(Cookie ck):method of HttpServletResponse interface is used to add cookie in response object.
public Cookie[] getCookies():method of HttpServletRequest interface is used to return all

Cookie Class

Constructor	Description
Cookie()	constructs a cookie.
Cookie(String name, String value)	constructs a cookie with a specified name and value.
Method	Description
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

Hidden Form Field

- In case of Hidden Form Field a hidden (invisible) textfield is used for maintaining the state of an user.
- In such case, we store the information in the hidden field and get it from another servlet. This approach is better if we have to submit form in all the pages and we don't want to depend on the browser.
- Example:

```
<input type="hidden" name="uname" value="Vim
```

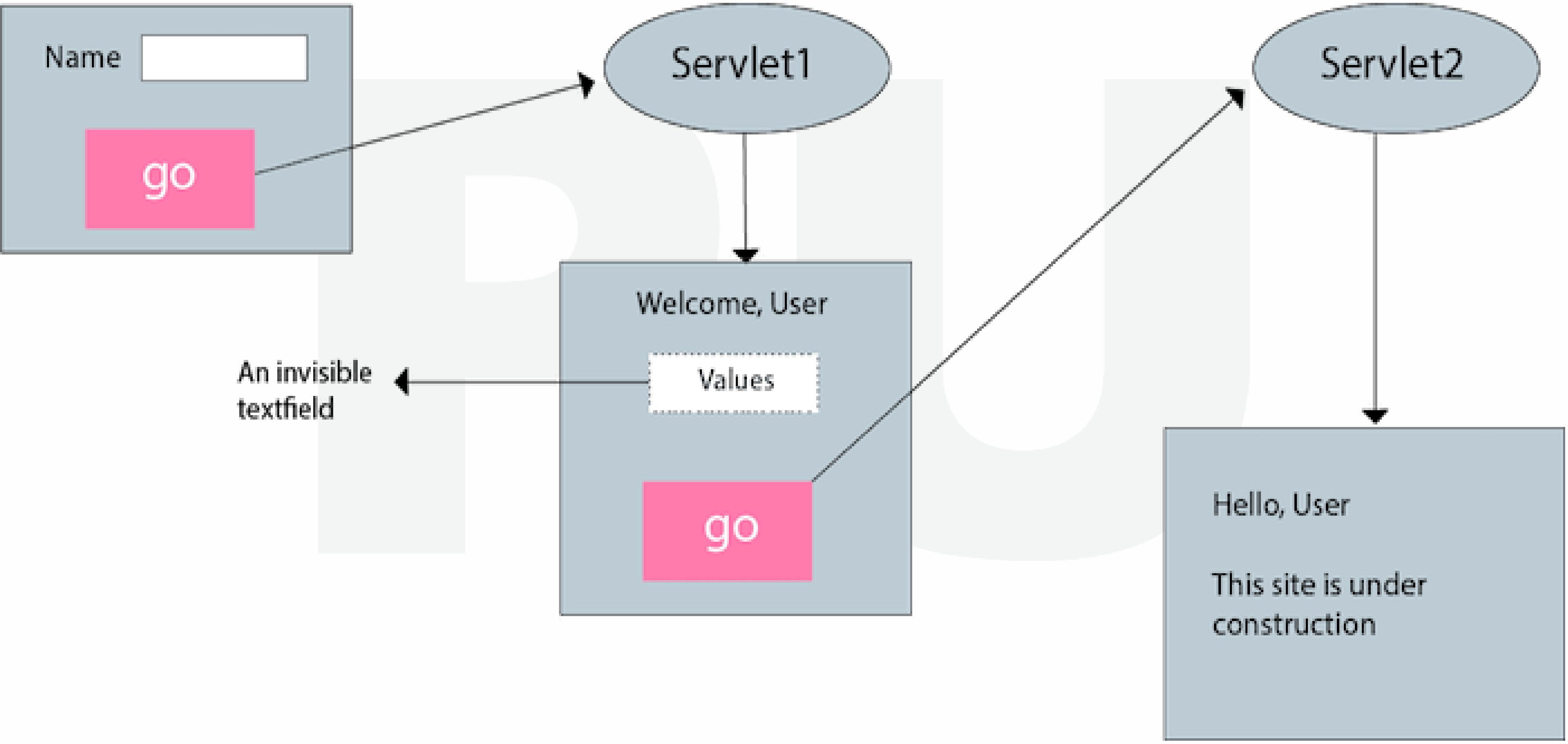
Hidden Form Field

Advantage of Hidden Form Field:
It will always work whether cookie is disabled or not.
Disadvantage of Hidden Form Field:
It is maintained at server side.
Extra form submission is required on each pages.
Only textual information can be used.

Parul® University

DIGITAL LEARNING CONTENT

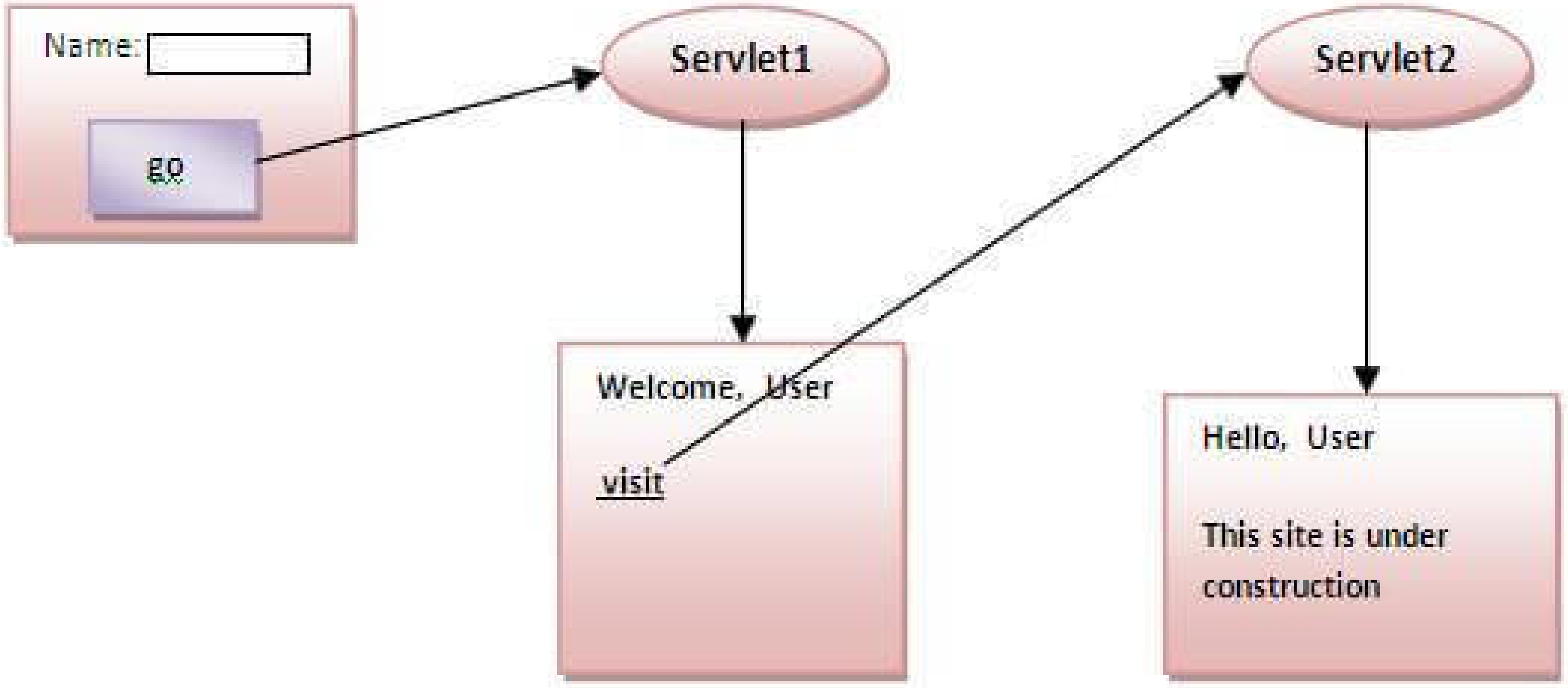
Hidden Form Field



URL Rewriting

- In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource. We can send parameter name/ value pairs using the following format:
- url?name1=value1&name2=value2&??
- A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&). When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can

URL Rewriting



URL Rewriting

Advantage of URL Rewriting

It will always work whether cookie is disabled or not (browser independent).

Extra form submission is not required on each pages.

Disadvantage of URL Rewriting

It will work only with links.

It can send Only textual information.

Http Session Interface

- In Http Session , container creates a session id for each user. The container uses this id to identify the particular user.
- An object of HttpSession can be used to perform two tasks:

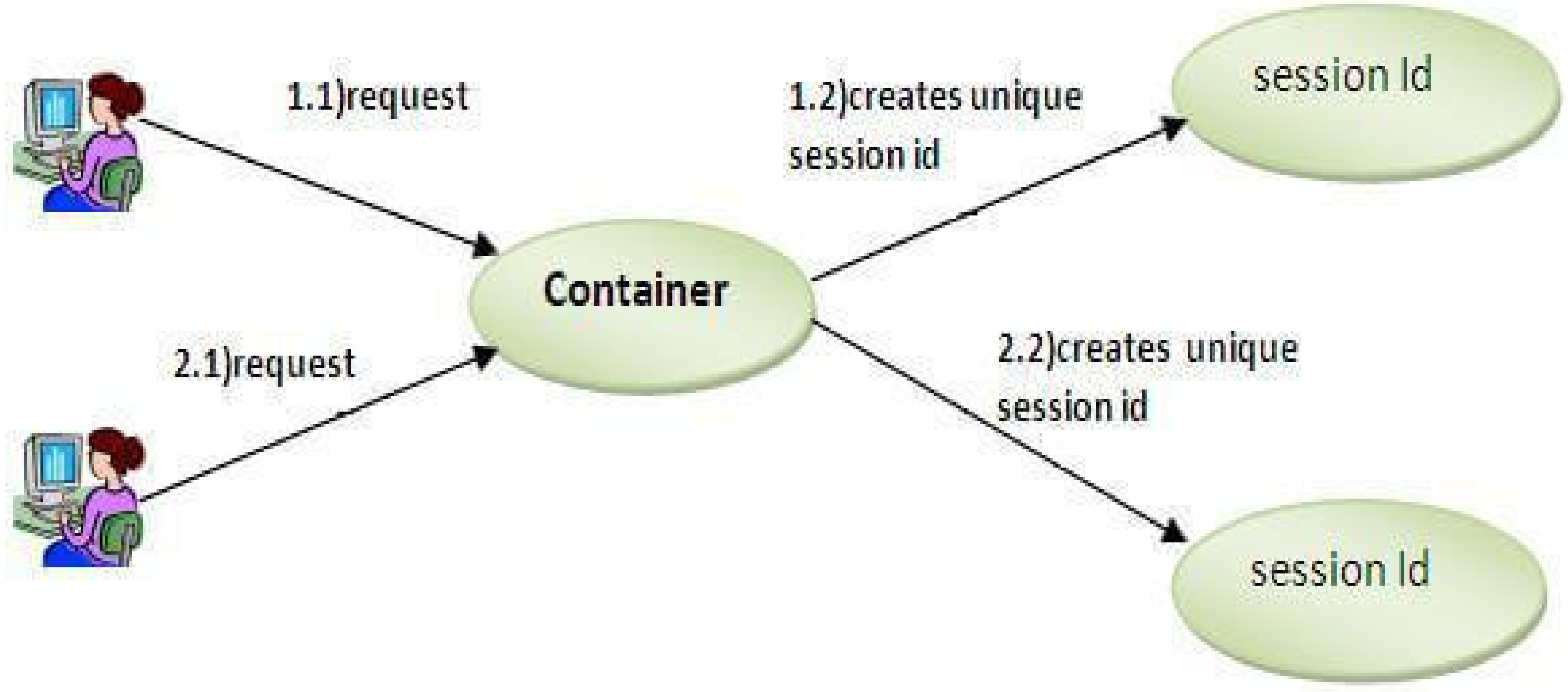
bind objects

view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.

Parul® University

DIGITAL LEARNING CONTENT

Http Session Interface



Http Session Interface

The HttpServletRequest interface provides two methods to get the object of HttpSession:

public HttpSession getSession(): Returns the current session associated with this request, or if the request does not have a session, creates one.

public HttpSession getSession(boolean create): Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

public String getId(): Returns a string containing the unique identifier value.

public long getCreationTime(): Returns the time when this session was created, measured in milliseconds since

Servlet Filter

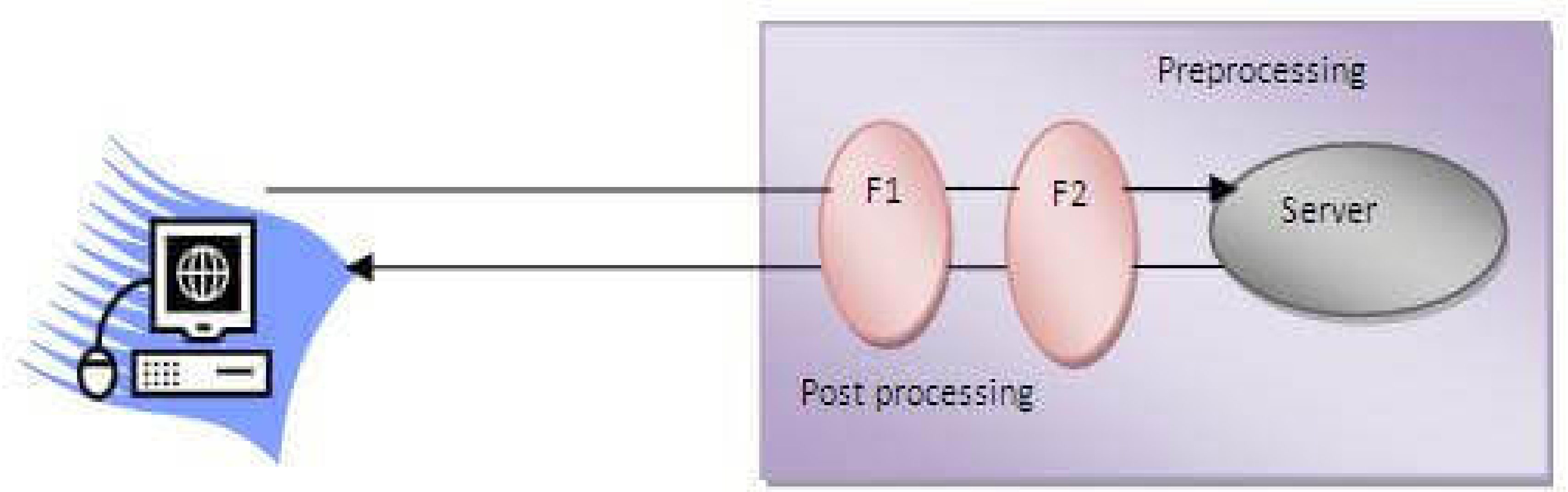
- A filter is an object that is invoked at the preprocessing and postprocessing of a request.
- It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc.
- The servlet filter is pluggable, i.e. its entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet.

Servlet Filter

Usage of Filter
recording all incoming requests
logs the IP addresses of the computers from which the requests originate
conversion
data compression
encryption and decryption
input validation etc.

Advantage of Filter
Filter is pluggable.
One filter don't have dependency onto another resource.
Less Maintenance

Servlet Filter



Filter API

Like servlet filter have its own API. The javax.servlet package contains the three interfaces of Filter API.

1) Filter interface

For creating any filter, you must implement the Filter interface. Filter interface provides the life cycle methods for a filter.

Method	Description
public void init(FilterConfig config)	init() method is invoked only once. It is used to initialize the filter.
public void doFilter(HttpServletRequest request,HttpServletResponse response, FilterChain chain)	doFilter() method is invoked every time when user request to any resource, to which the filter is mapped.It is used to perform filtering tasks.
public void destroy()	This is invoked only once when filter is taken out of the service.

Filter API

2) FilterChain interface

- The object of FilterChain is responsible to invoke the next filter or resource in the chain.
- This object is passed in the doFilter method of Filter interface.
- The FilterChain interface contains only one method:

public void doFilter(HttpServletRequest request, HttpServletResponse response): it passes the control to the next filter or resource.

Filter API

3) FilterConfig

An object of FilterConfig is created by the web container. This object can be used to get the configuration information from the web.xml file.

Methods of FilterConfig interface

There are following 3 methods in the FilterConfig interface.

public void init(FilterConfig config): init() method is invoked only once it is used to initialize the filter.

public String getInitParameter(String parameterName): Returns the parameter value for the specified parameter name.

public ServletContext getServletContext(): Returns the ServletContext object

Parul® University

Creating Custom filter

Steps:

1. Implement Filter interface.
2. Override methods: init(), doFilter(), and destroy().
3. Register the filter in web.xml or use annotations.

Example Code:

```
public class MyFilter implements Filter {
   public void init(FilterConfig config) throws
ServletException {
      // Initialization code
   }
   public void doFilter(ServletRequest request,
ServletResponse response, FilterChain chain)
         throws IOException, ServletException {
```

Creating Custom filter

```
// Pre-processing
        chain.doFilter(request, response); // Forward to the next
filter or servlet
        // Post-processing
    }
    public void destroy() {
        // Cleanup code
    }
}
```

Mapping Filters to Servlets

We can map Filters using two ways:

1. Declarative Mapping (web.xml):

```
<filter>
    <filter-name>MyFilter</filter-name>
    <filter-class>com.example.MyFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>MyFilter</filter-name>
    <servlet-name>MyServlet</servlet-name>
</filter-mapping>
```

Mapping Filters to Servlets

2. Annotation-Based Mapping:

```
@WebFilter(servletNames = {"MyServlet"})
public class MyFilter implements Filter {
    // Implementation
}
```

Parul® University

DIGITAL LEARNING CONTENT

Mapping Filters to Specific URL Patterns

1. Declarative Mapping (web.xml):

```
<filter-mapping>
    <filter-name>MyFilter</filter-name>
    <url-pattern>/secure/*</url-pattern>
</filter-mapping>
```

2. Annotation-Based Mapping:

```
@WebFilter(urlPatterns = {"/secure/*"})
public class MyFilter implements Filter {
    // Implementation
}
```

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in