What is Servlet?`

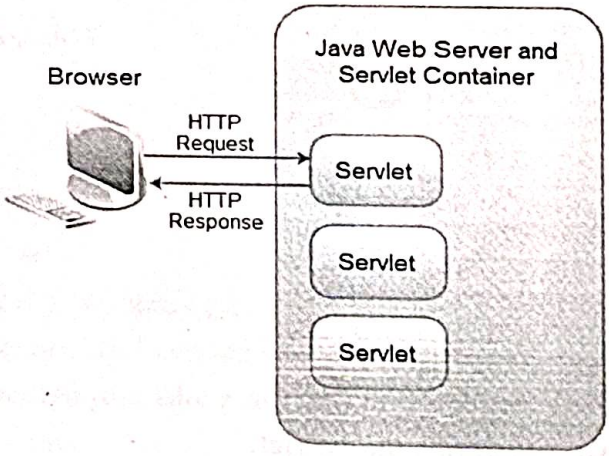
A Servlet is a Server Side Java program which extends the functionality of web application.

It is used to create dynamics web pages.

Key Features of Java Servlets:

- **Platform Independence:** Written in Java, servlets are platform-independent, ensuring compatibility across various operating systems.
- **Efficiency:** Unlike CGI scripts, servlets run within the server's process, eliminating the need to create a new process for each request, which enhances performance.
- **Robustness:** Leveraging Java's strong typing and exception handling, servlets provide a robust environment for web applications.

It runs inside a Servlet container



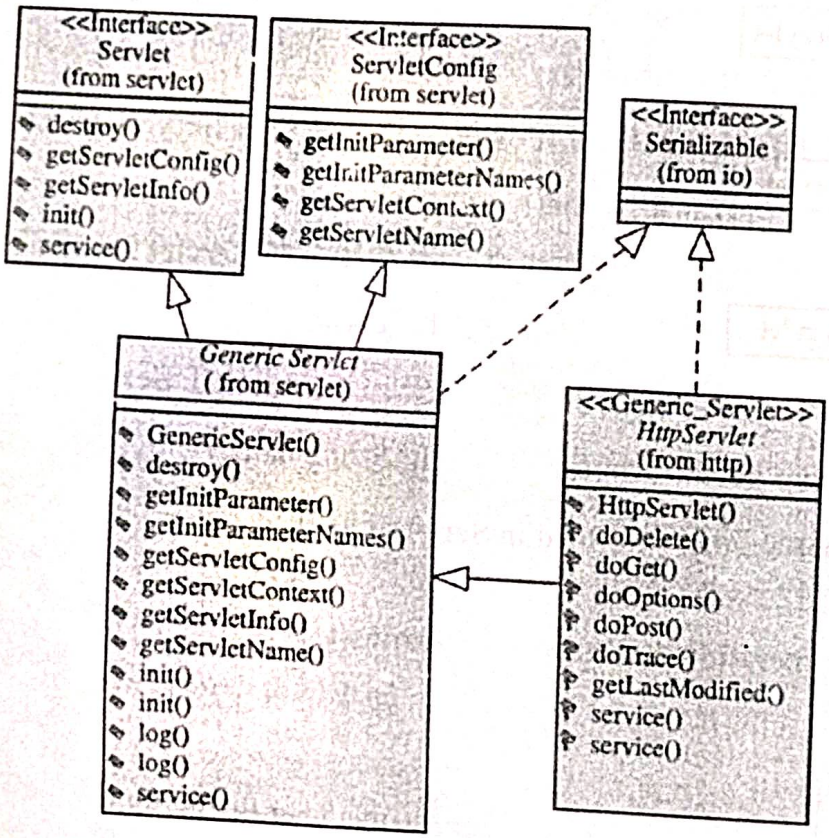
Servlets inside a Java Servlet Container

Servlet Containers: Java servlet containers are usually running inside a Java web server. A few common well known, free Java web servers are:

- Jetty
- Tomcat

The **javax.servlet** package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.

The **javax.servlet.http** package contains interfaces and classes that are responsible for http requests only.



Servlet API

What is the difference between Get and Post?

GET	POST
1) In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.
2)We can send only text data.	We can send text & binary data.

3) It not secured because data is exposed in URL bar.	It is more secured because data is not exposed in URL bar.
4) Get request can be bookmarked	Post request cannot be bookmarked
5) Get request is idempotent. It means second request will be ignored until response of first request is delivered.	Post request is non-idempotent
6) Get request is more efficient and used more than Post	Post request is less efficient and used less than get.

Note:- All information stored in the body of the request rather than in the URL,which provides for more privacy.POST method is used for sending credit card no., updated database etc.

Idempotent request:

By repeating the request multiple times , if there is no change in response such type of requests are Idempotent requests.

Ex: GET requests are Idempotent and POST requests are not Idempotent.

Safe request :

By repeating the same request multiple times , if there is no side-effect at server side , such type of requests are called safe-requests .

Ex: GET requests are safe , where as POST requests are not safe to repeat.

Triggers to send GET request :

1. Type url in the address bar and submit is always GET request.
2. Clicking hyperlink is always GET request.
3. Submitting the form , where method attribute specify with GET value is always GET request.

```
<form action="/test"  method="GET">
---------- ----
</form>
```

4. Submitting the form without method attribute is always GET request i.e., default method for form is GET.

How can you call a jsp from the servlet?

one of the way is RequestDispatcher interface for example:

```
RequestDispatcher rd=request.getRequestDispatcher("/login.jsp");
rd.forward(request,response);
```

SendRedirect

The sendRedirect() method of HttpServletResponseinterface 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.

Difference between forward() and sendRedirect() method

There are many differences between the forward() method of RequestDispatcher and sendRedirect() method of HttpServletResponse interface. They are given below:

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.
Example: request.getRequestDispacher("servlet2").forward (request,response);	Example: response.sendRedirect("servlet2");

Syntax of sendRedirect() method

```
response.sendRedirect("http://www.dacinfotech.com");
```

Servlet with Annotation (feature of servlet3):

Annotation represents the metadata. If you use annotation, deployment descriptor (web.xml file) is not required. But you should have tomcat7 as it will not run in the previous versions of tomcat. @WebServlet annotation is used to map the servlet with the specified name.

1. In Servlet 3.0, web.xml is optional.
2. Configuration part managed by annotation called @webServlet.
3. URL-pattern attribute is mandatory.
4. It reduces the complexity of web.xml

Example of simple servlet by annotation

There is given the simple example of servlet with annotation.

Simple.java

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
mport javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/Simple")
public class Simple extends HttpServlet {
   private static final long serialVersionUID = 1L;

   protected void doGet(HttpServletRequest request, HttpServletResponse re
sponse)
                    throws ServletException, IOException {


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

      out.print("<html><body>");
      out.print("<h3>Hello Servlet</h3>");
      out.print("</body></html>");
   }
```

```
}
```

What are the annotations used in Servlet 3?

There are mainly 3 annotations used for the servlet.

1. @WebServlet : for servlet class.
2. @WebListener : for listener class.
3. @WebFilter : for filter class.

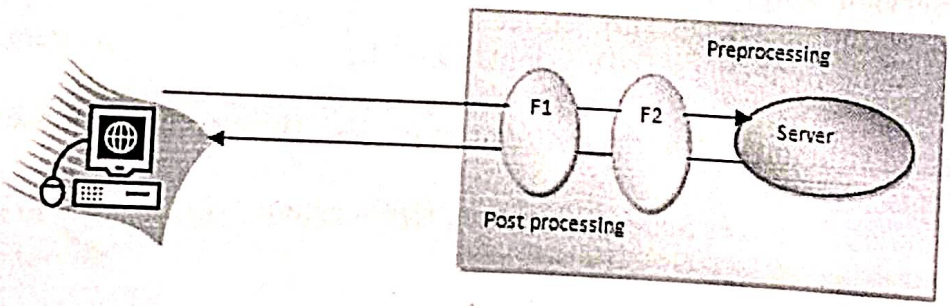
What is a Filter?

A filter is an object that is invoked at the preprocessing and post processing 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.

So maintenance cost will be less.



Note: Unlike Servlet, One filter doesn't have dependency on another 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

1. Filter is pluggable.
2. One filter don't have dependency onto another resource.
3. Less Maintenance

Filter API

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

1. Filter
2. FilterChain
3. FilterConfig

How to define Filter

We can define filter same as servlet. Let's see the elements of filter and filter-mapping.

```
<web-app>

<filter>
<filter-name>...</filter-name>
<filter-class>...</filter-class>
</filter>

<filter-mapping>
<filter-name>...</filter-name>
<url-pattern>...</url-pattern>
</filter-mapping>

</web-app>
```

Session Tracking in Servlets

Session simply means a connection for a particular interval of time.

Why Session Tracking?

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.

HTTP is stateless that means each request is considered as the new request.

Dynamic applications are built on the assumption that the same client accesses the application multiple times.

Client data is lost with every request being considered as a new request.

Session Tracking Techniques

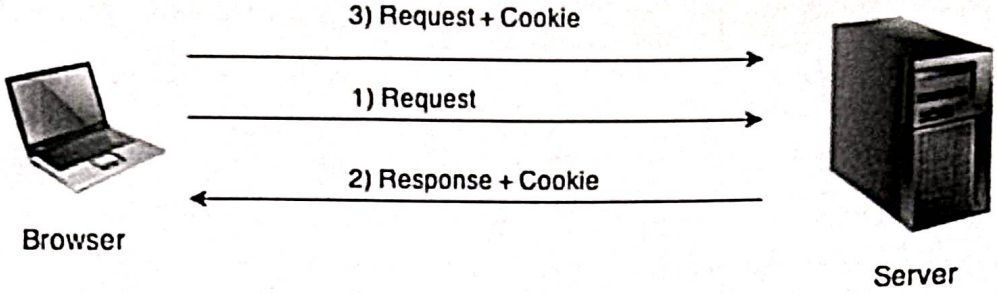
There are four techniques used in Session tracking:

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

1) Cookie

A cookie is a small piece of textual information sent by the server to the client, stored on the client, and returned by the client for all requests to the server.

- Stores data on the client-side (user's browser).
- Less secure than sessions for sensitive information (due to potential theft).



How to create Cookie?

Let's see the simple code to create cookie.

```
1. Cookie ck=new Cookie("user","kumar");//creating cookie object
2. response.addCookie(ck);//adding cookie in the response
```

How to delete Cookie?

Let's see the simple code to delete cookie. It is mainly used to logout or signout the user.

```
Cookie ck=new Cookie("user","");//deleting value of cookie
ck.setMaxAge(0);//changing the maximum age to 0 seconds
response.addCookie(ck);//adding cookie in the response
```

Adding Cookies to the response :

HttpServletResponse contains the following method to addCookie to the Response object.

Ex:

```
Cookie c=new Cookie(String name , String value) ;
response.addCookie(c);
```

How to get Cookies?

Let's see the simple code to get all the cookies.

```
Cookie ck[]=request.getCookies();
for(int i=0;i<ck.length;i++){
```

```
 out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and va
lue of cookie
}
```

URL Rewriting

```
// (Pseudocode, not recommended for real applications)

String sessionId = request.getSession().getId();

String targetUrl = "welcome.jsp?sessionid=" + sessionId;

response.sendRedirect(targetUrl);
```

Advantage of URL Rewriting

1. It will always work whether cookie is disabled or not (browser independent).
2. Extra form submission is not required on each pages.

Disadvantage of URL Rewriting

1. It will work only with links.
2. It can send Only textual information.
3. 1 Static page in the cycle cannot be the part of url rewriing
4. 2 User Book marks the Url

HttpSession interface

How to get the HttpSession object ?

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

1. public HttpSession getSession():Returns the current session associated with this request, or if the request does not have a session, creates one.
2. 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 class LoginServlet extends HttpServlet {

  protected void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    String username = request.getParameter("username");
    String password = request.getParameter("password");

    // (Assuming successful login logic)
    HttpSession session = request.getSession();
    session.setAttribute("loggedInUser", username);

    // Redirect to a welcome page
    response.sendRedirect("welcome.jsp");
  }
}
```

Hidden Form Fields in HTML

In this approach the unique token is embedded within each HTML form. For example

<input type="Hidden" name="uid" value="dac">

When the request is submitted, the server receives the token as part of the request, which in turn can be used to identify the client by matching the unique token. The servlet specification does not recommend this approach

Introduction to JSP UNIT 2

What is JSP?

1. JSP stands for Java Server Pages technology (JSP)
2. JSP is a combination of java and html
3. It is used to create dynamic web page.
4. It is an extension to the servlet technology.
5. A JSP page is internally converted into servlet.