SERVLETS PART-5

Servlet Forwarding:

Forwarding request & response of one servlet to another servlet is called as servlet forwarding.

The main advantage of servlet forwarding is modularity.

Servlet Including:

Including request & response of one servlet into another servlet is called as servlet including.

The main advantage of servlet including is reusability.

Forwarding Vs. Including

Servlet Forwarding	Servlet Including
1) Forwarding request & response of one servlet to another servlet is known as servlet forwarding.	1) Including request & response of one servlet into another servlet is known as servlet including.
2) The main advantage of servlet forwarding is modularity.	2) The main advantage of servlet including is reusability.
3) In forwarding, only one pair of request & response created by web container.	3) In including also only one pair of request & response created by web container.
4) Forwarding works within the server only.	4) Including also works within the server only.
5) Forwarding supports to forward servlet to server, JSP & HTML.	5) Including also supports to include servlet, JSP & HTML in a servlet.

Forwarding & Including Example:

login.html

```
<html>
<body bgcolor=yellow text=blue>
<center>
<h1><u>Login Form</u></h1>
<form method=POST action=login>
Username <input type=text name=uname><br>
Password <input type=password name=pword><br><br>
<input type=submit><input type=reset>
</form>
</center>
</body>
</html>
```

LoginServlet.java

```
import java.io.*;
import javax.servlet.*;
public class LoginServlet extends GenericServlet
{
	public void service(ServletRequest req, ServletResponse res)
	{
		try{
		String s1=req.getParameter("uname");
		String s2=req.getParameter("pword");
```

```
            ServletContext sc=getServletContext();
            if((s1.equals("abc"))&&(s2.equals("xyz")))
            {
                RequestDispatcher rd=sc.getRequestDispatcher("/welcome");
                rd.forward(req, res);
            }
            else
            {
                PrintWriter pw=res.getWriter();
                pw.println("<font color=red>");
                pw.println("Invalid Username/Password</font>");
                RequestDispatcher rd=sc.getRequestDispatcher("/login.html");
                rd.include(req, res);
            }
            }catch(Exception e)
            {
                System.err.println(e);
            }
        }
}
```

WelcomeServlet.java

```
import java.io.*;
import javax.servlet.*;
public class WelcomeServlet extends GenericServlet
```

```
{
	public void service(ServletRequest req, ServletResponse res)
	{
		try{
		String s=req.getParameter("uname");
		PrintWriter pw=res.getWriter();
		pw.println("<html><body bgcolor=cyan text=red><h1>");
		pw.println("Welcome "+s);
		pw.println("</h1></body></html>");
		}catch(Exception e)
		{
			System.err.println(e);
		}
	}
}
```

web.xml

```
<web-app>
<servlet>
<servlet-name>login</servlet-name>
<servlet-class>LoginServlet</servlet-class>
</servlet>
<servlet>
<servlet-name>welcome</servlet-name>
<servlet-class>WelcomeServlet</servlet-class>
```

```
</servlet>
<servlet-mapping>
<servlet-name>login</servlet-name>
<url-pattern>/login</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>welcome</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>
</web-app>
```

GenericServlet Vs HttpServlet

GenericServlet	HttpServlet
1) It is a protocol independant.	1) It is a http protocol dependant.
2) It supports common services. Examples: Forwarding & Including	2) It supports http specific services. Examples: Redirecting & Session Tracking
3) By using GenericServlet we can use common services only.	3) By using HttpServlet we can use both common services & http specific services because HttpServlet is a sub class of GenericServlet

javax.servlet.http.HttpServlet

Methods:

1) protected void doGet(javax.servlet.http.HttpServletRequest,
   javax.servlet.http.HttpServletResponse)
   throws javax.servlet.ServletException, java.io.IOException;
2) protected void doPost(javax.servlet.http.HttpServletRequest,
   javax.servlet.http.HttpServletResponse)
   throws javax.servlet.ServletException, java.io.IOException;
3) protected void service(javax.servlet.http.HttpServletRequest,
   javax.servlet.http.HttpServletResponse)
   throws javax.servlet.ServletException, java.io.IOException;
4) public void service(javax.servlet.ServletRequest,
   javax.servlet.ServletResponse) throws javax.servlet.ServletException,
   java.io.IOException;

doGet() method handles GET type requests only.

doPost() method handles POST type requests only.

service() method handles both GET & POST type requests.

To handle only GET type requests, override doGet() method only.

To handle only POST type requests, override doPost() method only.

To handle both GET & POST type requests and if the task is same then override non life cycle service() method only.

To handle both GET & POST type requests and if the task is different then override both doGet() & doPost() methods.