Unit -IV(Request Dispatching Methods)

Servlet Request Interface:

Working with Request parameters:

The ServletRequest interface provides the following methods to get request parameters:

String getParameter(String name) – Returns the value of a request parameter with the given name as a String format, or returns null value if the parameter does not exist.

String[] getparameter Values(String name) – returns an array of string objects containing all values in a request parameter or returns null value if the parameter does not exist.

Enumeration getParameterNames() –Returns an enumeration of string objects containing the names of the parameters in a request.

Working with Initialization parameters:

Servlet Container provides the ability to store startup and configuration information for a servlet in the form of initialization parameters , known as init parameters and context init parameters.

Init parameters are specific to Servlet configured in the deployment descriptor file, but the

Context init parameters are specific to all or multiple servlets configured in the deployment descriptor file.

Need for Initialization parameters:

To minimize the complexity of server side programming and reduce maintenance of web application .

Example 1:

web.xmlfile

```
<web-app>
    <servlet>
            <servlet-name>dummy1</servlet-name>
            <servlet-class>IniParamDetails</servlet-class>
            <init-param>
                      <param-name>email</param-name>
                       <param-value>ram@gmail.com</param-value>
            </init-param>

     </servlet>
          <servlet-mapping>
               <servlet-name>dummy1</servlet-name>
               <url-pattern>/iniparam</url-pattern>
```

```
           </servlet-mapping>
</web-app>
```

Working with Context parameters:

Context init parameters are specific to all or multiple servlets configured in the deployment descriptor file.

Example 2:

web.xmlfile

```
<web-app>
    <context-param>
                      <param-name>email</param-name>
                       <param-value>ram@gmail.com</param-value>
     </context-param>
    <servlet>
             <servlet-name>dummy1</servlet-name>
            <servlet-class>ContextParamDetails1</servlet-class>
    </servlet>
    <servlet-mapping>
           <servlet-name>dummy1</servlet-name>
           <url-pattern>/conparam1</url-pattern>
    </servlet-mapping>
   <servlet>
             <servlet-name>dummy2</servlet-name>
            <servlet-class>ContextParamDetails2</servlet-class>
    </servlet>
    <servlet-mapping>
           <servlet-name>dummy2</servlet-name>
           <url-pattern>/conparam2</url-pattern>
    </servlet-mapping>
</web-app>
```

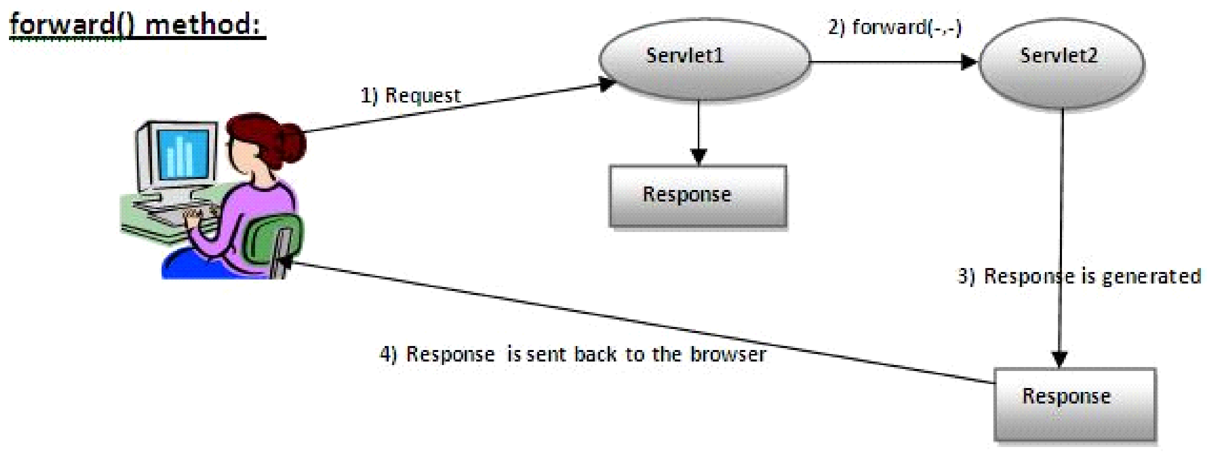
Describing RequestDispatcher Interface:

RequestDispatcher is an interface, implementation of which defines an object which can dispatch request to any resources(such as HTML, Image, JSP, Servlet) on the server.

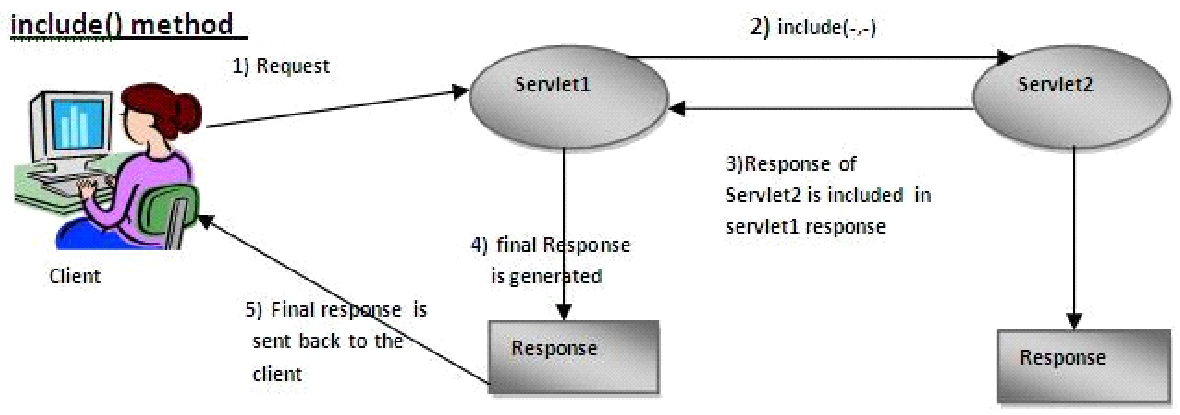
Methods of RequestDispatcher

RequestDispatcher interface provides two important methods

Methods	**Description**
voidforward(ServletRequest request, ServletResponse response)	forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server
voidinclude(ServletRequest request, ServletResponse response)	includes the content of a resource (servlet, JSP page, HTML file) in the response



As you see in the above figure, response of second servlet is sent to the client. Response of the first servlet is not displayed to the user.


include() method
1) Request
Servlet1
2) include(-,-)
Servlet2
3)Response of Servlet2 is included in servlet1 response
4) final Response is generated
Client
5) Final response is sent back to the client
Response
Response

A

s you can see in the above figure, response of second servlet is included in the response of the first servlet that is being sent to the client.