**RequestDispatcher**

RequestDispatcher is the interface, it is present in **javax.servlet** package. It is used to dispatch to the coming request into another page( like html , jsp , servlet ).

The RequestDispatcher object can be created by calling the getRequestDispatcher( ) method present in the ServletRequest.

It has two methods

forward(request, response)

include(request , response)

**forward** method is used to forward the coming request into another page ( like html, jsp , servlet)

**include** method is used to include the response of the other page ( like html, jsp , servlet).

Note:

When we include one servlet into another response of corresponding doXXX( ) method is included

If the corresponding doXXX( ) method is not found it  simply ignores the include

**sendRedirect**

sendRedirect( ) method is the method of HttpServletResponse interface. sendRedirect(path) is the method which is going to redirect the response to the other page (like html, jsp , servlet , other), but without passing the request and response objects. Hear the browser is made to work.

**Difference between forward and sendRedirect**

|  |  |
| --- | --- |
| **forward** | **sendRedirect** |
| The browser does not have any clue on what went in the behind the screen, because hear forwarding is done at the server side | It makes the browser to make the work. Such that it is going to happen at the client side |
| URL in browser changes | URL does not change |
| It is going to invoke corresponding doXXX( ) method at the destination, depending on http method at the request. | It invoke only doGet( ) method at the destination |
| It is faster | It is slower in operation |
| We can forward only to the internal resource | We can redirect the coming request to internal or any external resource |
| Eg : RequestDispatcher rd = request.getRequestDispatcher(“/url”);  **rd.forward( request , response );**  note: url can be any Html , JSP , servlet’s url, but only internal pages | Eg:  **response.sendRedirect(“/url”);**  note: url can be any Html , JSP , servlet’s url internal and also for external resource like  response.sendRedirect(“http://www.google.com”); |

**Note:** when we forward the request from one servlet to another servlet and if the corresponding overridden method does not present in the destination servlet than default implementation of the http servlet is invoked which in turn generates response with the error message.

**Attribute**

Attribute in servlets are the objects, it can be added , it can be retrieved if necessary it can also be removed. It can be used with the

request

context

session

methods setAttribute(key , value);    key:-String    value:- any object type

getAttribute(key); key:- String(name of attribute given while setting)

removeAttribute(key); key:-String(name of attribute given while setting)

**Context and Config**

**ServletConfig**

* It is the interface present in javax.servlet.\* package
* These objects are one for the servlet
* ServletConfig Object will be created during the initialization of servlet
* The ServletConfig Objects are public to the particular servlet they are not accessible to the other servlet
* ServletConfig Objects are available has long has the servlet is executing, but they get destroyed once the servlet execution is finished
* We can create the object of ServletConfig by calling the getServletConfig( ) method
* Inside the web.xml file we can declare the config object using the <init-param> tag, which should be declared inside <servlet> tag and after <servlet-class> tag
* Eg :     <servlet>

<servlet-name>Kar</servlet-name>

<servlet-class>com.vikas.Karnataka</servlet-class>

**<inti-param>**

**<param-name>CM</param-name>**

**<param-value>Yediyurapa</param-value>**

**<init-param>**

</servlet>

Creating the ServletConfig Object

ServletConfig   confi  =  getServletConfig( );

**getServletConfig( )** is the method of servlet interface, its implementation is given in the GenricServlet class. And it is inherited into HttpServlet class also.

Getting the data of the ServletConfig

We have two methods in the ServletConfig interface

**public Sting getInitParameter(“param-name”);**

**public  Enumaration  getInitParameterNames( );**

**ServletContext**

* ServletContext is a interface present in the javax.servlet.\*; package
* ServetContext Object is available for all the servlets in the application such that they are global to all the servlets.
* ServletContext objects are created at the time of web application deployment
* Has long has the web application is running the ServletContext objects will be available and it will be destroyed once the application is removed from the server
* In web.xml the we can declare the context object inside the <context-param> tag which should be under the <web-app> but not inside any <servlet> tags.
* Eg :

<context-param>

<param-name>PM</param-name>

<param-value>NarendraModi</param-value>

</context-param>

**Creating the ServletConfig Object**

**1st type**

ServletConfig   confi  =  getServletConfig( );

ServletContext  cont    =   confi.getServletContext( );

**2nd type**

ServletContext  cont    =   getServletContext( );

**3rd type**

ServletContext  cont    =   request.getServletContext( );

Getting the data of the ServletContext

We have two methods in the ServletContext  interface

**public Sting getInitParameter(“param-name”);**

**public  Enumaration  getInitParameterNames( );**

**Session Tracking**

Http protocol is a state less protocol, each request to the server is treated has the new request. So we need to make the server to remember the state of the user, so we make use of session tracking to maintain the state of an user. We can maintain it in 4 different ways

* URL rewriting
* Hidden tags
* Cookies
* HttpSession

**URL rewriting**

In URL rewriting the data or the required values are appended into the URL of the next servlet. We can send the data in the name and value pair

Eg :   url?name1=value1&name2=value2

/Welcome?name=vikas&age=24

When the user clicks the hyperlink the parameter which is stored in the URL is transferred into the server, in servlet we can get the values using getParameter( ) method to obtain a parameter value.

**Advantages**

It always work even if the cookie is disabled or not

Need not submit any additional form data

**Disadvantages**

It works only with the links ( hyperlink)

Only textual information is transferred

**Hidden form**

Hear we make use of hidden tag to store the data. We get this information using the servlet. But hear for each data we need to maintain one form tag.

pw.println(“<form action= “url”>”);

pw.println(“<input type= “hidden” name= “mail” value= “[vik@mail.com](mailto:vik@mail.com)”> ”);

pw.println(“<form>”);

**Advantage**

It works even if the cookie is disabled

**Disadvantage**

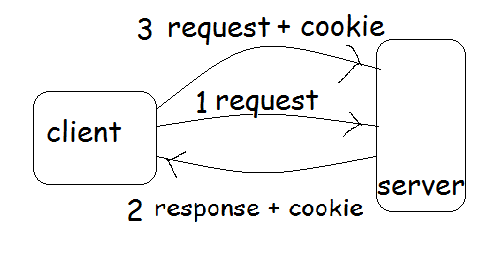
It is maintained at the server side

The form data is required

Only text data can be transformed

**Cookies**

Cookies are the small piece of information stored at the client side, when the request come from the client the servelt will create the Cookie and send it to the client along with the responce. When the next request goes to the server the request will also contain the cookies. So in the servlet we can get the cookie and get the information stored inside it.



Cookie is the class present in the **javax.servlet.http** package. This class contain lot of methods inside it. There is only one parameterised constructor inside this class. We can create the Object of the Cookie class by using this constructor.

**Cookie(String name , String value);**

There are few methods present in the class like

public void **setMaxAge**(int timeInSeconds)

public String **getName( )**

public String **getValue( )**

public void  **setValue(String value)**

public int **getMaxAge( )**

Some other methods to add the cookie and get the cookie are

public  void **addCookie(Cookie ck)**

public Cookie[ ] **getCookies( )**

**Eg:**

To create and add the cookie

**Cookie c1 = new Cookie(“name” , “Vikas”);**

**request.addCookie(c1);**

To get cookies

**Cookie c[ ] = response.getCookies( );**

**Advantages**

Easy and simple to maintain the state

They are stored at the Client side

**Disadvantages**

It of no use if the cookie is disabled at the browser

Only text information is stored at the cookies

**HttpSession**

This is the interface. Has mentioned above three ways are not much used. We make use of HttpSession interface to store the user details to make the server to remember that he his the person who was requested before. This interface provides lot of methods, to store , retrieve and to create the session.

Createing the session object can be done by calling

HttpSession session = request.getSession( );

HttpSession session = request.getSession( boolean );

If boolean expression  is true a new session is created. If we call empty get session method it returns the existing session object. If Boolean is false, it checks weather the session object is present if it is present it return same session object or else it returns NULL.

**Methods**

public void **setAttribute**(String name , Object value) this method is used to add the object into the created session object.

Public Object **getAttribute**(String name)  this method is used to get the Object stored in the session with the given name. If the object is present with that name it returns the object or else it returns the  NULL.

Public Enumaration **getAttributeNames**( ) this method is going to return the name of all the objects that are present in the session.

             Public void **removeAttribute**(String name) this method will remove the Object from the session which is present with the given name.

Public long **getCreationTime( )** used to get the time when the session was created.

Public String **getId**( ) each session have one session-id we can get that session id by calling the above method.

Public long **getLastAccessedTime( )** used to get the last accesses time. The time when the session was lastly accessed by the user.

Public void **invalidate( )** this method is used to kill the session.

JSP

JSP means the (java sever pages) is a technologies to develop the web pages. These pages are more easy to maintain than the servlet. Jsp pages are opposite to servlet, because hear the java code is present inside the html code. Jsp can do everything what the servlet can do. Because the jsp pages are converted into servlet by the web container.

Life cycle of JSP

The JSP pages are converted to servlet. The translation of JSP to servlet and loading and running is called has life cycle of JSP. It contains the following steps

* Translating the JSP page to servlet code
* Compile that servelt class into .class file
* Load the servlet class into memory
* Creating the object of servlet
* Initializing the instance members by calling the **jspInit( )** method
* Request processing by calling the **\_jspService ( )** method
* Destroying by calling the **jspDestroy( )** method

JSP has mainly 4 tags

Directive tag <%@ %>

Declaration tag <%! %>

Expression tag <%= %>

Scriptlet tag <% %>

JSP implicit objects

The jsp conatins few of the objects like out, request , response , session , config , application and ect