**ServletRequest Interface**

An object of ServletRequest is used to provide the client request information to a servlet such as content type, content length, parameter names and values, header informations, attributes etc.

There are many methods defined in the ServletRequest interface. Some of them are as follows:

**public String getParameter(String name):**

is used to obtain the value of a parameter by name

**public String[] getParameterValues(String name)**

returns an array of String containing all of given parameter name. It is mainly used to obtain values of a Multi select list boxl.

**java.util.Enumeration getParameterNames()**

returns an enumeration of all of the request parameter names.

**public int getContentLength()**

Returns the size of the request entity data, or -1 if not known.

**public String getCharacterEncoding()**

Returns the character set encoding for the input of this request.

**public String getCharacterEncoding()**

Returns the character set encoding for the input of this request.

**public String getContentType()**

Returns the Internet Media Type of the request entity data, or null if not known.

**public ServletInputStream getInputStream() throws IOException:**

Returns an input stream for reading binary data in the request body.

**public abstract String getServerName()**

Returns the host name of the server that received the request.

**public int getServerPort()**

Returns the port number on which this request was received.

### Example of ServletRequest to display the name of the user

In this example, we are displaying the name of the user in the servlet. For this purpose, we have used the getParameter method that returns the value for the given request parameter name.

**index.html**

1. <form action="welcome" method="get">
2. Enter your name<input type="text" name="name"><br>
3. <input type="submit" value="login">
4. </form>

**DemoServ.java**

1. **import** javax.servlet.http.\*;
2. **import** javax.servlet.\*;
3. **import** java.io.\*;
4. **public** **class** DemoServ **extends** HttpServlet{
5. **public** **void** doGet(HttpServletRequest req,HttpServletResponse res)
6. **throws** ServletException,IOException
7. {
8. res.setContentType("text/html");
9. PrintWriter pw=res.getWriter();
11. String name=req.getParameter("name");//will return value
12. pw.println("Welcome "+name);
14. pw.close();
15. }}

**Types of HttpRequest methods:**

Based on type of information requested by the browser HttpRequest methods are divided as follows

1. GET
2. POST
3. HEAD
4. OPTIONS
5. PUT
6. DELETE
7. TRACE

**GET**

1. We can use get request if we are expecting information from the server .
2. Usually for the get request read operation will be performed at server side . Hence status of application won't be changed in get request.
3. In GET request end-users provided information will be appended to the url as the part of Query string.
4. As the end-users information is visible in url. there may be a chance of security problems will raise. Hence sensitive data we can't send by using get request.
5. The length of the url is fixed . Hence we can send only limited amount of information by get request .
6. Only character data is allowed in url . Hence we can't send binary data by using get request.
7. As extra client provided information is available in the url , Bookmarking of url is possible .

##### 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>

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

<form action="/test">

----------

---------

</form>

**POST**

1. If we want to post huge amount of information to the server then we should go for POST.   
   Ex: uploading our resume in job portal.
2. Usually in post requests update operation will be performed. The state of operation will be changed.
3. In post request , client information will be encapsulated in the request body instead of appending to the url . Hence we can send sensitive data by using POST request.
4. There is no limit on size of request body . Hence we can send huge amount of information to the Server.
5. We can send binary data also in addition to text data.
6. Bookmarking of POST request is not possible.
7. POST requests are not Idempotent and not-safe to repeat.

###### **Triggers to send POST request :**

There is only one-way to send POST request that is to use the form with method attribute value is POST . 

<form action="/test" method="POST">

----------

---------

</form>

I.e., without having the form there is no chance of sending POST request.

|  |  |
| --- | --- |
| GET | POST |
| If we are expecting information from server , then we should go for GET . | If we want to post huge information to server , then we should go for POST . |
| Usually read operation will be performed. | Write & update operators will be performed . |
| Client data will be appended to url in the form of Query-String . | Client data will be encapsulated in request body |
| We can't send sensitive information . | We can send sensitive information . |
| We can send only limited information. | We can send Huge information. |
| We can send only text data. | We can send text & binary data. |
| Bookmarking is possible . | Bookmarking is not- possible . |
| GET requests are Idempotent & Safe. | POST requests are not-Idempotent & not-Safe. |
| Multiple ways to send GET request. | Only one way to send POST request. |

**PUT**

1. We can use PUT method for placing a resource at Server side where the location is specified by URL.
2. At the specified location , if already another resource present then the old resource is replaced with provided new resource .
3. By means of status code we can identify whether replacement is happend or not .
4. 200 means replacement happen , 201 means replacement not happen.
5. PUT method is Idempotent but not Safe .

DELETE:

* We can use this method for deleting a perticular resource from Server side. It is exactly counter part of PUT method.
* DELETE is Idempotent , but not-Safe.

**Note:** As the PUT & DELETE methods are not safe. Most of the web-servers won't allow these methods by default.  
To allow these methods at server side some configuration changes are required.