

Adv. Java means DURGA SIR..

ADV.JAVA

With

SCWCD / OCWCD

Servlets Material

1. Servlet Technology Model



DURGA M.Tech

(Sun certified & Realtime Expert)

Ex. IBM Employee

**Trained Lakhs of Students
for last 14 years across INDIA**

India's No.1 Software Training Institute

DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

Servlet Technology Model

Agenda:

1) Servlet API and life cycle

- javax.servlet Package :
 - Interfaces : (14)
 - Classes : (9)
 - Exceptions : (2)
- javax.servlet.Servlet interface :
- load-on-startup
- Life Cycle of the Servlet that implements Servlet interface :
- GenericServlet(AC):
- javax.servlet.http package :
 - Interfaces : (8)
 - Classes : (7)
- Structure of HttpRequest :
- Structure of HttpResponse :

2) Http Methods

- GET
- POST
- HEAD
- OPTIONS
- PUT
- DELETE
- TRACE
 - HttpServlet
 - Life Cycle of HttpRequest

3) HttpServletRequest

- To retrieve request parameters
- To retrieve request headers
- To retrieve request cookies
 - Retrieving client & Server information from the request

4) HttpServletResponse

- To set response headers
- To set ContentType of response
- To get Text Stream for response
- To get Binary stream for response
- To perform redirecting
- To add Cookies to the response
- Differences between send-Redirection and forward mechanism :

www.durgasoftonlinetraining.com



**Online Training
Pre Recorded Video
Classes Training
Corporate Training**

**Ph: +91-8885252627, 7207212427
+91-7207212428**

 **USA Ph : 4433326786**

E-mail : durgasoftonlinetraining@gmail.com

Introduction :

2-tier:

known and fixed number of clients we can access 2-tier.

Web Application: collection of web resources.

Web Resource: each web resource is capable of generate one web page.

(Ex: html , servlets, Jsp's , java script)

Web resources :

2 types

- static
- Dynamic

A web page whose content is fixed i.e., static web page

Ex: html , js , images

A web page whose content will be changed dynamically based on time of request generate input values of the request.

Ex:Servlets, JSP's , Server side java script(ssjs) , PHP

Based on the place client side web resources program executed.(in browser)

Based on the place server side web resources program executed but not resides .(in server side)

WEB APPLICATION REQUIREMENTS :

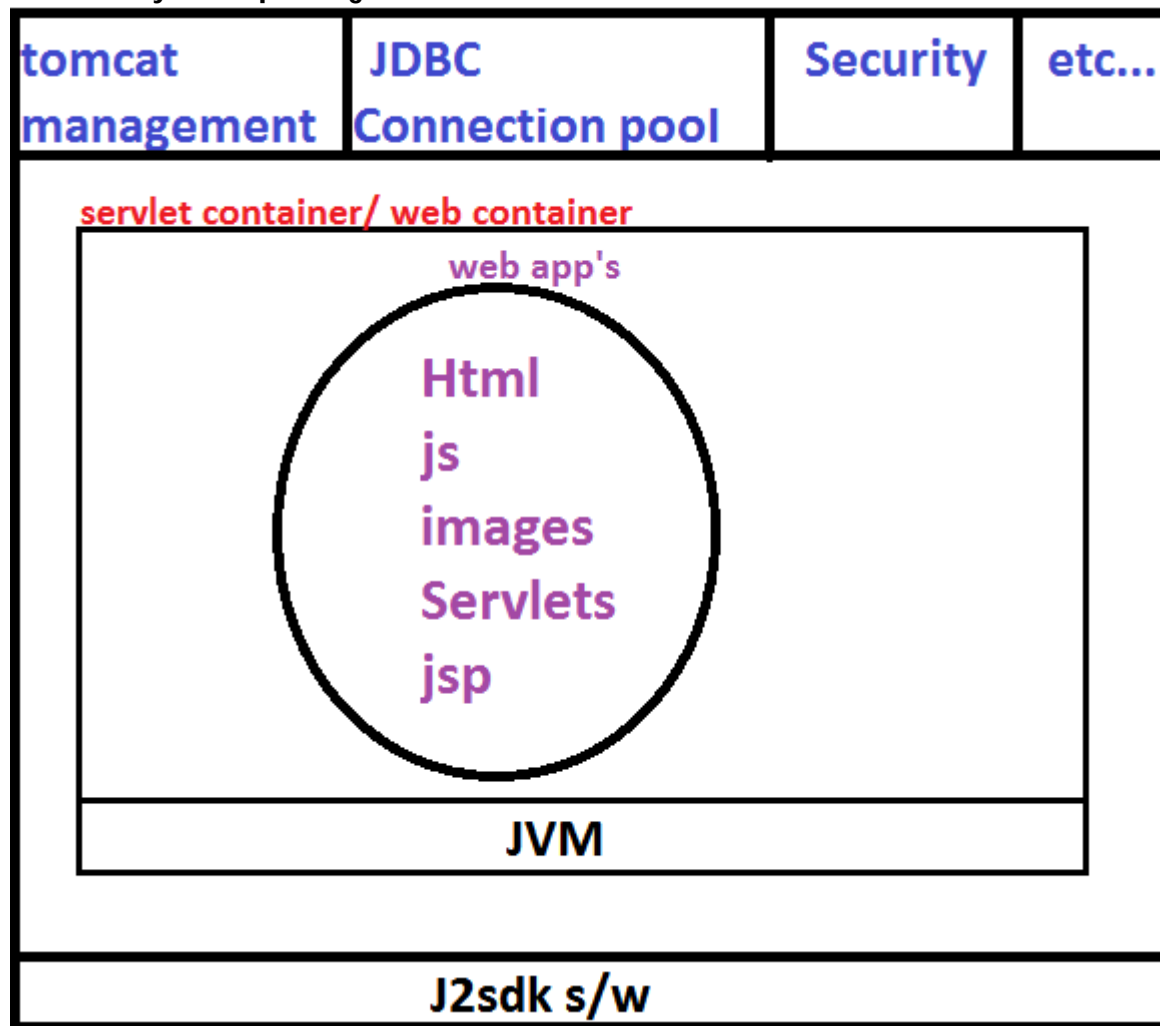
1. Browser software.
2. Technology to develop client side.
3. Technology to develop server side.
4. User software.
5. DataBase software.

WEB Server Software :

Web server software is special piece of s/w (or) special s/w which can manages the web applications and web resources we executes this web resources automatically or dynamically



when ever your requesting .



Responsibilities of web server :

1. It collects all client http requests.
2. It passes to appropriate web resources of web application.
3. Web server proceed built middle ware software.
4. container built in controller software.
5. Gives resultant out put.

Responsibilities of Servlet Container :

1. To provide the environment to manage web resource of Web Application.
2. It Perform total life cycle operations.
3. In case of JSP to generate corresponding Generated Servlet.

J2SDK software is instalable.

(through cmd prompt)

J2EE is not a instalable software , It is a specification,(not through cmd prompt)

Specification is a document , it contain set of RULES(interfaces) and GUIDELINES(classes).

SERVLET ia not a technology , It is a specification.

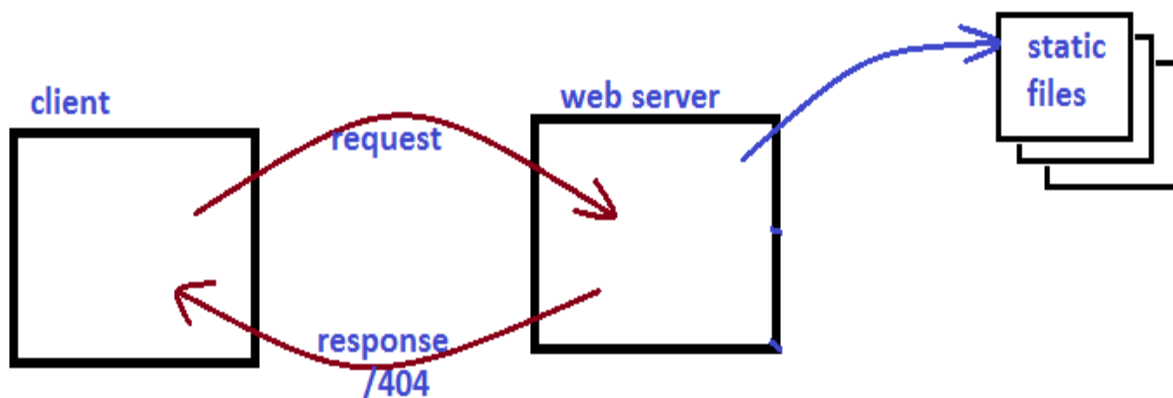
Specification related package consists of more number of INTERFACES and less number of CLASSES.

RULES ---> interfaces(method declarations)

GUIDELINES----> classes(concrete methods)

In servlets we use the instance variables are not Thread Safe.

Web Programming for Static information :



Client sends a request for a static file.

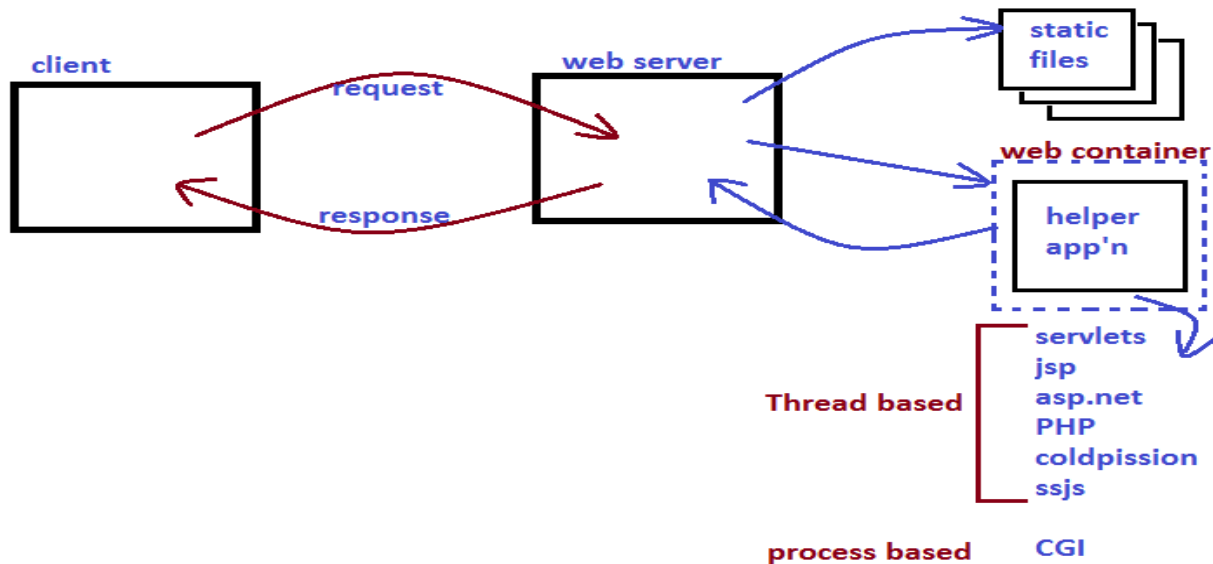
Web server searches whether the requested static file is available or not at Server side.

If the requested resource is available then it will return that static file as response.

If it is not available ,then web server sends 404 status code saying RequestResource is not available.

To serve static file, no processing is required at server side .Hence webserver always loves to serve static information.

Web Programming for Dynamic information :



Client sends a request to the Web server.

Web server checks whether the request is for static or Dynamic information.

If the request is for static information , web server searches for the required Static file.

If it is available it returns that file , other wise it returns 404 status code.

If the request is for Dynamic information web server forwards the request to some Helper Application.

Helper Application analyzes and process the request and generate required dynamic information.

Helper Application forwards that response to the web server and web server forwards that response to the client.

The following are various possible Helper Applications at Server side.

- 1.Servlet
- 2.Jsp
- 3.Asp.net
- 4.PHP
- 5.cold fusion
- 6.CGI
- 7.server side Java Script.

A servlet is a Server side web component managed by web container for generation of

dynamic information.

Web container is the best assistant to the programmer . It maintains entire life cycle of the servlet.

So that programmer has to concentrate only on Business logic. The remaining operations Instantiation of Servlet , Executing life cycle, Destroying the Servlet object and etc, taken care by Web container.

TYPES OF WEB CONTAINER:

There are 3 types of web containers are possible.

1. **Stand alone :**

Both web server and web container are available in a single integrated component , such type of web container are called Stand-alone web container.

Ex : Tomcat

This type of web containers are best-suitable for small scale applications and rarely used.

2. **In-Process Web container :**

If the web container runs in same address space(same machine) of web server and it is available as plug-in such type of web containers are called In-Process web container.

In this case both web server and web container need not be from the same vendor.

3. **Out-Process Web container:**

Web server and web container both are running on different machines , Web container is attached to the web server externally. Such type of web containers are called Out-process web containers.

we can configure front-end apache has to forward the request to the back-end weblogic server. These type of web containers are industry using.

This type of web Containers are most commonly used web containers.



LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...

JAVA MEANS DURGASOFT

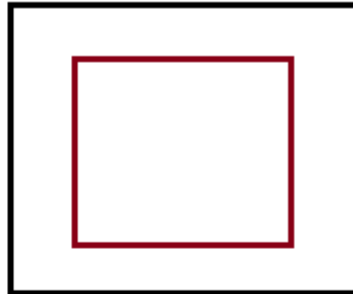
INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE

AN ISO 9001:2008 CERTIFIED
DURGA
SOFTWARE SOLUTIONS

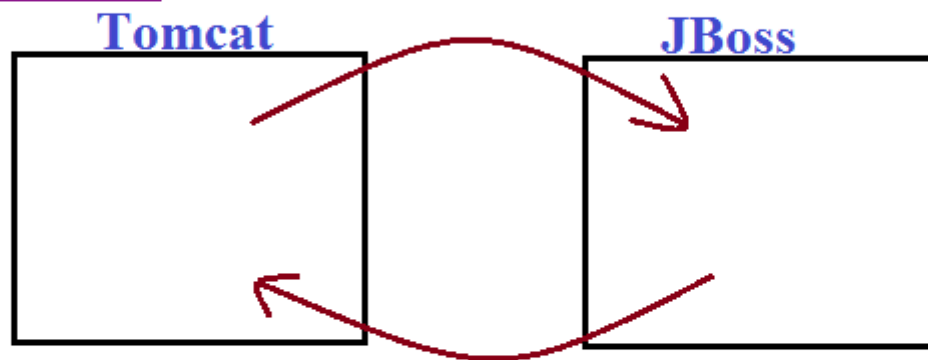
#202 2nd FLOOR
www.durgasoft.com

040-64512786
+91 9246212143
+91 8096969696

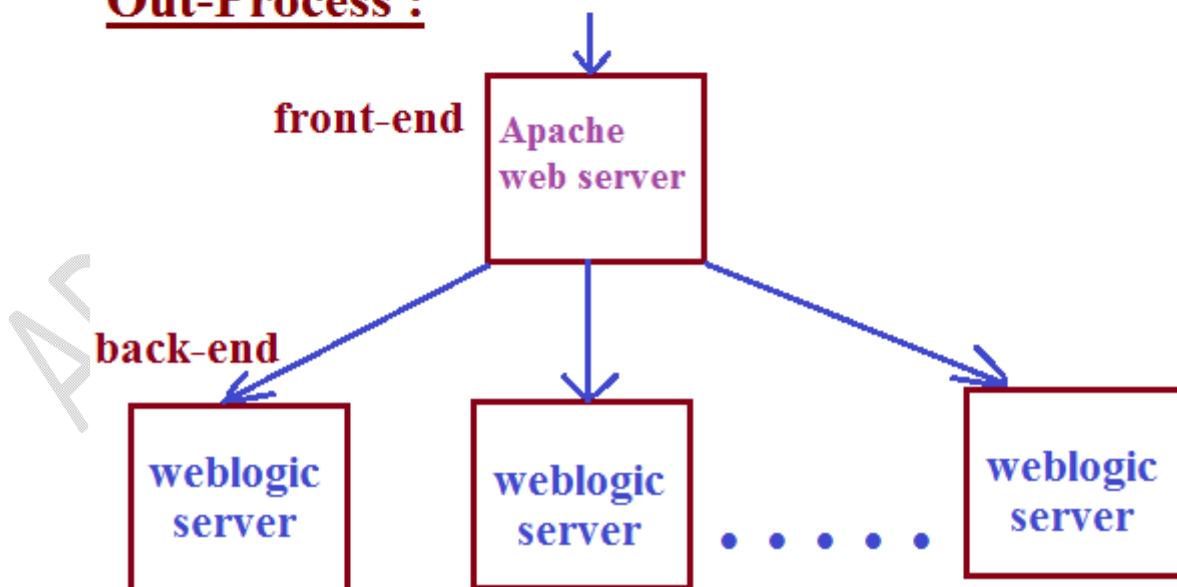
Stand alone :



In-Process :



Out-Process :



Differences between CGI and Servlets :

<u>CGI</u>	<u>SERVLETS</u>
It is Process based i.e, for every request a separate process will be created and it is responsible to generate required response.	It is thread-based i.e, for every request a new thread will be created and it is responsible to process the request.
Creation and Destruction of process for every request is costly. If the number of requests increase it will effects performance of the system.Hence CGI technology fail to the destroy of scalable.	Creation and Destruction of new thread for every request is not costly . Hence there is no effect on performance even though number of requests increases due to this it succeeds to delevery scalable.
Two process never share same address space(memory).Hence there is no chance of concurrency, inconsistency problems and synchronization is not required.	All threads share common address space hence there may be a chance of concurrency, inconsistency problems.
CGI programs can be written in multiple languages.But most commonly used language is PERL.	Servlets can be written only in Java.
Most of the CGI languages are not object oriented.Hence we are missing benefits of OOPs.	Java language itself is Object-Oriented . Hence we can get all benefits of OOPs.
CGI technology is platform dependent.	Servlet technology is platform independent.

FAST CGI :

It improves performance when compared with traditional CGI. In this case web container maintains a pool of process so that a single process can serve multiple requests one by one.

Servlet Technology Model :

1. servlet API and life cycle
2. Http methods
3. HttpServletRequest
4. HttpServletResponse

Servlet deffination : Servlet is a single instance multiple thread based server side Technology to develop Dynamic web resources of web application.

Servlet API :

We can develop Servlets by using the following 2 packages.

1. **javax.servlet**: This package defines several classes and interfaces to develop servlets from scratch irrespective of any protocol.
2. **javax.servlet.http** : It is the sub package of javax.servlet and contains several convenient classes and interfaces to develop http based servlets.

javax.servlet Package :

Interfaces of javax.servlet Package :(14)

- 1) **Servlet :**
Every servlet in java should implement servlet interface either directly or indirectly. ie, Servlet interface acts as a root interface for all java Servlets.
This interface defines the most common methods (including life cycle methods) which are applicable for any servlet object.
- 2) **ServletRequest :**
ServletRequest object can be used to hold client data.
ServletRequest interface defines several methods to acts as end users provided data from the request object.
Ex: getParameter()
- 3) **ServletResponse :**
ServletResponse object can be used to prepare and send responds to the client.
ServletResponse interface defines several methods which are required for preparation of

response.

Ex: `getWriter()`, `setContentType()`

4) **ServletConfig :**

For every servlet , web container creates one `ServletConfig` object to hold its configuration information like logical name of the Servlet , instantiation parameters etc.

`ServletConfig` interface defines several methods to access servlets configuration information.

Ex: `getServletName()`
`getInitParameter()`

5) **ServletContext :**

For every web application, web container will create one `ServletContext` object to hold application level configuration information like name of the app'n and `ServletContext` parameter etc.

Note : *ServletConfig is per Servlet , where as ServletContext is per web application.*

6) **RequestDispatcher :**

By using `RequestDispatcher` object we can dispatch the request from one component to another component.

This interface defines two methods.

1. `forward()`
2. `include()`

7) **SingleThreadModel :**

- Servlet technology is single instance multi threaded model. i.e, multi threads can operate simultaneously on the servlet object . Hence there may be a chance of data inconsistency problems.
- we can resolve these problems by using `SingleThreadModel` interface. If a servlet implements `SingleThreadModel` interface, then a single thread can access servlet object at a time i.e, Servlet can process only one request at a time. i.e, the main objective of `SingleThreadModel` interface is to provide thread-safety.
- But the problem with `SingleThreadModel` is Servlet can process only one request at a time which impacts performance of the system. Because of this problem `SingleThreadModel` interface is not recommended to use and it is deprecated in Servlet 1.3 version without introducing any replacement.
- we can provide thread-safety to the Servlet by using `synchronized` keyword.
- `SingleThreadModel` interface doesn't contain any methods . It is a marker interface.
- **Note:** In addition to above interfaces `javax.servlet` package defines the following interfaces also .

8) **Filter**

9) **FilterConfig**

10) **FilterChain**

----- to implements Filter Concept.

11) **ServletRequestListener**

12) **ServletRequestAttributeListener**

13) **ServletContextListener**

14) [ServletContextAttributeListener](#)

----- to implements Listener Concept.

Classes of javax.servlet package : (9)**1) [GenericServlet :](#)**

This class implement Servlet interface and provides default implementation for every method except service(). Hence it is an abstract class.

we can use this class as a base class to develop protocol independent servlets.

2) [ServletOutputStream :](#)

we can use this class object to send binary data(pdf files , image files , video/audio files etc) as response to the client.

3) [ServletInputStream:](#)

we can use ServletInputStream objects to read binary data send by the client.

Note:

In addition to above classes javax.servlet package defines the following classes also

4) [ServletRequestWrapper](#)**5) [ServletResponseWrapper](#)**

-----to implement wrapper concept.

6) [ServletRequestEvent](#)**7) [ServletRequestAttributeEvent](#)****8) [ServletContextEvent](#)****9) [ServletContextAttributeEvent](#)**

-----to define Events for Listeners.

All Events are classes and Listeners are interfaces

Exceptions of javax.servlet package :

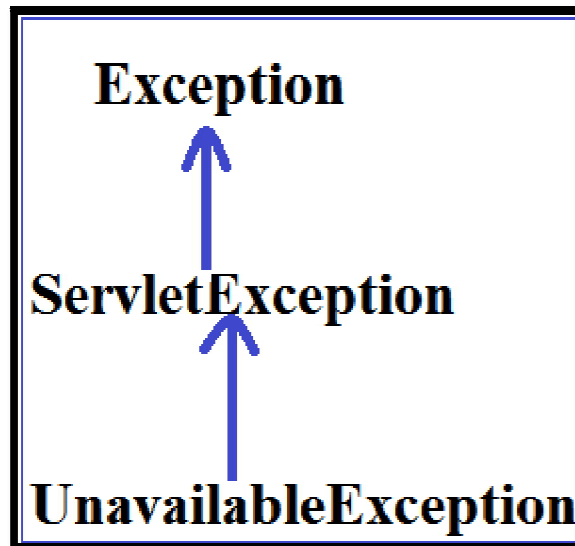
javax.servlet package defines the following 2 exceptions.

1. [ServletException.](#)

Servlet can throw this exception whenever it faces any difficulty while processing client request.

2. **UnavailableException** :

It is the child class of ServletException and it is deprecated .



Note :

In total `javax.servlet` package defines

14 - Interfaces

9--Classes

2--Exceptions

javax.servlet.Servlet interface :

Every servlet in java should compulsory implement Servlet interface either directly or indirectly i.e, Servlet interface acts as a root interface for all servlets.

This interface defines the most common methods which can be applicable for any Servlet object.

The following are the methods defined in Servlet interface.

1. `init()`
2. `service()`
3. `destroy()`
4. `getServletConfig()`
5. `getServletInfo()`

1. `init()`:


```
public void init(ServletConfig config) throws ServletException
```

This method will be executed only once by the web container immediately after Servlet instantiation to perform initialization activities .

Web container won't place Servlet object into service() method in the following cases :

1. If init() throws ServletException.
2. If init() doesn't return within the time period specified by web container.

In the above cases web container makes that servlet object eligible for Garbage Collection without calling any other life cycle method .

In this case web container creates a new Servlet object to provide service.

2. service() :

```
public void service(ServletRequest request , ServletResponse response )  
throws ServletException,IOException
```

web container calls this method for every request to provide response.
Entire servicing logic/business logic , we have to define in this method only.

3. destroy() :

```
public void destroy();
```

This method will be executed only once by the web container to perform clean up activities , when ever web container takes servlet object from out of service. This usually happens at the time of application undeployed (or) at the time of server shut down (or) Web Container requires some free memory .

when ever web container calls destroy() method , it may not be executed immediately. It will wait until completing all currently executing threads.

NOTE : `init()` , `service()` , `destroy()` are called life cycle methods of Servlet.
we can call `destroy()` explicitly from the `init()` and `service()` , in this case `destroy()` will be executed just like a normal method call and servlet object won't be destroyed .

4. `getServletConfig` :

```
public ServletConfig getServletConfig();
```

This method can be used to get `ServletConfig` object .
By using this config object Servlet can get its configuration information.

5. `getServletInfo ()` :

```
public void getServletInfo()
```

This method returns information about Servlet like Author, version , copy right information etc.

Demo program the develop a servlet by implementing Servlet interface

FirstServlet.java

```
1) package com.jobs4times;  
2) import java.io.IOException;  
3) import java.io.PrintWriter;  
4) import javax.servlet.Servlet;  
5) import javax.servlet.ServletConfig;  
6) import javax.servlet.ServletException;  
7) import javax.servlet.ServletRequest;  
8) import javax.servlet.ServletResponse;  
9)  
10) public class FirstServlet implements Servlet {  
11)  
12)     static {  
13)         System.out.println("Servlet class loading");  
14)     }  
15)     public FirstServlet () {
```

```
16) System.out.println("servlet instantiation");
17) }
18)
19) ServletConfig config;
20) public void init(ServletConfig config) throws ServletException {
21)     this.config=config;
22)     System.out.println(" we are in init() method ");
23) }
24) public void service(ServletRequest request, ServletResponse response)
25)     throws ServletException, IOException {
26)     System.out.println("We are in service() method ");
27)     PrintWriter out=response.getWriter();
28)     out.println("welcome to SCWCD");
29) }
30) public void destroy() {
31)     System.out.println("We are in destroy() method ");
32) }
33) public ServletConfig getServletConfig() {
34)     return null ;
35) }
36) public String getServletInfo() {
37)     return "Written by Jobs4Times " ;
38) }
39) }
```

web.xml

```
1) <web-app>
2) <servlet>
3) <servlet-name>first</servlet-name>
4) <servlet-class>com.jobs4times.FirstServlet</servlet-class>
5) <load-on-startup>5</load-on-startup>
6) </servlet>
7) <servlet-mapping>
8) <servlet-name>first</servlet-name>
9) <url-pattern>/fs</url-pattern>
10) </servlet-mapping>
11) </web-app>
```

<http://localhost:8080/SCWCD1A/fs>

1. When ever we are sending the request ,webserver checks whether this request is for static or dynamic information by using URL pattern.

2. If the request is for static information , Webserver searches for the required static file and provides required response.
3. If the request is for dynamic information , Webserver forwards that request to webcontainer .
The webcontainer identifies the corresponding servlet class by using web.xml
4. Webcontainer loads that .class file , perform instantiation , and execute init() and service() methods and provide required response to the webserver
5. webserver inturn forwards that response to end-user.

WithOut <load-on-startup> :

First Request:

1. loading .class file (/ servlet loading)
2. Instantiation
3. init()
4. service()

Second Request: service()

With <load-on-startup> :

At server startup/Application deployment

1. loading .class file (/ servlet loading)
2. Instantiation
3. init()

First Request : service()

Second Request : service()

Note:

The allowed values for <load-on-startup> tag is an integer i.e., +ve , zero , -ve

In the case of <load-on-startup> in deployment descriptor

- Highest value to give the least priority
- Lowest value give high Priority
- Zero will give last priority
- -ve value means ignore the <load-on-startup> concept.

If we give the two servlets having the same <load-on-startup> value we can't expect execution order or behaviour .

What is the advantage of <load-on-startup> :

We can equalize the response time of first request and remaining requests.

The main disadvantage of <load-on-startup> is it increases server start up time without any specific requirement don't configure this <load-on-startup> on servlet.

Note:

From servlet 2.5v onwards a single servlet can be mapped with multiple <url-pattern>tags i.e., we can take multiple <url-pattern> tags with in single <servlet-mapping> tag.

Ex:

```
1) <servlet-mapping>
2)   <url-pattern>/test </url-pattern>
3)   <url-pattern>/test </url-pattern>
4) </servlet-mapping>
```

Note :

Whenever we are writing 2 servlets having same url-pattern , if we are sending a request to particular url-pattern

The order of evaluation of web.xml is decided by the underling webserver , there is no specification rules

- In the case of Tomcat the order of evaluation of web.xml is Bottom to Top.
- In the case of Weblogic the order of evaluation of web.xml is Top to Bottom.

Life Cycle of the Servlet that implements Servlet interface :

1. Servlet class loading by class loader.(It is part of Jvm)
2. Servlet Instantiation by webcontainer , For this web container always calls public-no-argument constructor . Hence Every servlet should compulsory contain public-no-argument constructor.Otherwise we will get RuntimeException saying java.lang.Instantiation exception .
3. Execution of init(-) by webcontainer
Note: The above 3 steps will be performed Generally at the time of first request.If<load-on-startup> is configured these will be executed at the time of either server startup or at the time of application deployment.
4. Execution of service() by web-container.
5. Execution of destroy()

Note:If we are invoking destroy() explicitly inside service() that time any exception will be raised the exception will show to the end-user, when we won't handle this situation .

If web-container calls `destroy()` then that time the exception will be raised the exception that suppressed by the web-container.

constructor Vs `init()` :

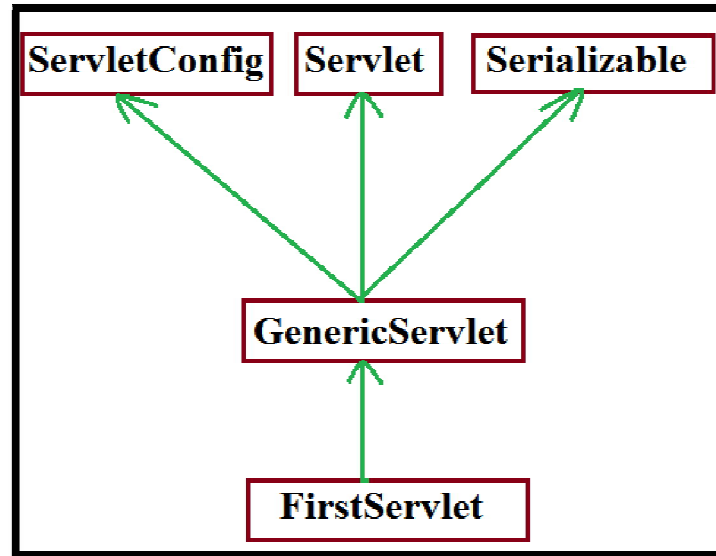
- In general , we can use constructor to perform initialization activities but in old versions of java, constructor cannot accept dynamically generated class name as argument.
- To perform initialization of servlet compulsory we should provide `ServletConfig` object as argument. whose class name is dynamically generated by web-container
- As constructor cannot accept these dynamically generated names, Sun people ignores constructor concept and introduced a specific method `init()` to perform initialization activities , which can take dynamically generated class names as the arguments.
- In the older versions , constructor cannot take any arguments but to perform initialization of a servlet compulsory we should provide `ServletConfig` as argument.
- Hence we can't use constructor to perform initialization activities , Sun people introduced a specific methods `init()` . for this , which can take `ServletConfig` as argument.

`destroy()` Vs `finalise()` :

- Before destroying any object Garbage collector always calls `finalise()` to perform cleanup activities. But we can't expect exact behaviour of Garbage collector. which is vendor dependent.
- Hence instead of depending on `finalise()` , Sun people introduced a specific method `destroy()` to perform cleanup activities , which should be executed always.

`GenericServlet(AC)`:

- We can develop servlet by implementing `Servlet` interface directly. In this approach compulsory we should provide implementing for all methods of `Servlet` interface whether it is required or not . This approach increase length of the code and reduces readability. We can resolve this problem by using `GenericServlet`.
- `GenericServlet` implements `Servlet` interface and provide default implementation for every method except `service()`. Hence it is an abstract class.
- We can develop servlets very easily by extending `GenericServlet`, instead of implementing `Servlet` interface directly. In this approach we have to provide implementation only for required methods instead of implementing all. It reduces length of the code and improves readability.
- `GenericServlet` is protocol independent servlet.
- `GenericServlet` implements `ServletConfig` and `Serializable` interfaces also.



Demo program to develop Servlet by extending GenericServlet :

```

1) public class FirstServlet extends GenericServlet {
2)     public void service(ServletRequest request, ServletResponse response)
3)         throws ServletException, IOException {
4)         PrintWriter pw=response.getWriter();
5)         pw.println("Developing Servlet by GenericServlet");
6)         System.out.println("This is Service() in First Servlet ");
7)     }
8) }
  
```

Internal implementation of GenericServlet :

```

1) public abstract class GenericServlet implements Servlet , ServletConfig , Serializable {
2)     private transient ServletConfig config ;
3)     public void init(ServletConfig config) throws ServletException {
4)         this.config=config ;    //webcontainer purpose
5)         init();
6)     }
7)     public void init() throws ServletException { //programmer purpose
8)     }
9)     public abstract service(ServletRequest request, ServletResponse response)
10)        throws ServletException, IOException ;
11)     public void destroy() {
12)     }
13)     public ServletConfig getServletConfig() {
14)         return config;
15)     }
  
```

```

16)    public String getServletInfo() {
17)        return "this is GenericServlet" ;
18)    }
19)    -----
20)    -----
21) }

```

GenericServlet contains 2 init() , we can override init() in our Servlet as follows .

1st way :

```

1)    public void init(ServletConfig config ) throws ServletException {
2)        // my own initialization
3)    }

```

This approach is not recommended , because we are not saving config object for the future purpose . Hence in our servlet any one calls getServletConfig(), this method returns null .

If we are calling getServletName() then we will get Runtime Exception saying java.lang.NullPointerException

2nd way :

```

1)    public void init(ServletConfig config ) throws ServletException {
2)        super.init(config);
3)        .....
4)    }

```

This approach is valid , but not recommended to override in Servlet , because internally 3 init () are executed , which impacts performance of the system .

3rd way :

```

1)    public void init() throws ServletException {
2)        // my own initialization activities
3)    }
4)

```

This way is highly recommended to define init() in our servlet.

LEARN FROM EXPERT & DIAMOND FACULTIES OF AMEERPET...

JAVA MEANS DURGASOFT

INDIA'S NO. 1 SOFTWARE TRAINING INSTITUTE

AN ISO 9001:2008 CERTIFIED

DURGA

SOFTWARE SOLUTIONS

#202 2nd FLOOR

www.durgasoft.com

040-64512786

+91 9246212143

+91 8096969696

How many init() present in GenericServlet and explain the need of it ?

2 init() methods are available in GenericServlet.

One is for web-containers purpose and the Second one is programmers purpose.

In how many ways we can override init() in our Servlet and which is the best way ?

3 ways 3rd one is the best way.

In GenericServlet why config variable declared as transient ?

1. Every Servlet in java is Serializable. At the time of Serialization of servlet object the corresponding config object is serialized automatically . Because it is part of object-graph of Servlet object.
2. If we are serializing config object , from that config object hacker may get corresponding context object reference. Once hacker got context object he can able to perform any operation on our application , which is never be recommended security wise. Hence at the time of serialization of servlet object , we can't Serialize config object. Due to this reason config variable declared as transient.

2. javax.servlet.http package :

This package defines more convenient classes and interfaces to define http based Servlets .
//protocol dependent

Interfaces of javax.servlet.http : (8)

- 1) **HttpServletRequest** :
It is the child interface of ServletRequest .
HttpServletRequest object can be used to hold client information.
- 2) **HttpServletResponse** :
It is the child interface of ServletResponse.
This object can be used to prepare and send response to the client.
- 3) **HttpSession** :
We can use HttpSession object to remember the client information across multiple requests. i.e., we can use session object for session management purpose.

- 4) **HttpSessionListener** :
To implement http based Listener
- 5) **HttpSessionAttributeListener** :
To implement http based Listener
- 6) **HttpSessionBindingListener** :
To implement http based Listener
- 7) **HttpSessionActivationListener** :
To implement http based Listener
- 8) **HttpSessionContext** :
deprecated and hence not recommended to use

Classes of javax.servlet.http package : (7)

- 1) **HttpServlet** :
It is the child class of GenericServlet . It can be used for developing http based Servlets .
- 2) **Cookie** :
We can use Cookie objects to implement Session management.
- 3) **HttpSessionEvent** :
To define Events for http based Listeners.
- 4) **HttpSessionBindingEvent** :
To define Events for http based Listeners.
- 5) **HttpServletRequestWrapper** :
To define http based Wrappers.
- 6) **HttpServletResponseWrapper** :
To define http based Wrappers.
- 7) **HttpUtils** :
Deprecated , not recommended to use .

Note:

In total javax.servlet.http package defines

8 -----> interfaces

7 -----> classes

Webserver and webclient communicate by using some common language ,which is nothing

but HTTP .

Http defines a standard structures for HttpRequest and HttpResponse .

Structure of HttpRequest :

Browser always sends the request in the following format :

Request Line
Request Headers
Request Body

Request Line :

GET	/login.jsp	HTTP/1.0
(Request method)	(Request URI)	(protocol version used by browser)

Request Headers :

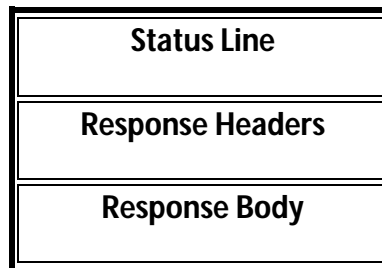
The request headers describes configuration information of the browser like

1. media types accepted by browser , language accepted by browser , encoding types supported by browser etc.
2. Web server use this information to send customized responses to the client .

Request Body :

1. It contains end-user provides information for the GET request it is optional where as for the POST request it is mandatory.
2. This is the structure of HttpRequest , If the browser sends request in this format , then only webserver understands the request . i.e, webserver understandable form.

Structure of HttpResponse :



Status Line :

200	OK	HTTP/1.1
(Status code)	(Description)	(protocol version used by server)

Status Codes :

1xx -----> Informational
 2xx----->Successful
 3xx----->Redirectionaal
 4xx----->client error
 5xx----->Server error

Response Headers :

These will provide configuration information of the server and information about the response [meta data] like content type of response , content length , last modified data etc., Browser will use these response headers to represent properly the response to the end-user.

Response Body :

It contains the original response provided by webserver.

If the server sends the response in the above form , then only browser understands the response i.e., it is browser understandable form.

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
8. CONNECT
9. MOVE
10. LOCK
11. PROFIND

1 - 3 methods introduced in http 1.0
4 - 11 methods introduced in http 1.1
1 - 7 are Big Http methods

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.

- 1) `<form action="/test" method="GET">`
- 2) -----
- 3) `</form>`

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

- 1) `<form action="/test">`
- 2) -----
- 3) -----
- 4) `</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 .

- 1) `<form action="/test" method="POST">`
- 2) `-----`
- 3) `-----`
- 4) `</form>`

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

Differences between GET & POST :

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.

HEAD :

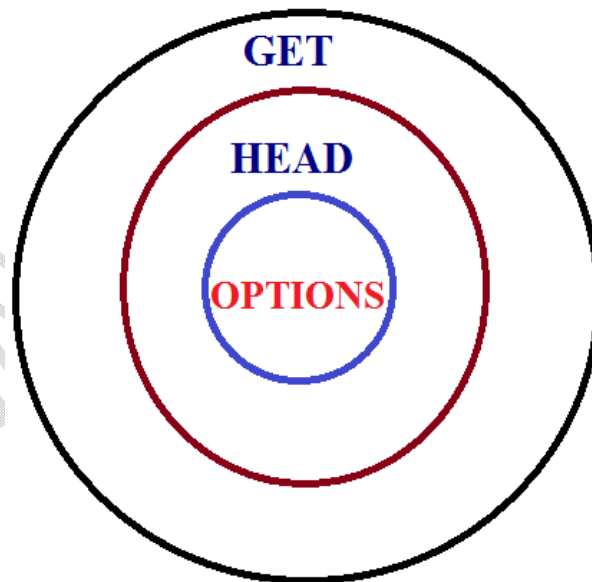
- We can use this method to get only response header information like content type , content length , last modified date etc., but no response body i.e., head request retrives always respone header but not response body.

- For HEAD request , internally doGet() will be executed. HEAD request is part of GET request.
- HEAD requests are Idempotent & Safe.

```
1) HEAD request -----> doHead {  
2)     doGet {  
3)         take only header part of GET response and  
4)         provide that as response to HEAD request.  
5)     }  
6) }
```

OPTIONS :

- This method is for getting supporting http methods to retrieve a particular resource from the Server side .
- OPTIONS method is Idempotent and Safe.
- **Note:** HEAD response is the part of GET response and OPTIONS is the part of HEAD response .



response from OPTIONS method request :

Http 1.1 | 200 | OK

Host : www.jobs4times.com

Server : Apache

Date : Fri , 08 Nov

Allow : OPTIONS , HEAD , GET , POST
Content length : 0

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 happen 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.

TRACE :

We can use this method for debugging purposes. If we want to know what request , server getting exactly as response, then we should go for TRACE method.

TRACE method is Idempotent and Safe.

Method	Is Idempotent ?	Is Safe ?
GET	Yes	Yes
POST	No	No
HEAD	YES	YES
OPTIONS	Yes	Yes
PUT	Yes	No
DELETE	Yes	No

TRACE	Yes	Yes
-------	-----	-----

Note : The only non-idempotent method is POST .

The following methods are not safe : POST , PUT , DELETE .

HttpServlet :

We can use HttpServlet class as a base class to develop Http based Servlets.

It is the child class of GenericServlet.

For every Http method XXX , HttpServlet class contains the corresponding doXxx() methods.

protected void doXxx() throws ServletException, IOException

Ex:

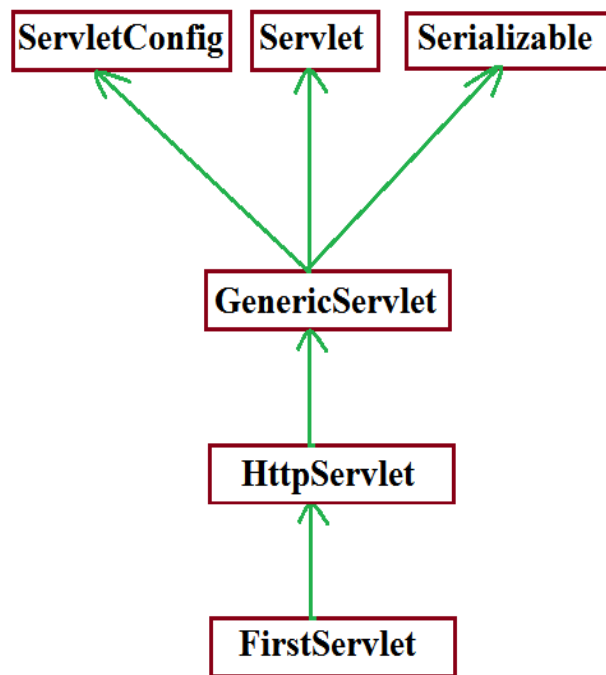
protected void doGet(HttpServletRequest request , HttpServletResponse response) throws ServletException, IOException

doGet(), doPost(), doHead(), doOption(), doPut(), doDelete(),doTrace() .

HttpServlet contains 2 service() methods.

1. **public void service(ServletRequest request , ServletResponse response) throws ServletException, IOException**
2. **protected void service(HttpServletRequest request , HttpServletResponse response) throws ServletException, IOException**





Ex: Demo program for HttpServlet.

login.html

We can use this to send post request to the server

```
1) <html>
2) <body><h1>This is HttpServlet Demo</h1>
3) <form action="/scwcd1c/test" method="post">
4)   Enter Name : <input type="text" name="uname" >
5)   <input type="submit">
6) </form>
7) </body>
8) </html>
```

FirstServlet.java

```
1) import javax.servlet.*;
2) import javax.servlet.http.*;
3) import java.io.*;
4) public class FirstServlet extends HttpServlet {
5)     public void doGet(HttpServletRequest request, HttpServletResponse response)
6)         throws ServletException, IOException {
7)         PrintWriter out=response.getWriter();
8)         String name=req.getParameter("uname");
9)         out.println("Hello"+name+"Good Morning , This is doGet method");
```

```

10)    }
11)    public void doPost(HttpServletRequest request, HttpServletResponse response)
12)        throws ServletException, IOException {
13)        PrintWriter out=response.getWriter();
14)        String name=req.getParameter("uname");
15)        out.println("Hello"+name+"Good Morning , This is doPost method");
16)    }
17) }

```

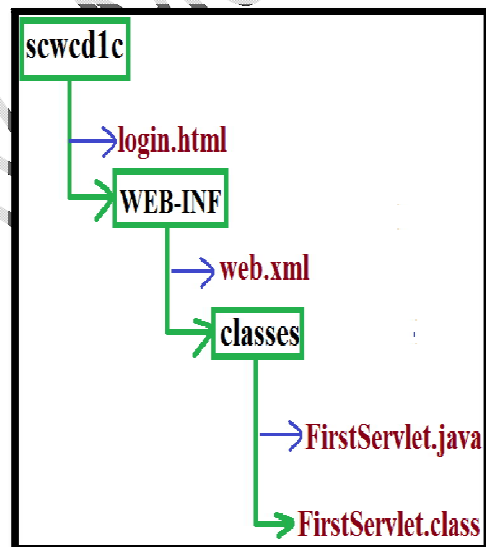
web.xml

```

1) <web-app>
2) <servlet>
3)   <servlet-name>first</servlet-name>
4)   <servlet-class>FirstServlet</servlet-class>
5) </servlet>
6) <servlet-mapping>
7)   <servlet-name>first</servlet-name>
8)   <url-pattern>/fs</url-pattern>
9) </servlet-mapping>
10) </web-app>

```

Deployment Structure :



Life Cycle of HttpRequest :

- 1) When ever we are submitting the form , browser prepares HttpRequest and send to the Server.

- 2) Webserver checks whether the request is for static or for dynamic information, by using URL.
- 3) If the request is for static information , webserver provide required response.
- 4) If the request is for dynamic information , then webserver forwards that request to web-container.
- 5) Webcontainer identifies the corresponding matched servlet by using web.xml
- 6) Webcontainer checks whether the corresponding Servlet object is already available or not . If it is not already available , then webcontainer loads the corresponding .class file and perform instantiation by executing public no argument constructor .
- 7) After instantiation , immediately webcontainer creates the ServletConfig object and invoke init() by passing that config object as argument.
- 8) Webcontainer creates ServletRequest and ServletResponse object and by passing those as arguments , It will invoke public service().
- 9) With in that service() , request and response objects will be type casted into HttpServletRequest and HttpServletResponse and invoke protected service() by passing those as arguments.

```

1) public void service(ServletRequest req , ServletResponse res )
2)     throws ServletException, IOException {
3)     HttpServletRequest req=(HttpServletRequest)req ;
4)     HttpServletResponse res=(HttpServletResponse)res ;
5)     service(req1 , res1);
6) }

```

- 10) With in the protected service() we will identify HttpRequest method and corresponding doXxx() will be invoked. If it is not a valid Http method then it is method will return 501 status code response saying invalid Http method .

```

1) protected void service(HttpServletRequest request,HttpServletResponse response)
2)     throws ServletException, IOException{
3)     String method=request.getMethod() ;
4)     if(method.equals("GET")) {
5)         doGet(req , res) ;
6)     }
7)     else if(method.equals("POST")) {
8)         doPost(req , res) ;
9)     }
10)    else if(method.equals("HEAD")) {
11)        doHead(req , res) ;
12)    }
13)    .....
14)    .....
15)    else {
16)        prepare response with 501 Status code saying Invalid Http Method.
17)    }

```

18) }

11) If our Servlet class contain required doXxx() , then it will be executed . Otherwise parent class doXxx() will be executed.

NOTE :

1. For every web application, web container creates one ServletContext object at the time of application deployment.
2. For every Servlet , web container creates one ServletConfig object just before calling init().
3. For every request web container creates one ServletRequest and one ServletResponse objects just before calling service() . Once service() completes both request and response objects will be eligible for Garbage Collector(GC).

Case 1 : If we are writting public service() in our Servlet , then for any type of request [GET or POST] only service() will be executed.

Case 2 : If our Servlet contains both public service() and protected service() , then for every request only public service() will be executed.

Case 3 : If our Servlet contains both service() and doGet() , then for any request including GET , only service() will be executed. And there is no chance of executing doGet().

Note: It is never recommended to place service() in our Servlet. i.e., defining service() in our Servlet is stupid kind of activity.

Case 4 : If we are sending get request , but our Servlet doesn't contain doGet() , then HttpServlet doGet() will be executed which provides response with 405 status code saying Http method GET is not Supported by this url.

Case 5 : If we are sending post request , but our servlet doesn't contain doPost() , then HttpServlet doPost() will be executed , which provide 405 status code saying Http method POST is not Supported by this url.

Case 6 : HttpServlet class doesn't contain any abstract method , still it is declared as abstract class what is the reason ?

For majority methods in HttpServlet , proper implementation is not available and these are just to send error information .

Hence if we are creating HttpServlet object directly and calling these methods , we will get just error information , with that we can't do anything.

To prevent instantiation of HttpServlet , Sun people declared HttpServlet class as abstract .

Case 7: To provide same response for both GET and POST request , we have to implement doGet() and doPost() as follows .

Ex:

```
1) public class FirstServlet extends HttpServlet {  
2)     public void doGet(HttpServletRequest request,HttpServletResponse response)  
3)         throws ServletException, IOException {  
4)         // implemented required information  
5)     }  
6)     public void doPost(HttpServletRequest request,HttpServletResponse response)  
7)         throws ServletException, IOException {  
8)         doGet() ;  
9)     }  
10) }
```

Case 8 : In our Servlet , It is not recommended to override the following methods . Because these methods are properly implemented inside HttpServlet.

1. service(ServletRequest request, ServletResponse response)
2. service(HttpServletRequest request, HttpServletResponse response)
3. doHead()
4. doOptions()
5. doTrace()

Case 9 : In our Servlet , it is highly recommended to override the following methods , because these methods doesn't have proper implementation in HttpServlet.

6. doGet()
7. doPost()
8. doPut()
9. doDelete()

Case 10 : To provide response to HEAD request , we have to override either doHead() or doGet() , otherwise we will get 405 status code saying Http method GET is not supported by this url.

HttpServletRequest :(I)

Using HttpServletRequest , write code to

1. Retrieve Form parameters
 1. getParameter()
 2. getParameterValues()

3. `getParameterNames()`
4. `getParameterMap()`
2. Retrieve request Headers
 1. `getHeader()`
 2. `getHeaders()`
 3. `getHeaderNames()`
 4. `getIntHeader()`
 5. `getDateHeader()`
3. Retrieve Cookies
 1. `getCookies()`

Retrieve Form parameters :

- Form parameters are key-value pairs , where both key and values are String objects only.
- A parameter can be associated with either a single value or with multiple values.
- ServletRequest interface defines the following methods to retrieve Form parameters at server side .
 1. `getParameter()`
 2. `getParameterValues()`
 3. `getParameterNames()`
 4. `getParameterMap()`

1. `getParameter()` :

```
public String getParameter(String pname)
```

- It returns the value associated with specified parameter.
- If the parameter associated with multiple values then this method simply returns only first value.
- If the specified parameter not available then we will get null argument is case-sensitive .

```
String user=request.getParameter("uname");
```

2. `getParameterValues()`

```
public String[ ] getParameterValues(String pname)
```

- It returns all values associated with specified parameter.

- If the parameters not available , we will get null .
- Argument is Case-Sensitive.
- Ex:

```
1) String[] course=request.getParameterValues("course");
2) for(String s:course) {
3)     out.println(s);
4) }
```

3. getParameterNames()

```
public Enumeration getParameterNames()
```

- Returns all Form parameters names.
- If the request is not associated with any form parameters. Then we will get empty enumeration object but not null.

Ex:

```
1) Enumeration e= request.getParameterNames() ;
2) while(e.hasMoreElements() {
3)     String pname=(String)e.nextElement() ;
4)     String pvalue=request.getParameter(pname);
5)     out.println(pname+" "+pvalue);
6) }
```

4. getParameterMap()

```
public Map getParameterMap()
```

- Returns the map object containing parameter names as keys and parameter values as map values.
- Keys are strings and values are string[] .
- If the request doesn't associated with any parameters , this method returns empty map object , but not null.

key(String)	value(String[])
subject	{SCJP , SCWCD }
user	{ arun }
.....
-----	-----

Ex:

```
1) Map m = request.getParameterMap();
2) for(Object o : m) {
3)     Map.Entry m1=(Map.Entry) o;
4)     String pname=(String)m1.getKey();
5)     String[] pvalue=(String[])m1.getValue();
6)     out.print(pname+".....");
7)
8)     for( String s1 : pvalue ) {
9)         out.println(s1 + " , ");
10)    }
11)    out.println(" ");
12) }
13) output :
14) subject .....SCJP , SCWCD
15) user.....arun
```

Example :

```
1) Map m=req.getParameterMap();
2) Set keySet=m.entrySet();
3) Iterator itr=keySet.iterator();
4) while(itr.hasNext()) {
5)     Map.Entry entry=(Map.Entry)itr.next();
6)     String pname=(String)entry.getKey();
7)     out.println(pname);
8)     String[] pvalues=(String[])entry.getValue();
9)     out.println(pvalues);
```

Example :

```
1) public void doPost(..) {
2)     response.setContentType();
3)     PrintWriter pw=res.getWriter();
4)     out.println("");
5)     out.println("name :"+ request.getParameter("uname"));
6)     .....
7)     .....
8) }
```

Retrieving Request headers :

- For every request browser sends its configuration information in the form of request headers. These may include the media types accepted by browser , encoding types supported by browser the type of browser etc.
- Server uses these request headers to send customized responses to the client.

The following are some of the important request headers.

- 1) Accept :
media types accepted by browser.[like txt/html , pdf , ppt , jar]
- 2) Accept-Encoding:
Encoding types supported by browser.
- 3) User-agent:
It represents the type of the browser.
- 4) Content-length:
It represents the length of request body.
- 5) Cookie :
Used to send cookies for Session management.

Utilities of Request headers :

1. By using Accept-Encoding request header at Server side we can identify whether browser supports compression or not.
2. If browser supports compressed form , we can send response in compressed form. So that we can save download time and bandwidth.
3. By using User-agent request header at server side we can identify the type of browser which sends the request. Based on that we can send browser compatible customized responses.
4. By using Cookie request header we can send Cookie to the server so that we can achieve session management.

HttpServletRequest defines the following methods to retrieving header information at server side

1. getHeader()

```
public String getHeader(String hname)
```

- It returns the value associated with specified request header.
- If the header associated with multiple values then this method returns only 1st value .
- If the specified request header is not available then we will get null.
- Argument is not-case sensitive .

Ex:

```
String hvalue=request.getHeader("accept");
```

output :

```
accept -- text/html , application/xhtml+xml , application/xml ;
```

2. getHeaders()

```
public Enumeration getHeaders(String hname)
```

- Returns all values associated with specified header .
- If the specified header not available , then this method returns empty Enumeration object, but not null.
- Argument is not-case sensitive .

3. getHeaderNames()

```
public Enumeration getHeaderNames()
```

Returns all request header names associated with that request object.

HttpServletRequest defines the following more convenient methods to retrieve int and date values(Headers) directly .

4. getIntHeader()

```
public int getIntHeader(String hname);
```

Ex:

```
String length=request.getIntHeader("content-length");  
int l=Integer.parseInt(length);  
(OR)  
int l=request.getIntHeader("content-length");
```

If the specified request header is not associated with int value , then this method returns NumberFormatException.

```
req.getIntHeader("user-agent");//RE:java.lang.NumberFormatException
```

If the specified request header is not available in `getIntHeader()` , this method returns "-1" but not null.

5. `getDateHeader()`

```
public long getDateHeader(String hname);
```

- Returns the date value associated with specified header as no. of milliseconds Since Jan 1st 1970.
- If we pass these milliseconds as arguments to Date constructor. We will get exact date
Ex:

```
long l=request.getDateHeader("date");  
Date d=new Date(l);
```

- If the specified header is not associated with the date , then we will get RuntimeException saying IllegalArgumentException Exception.
- If the specified request header is not available in `getDateHeader()` , this method returns "-1" but not null.

Write a Servlet to print all request-headers associated with the request.

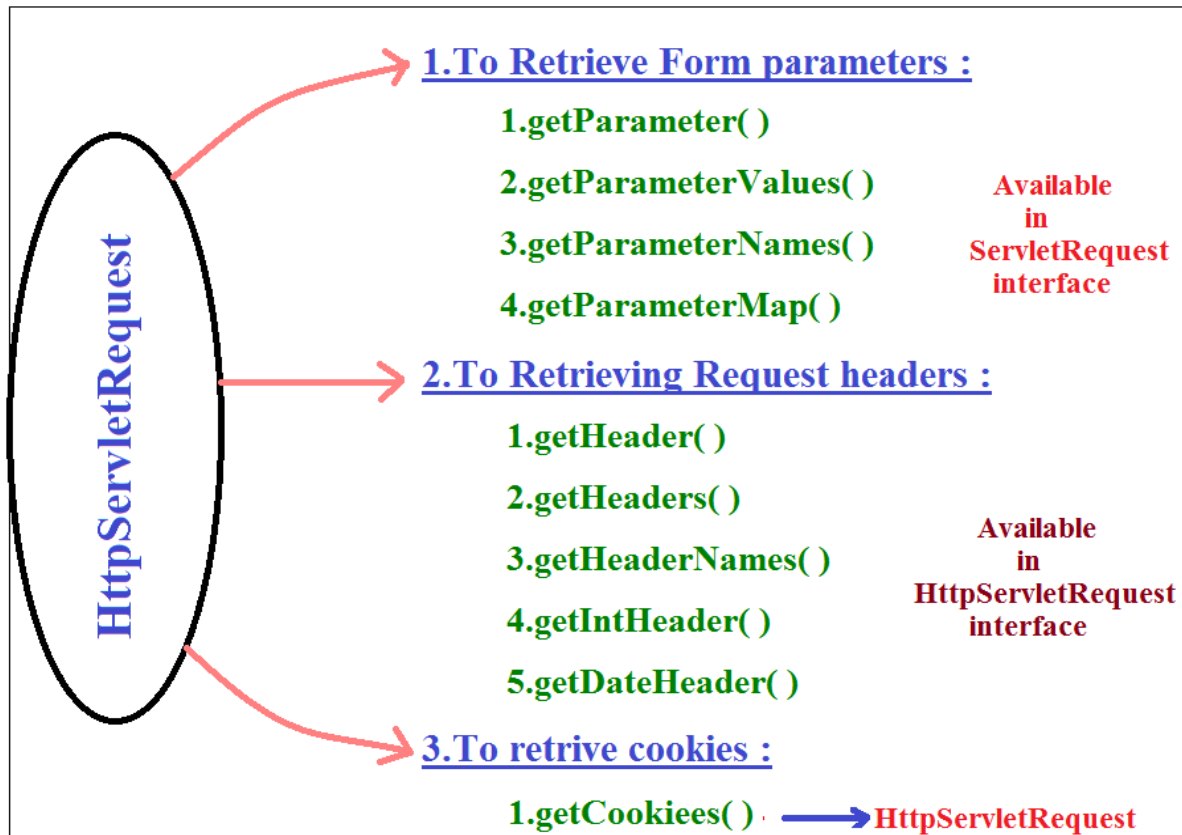
EX:

```
1) public class ReqHeader extends HttpServlet {  
2) public void doGet(HttpServletRequest request,HttpServletResponse response)  
3)     throws ServletException, IOException {  
4)     PrintWriter out = response.getWriter() ;  
5)     Enumeration e = request.getHeaderNames() ;  
6)     while(e.hasMoreElements()) {  
7)         String hname=(String)e.nextElement();  
8)         String hvalue=request.getHeader(hname);  
9)         out.println(hname+"-----"+hvalue);  
10)    }  
11) }  
12) }
```

Retriving Cookies from the request :

HttpServletRequest interface defines the following methods to retrieve cookies from the request object .

```
Cookie[] c = request.getCookies() ;
```



Retrieving client & Server information from the request :

ServletRequest interface defines the following methods to retrieve client and server information from request.

```
public String getRemoteHost()
```

Returns fully qualified name of the client which sends the request.

```
public String getRemoteAddr()
```

Returns IP-address of the client .

```
public String getRemotePort()
```

Returns the port no. on which the client is running .

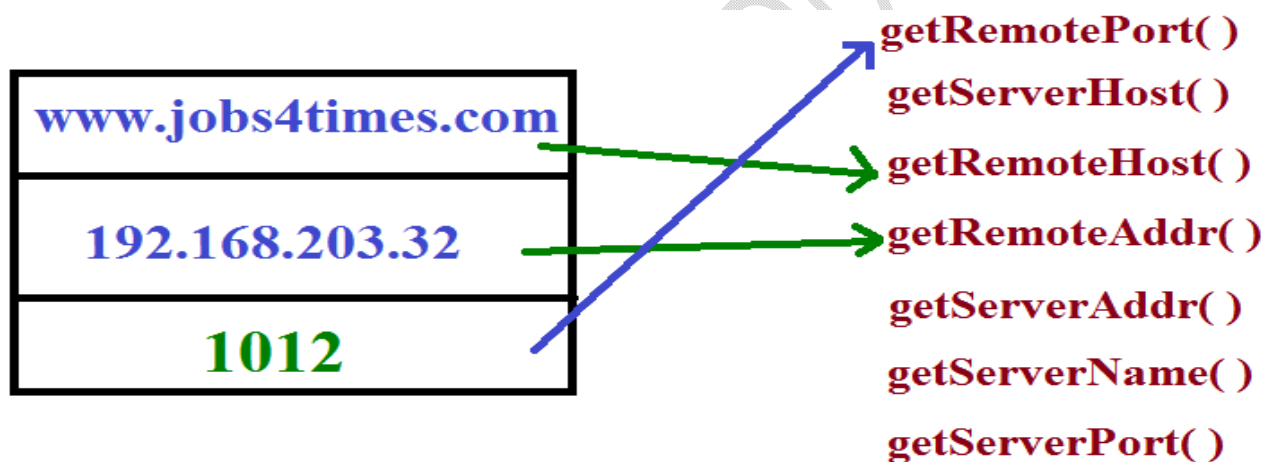
```
public String getServerName()
```

Returns the name of the Server to which the request has been sent .

```
public String getServerPort()
```

Returns the port no. on which server is running .

A request triggered from www.jobs4times.com with an IP-address 192.168.203.32 running on port 1012. Place the appropriate ServletRequest methods to their corresponding values.



HttpServletResponse :

By using HttpServletResponse interface , write code

1. To set Response Headers .
2. To set content type of Response .
3. To acquire text stream for response.
4. To acquire binary stream for response .
5. To redirect the request to another URL .
6. To add cookies to response.

Setting HttpServletResponse Headers :

- Http response header describes the configuration information of the server and information about the response like content-type , content-length , last modified data etc.,
- Browser using these response headers to display response body properly to the end user.

HttpServletResponse defines the following methods to add headers to the response .

```
public void addHeader(String hname , String hvalue )
```

- If the specified header name is already available , to the existing values , this new value will also be added.
- Replacement will not Occur .

```
public void setHeader(String hname , String hvalue )
```

- If the specified header is already available . Then the old value will be replaced with new value .

Some times headers associated with int & date values .

1. `public void addIntHeader(String hname , int hvalue)`
2. `public void setIntHeader(String hname , int value)`
3. `public void addDateHeader(String hname , long ms)`
4. `public void setDateHeader(String hname , long ms)`

Note:

If the header associated with multiple values . Then we have to use `addHeader()`

If the header associated with only one value , then we should go for `setHeader()`

Set Content type of Response :

Content type header represents MIME type (multipurpose internet mail extension) of the response.

Common MIME Types :

1. `text/html` -----> Html text as response
2. `text/xml` -----> xml document as response
3. `image/jpeg` -----> JPEG as response
4. `image/png` -----> png as response

5. application/ms-excel -----> excel sheet as response
6. application/ms-word -----> word document as response
7. application/pdf -----> PDF file as response
8. application/jar -----> jar file as response

We can set MIME type by using the following 2 ways

By ServletResponse interface :

It contains the following method to set the response.

```
public void setContentType(String mimeType)
```

Ex:

```
response.setContentType("application/PDF")
```

By HttpServletResponse interface :

It contains setHeader() to set content type.

```
public void setHeader(String hname , String hvalue)
```

Ex:

```
response.setHeader("content type" , "text/html");
```

Note: In general response.setContentType() is recommended to use.

To acquire text stream for the response :

1. We can send text data as the response by using PrintWriter object .
2. We can get PrintWriter object by using getWriter() of ServletResponse .

```
public PrintWriter getWriter() throws IOException
```

Ex:

```
PrintWriter out = response.getWriter() ;
```

To acquire Binary stream as response :

1. We can send Binary information(video , image files) by using ServletOutputStream object.
2. By using getOutputStream() of ServletResponse we can get this object .

```
public ServletOutputStream getOutputStream() throws IOException
```

Ex:

```
ServletOutputStream sos=response.getOutputStream () ;
```

Demo program to send image file as response to the client

```

1) import javax.servlet.*;
2) import javax.servlet.http.*;
3)
4) public class BinaryStream extends HttpServlet {
5)     public void doGet(HttpServletRequest request, HttpServletResponse response)
6)         throws ServletException, IOException{
7)         response.setContentType("image/jpeg");
8)         ServletOutputStream sos = response.getOutputStream() ;
9)
10)        //File f= new File("c:\\tomcat\\sunset.jpeg");
11)
12)        String path=getServletContext().getRealPath("sunset.jpeg");
13)        File f= new File(path);
14)        FileInputStream fis=new FileInputStream(f);
15)
16)        byte[] b=new byte[(int)f.length()];
17)        //f.length() returns long so we have to convert into int.
18)        //[] --> accepts int only
19)
20)
21)        fis.read(b);    //reading the image & place into byte[]
22)        sos.write(b);    //write byte[] to response
23)        sos.flush();
24)        sos.close();
25)    }
26) }

```

To send XML data as a response :

```

1) response.setHeader("content-Type", "text/xml");
2) PrintWriter out=response.getWriter();
3) out.print("<?xml version='1.0'>");
4) out.print("<greeting language='en_US'>");
5) out.print("hello world");
6) out.print("</greeting>");

```

Note :

At any point of time we can get either PrintWriter object (or) ServletOutputSteram object but not both , simultaneously.

Otherwise we will get RuntimeException saying IllegalStateException

Because from the Servlet we can send only one response at a time.

```

1) public void doGet() {
2)     PrintWriter out=response.getWriter();
3)     ServletOutputStream sos =response.getOutputStream() ;
4)     -----

```

```
5) -----  
6) }
```

Output: RuntimeException saying **IllegalStateException** getWriter() has already been called for this response.

IIQ: By using which of the following stream we can send both binary & text data as response from the Servlet ?

1. PrintWriter()
2. ServletOutputStream ()
3. ServletInputStream ()
4. None of the above ()



FREE TRAINING VIDEOS

You Tube **3000+ VIDEOS**

www.youtube.com/durgasoftware



www.durgajobs.com

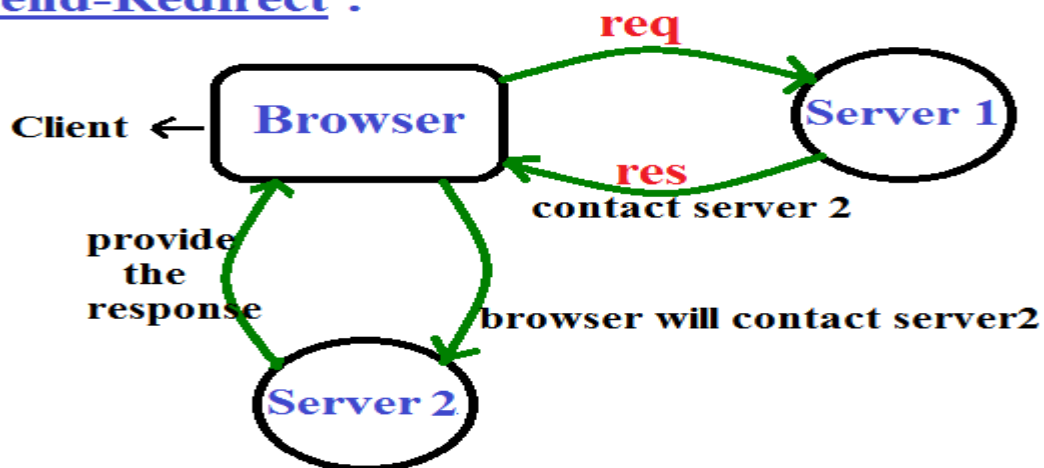
Continuous Job Updates for every hour

Fresher Jobs **Govt Jobs** **Bank Jobs**

Walk-ins **Placement Papers** **IT Jobs**

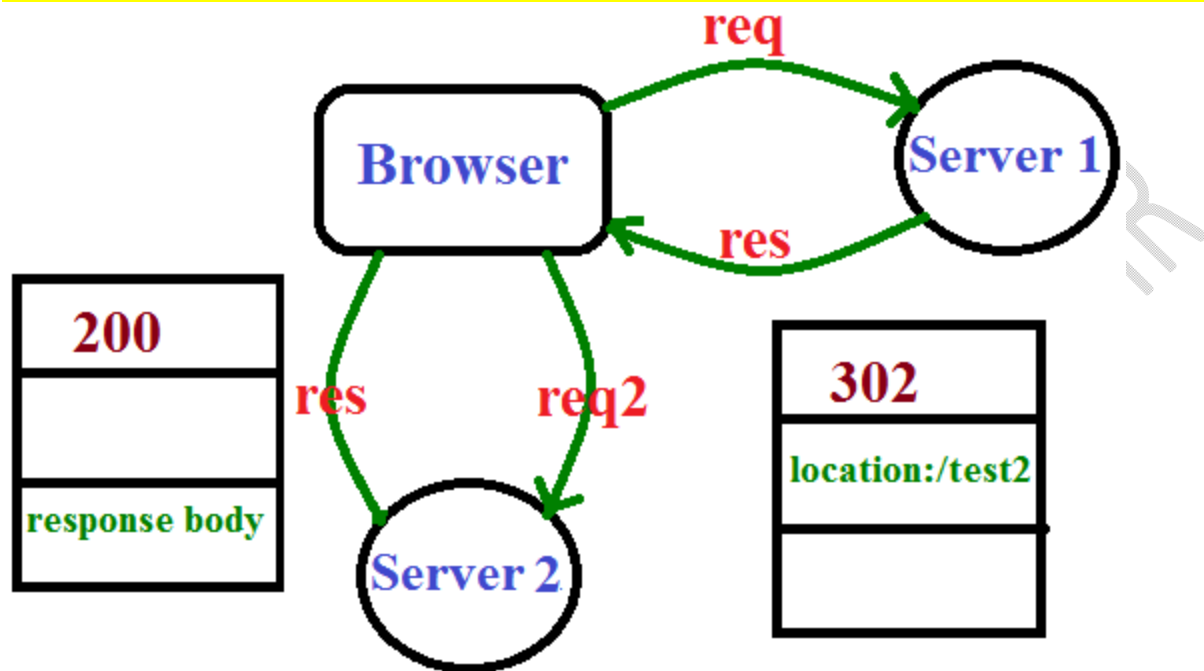
Interview Experiences

Complete Job information across India

Re-directing HttpRequest to another url :Forward :Include:send-Redirect :

Forward & include should work when both servlets should be in same server .
 Some times we have to re-direct the request to another url we can achieve this by using
 sendRedirect() of HttpServletResponse .

```
public void sendRedirect(String targetpath) throws IOException
```

Process of send Redirection :

**Here operation takes place at browser
(i.e., client-side)**

1. Browser sends request to Servlet1
2. If servlet1 is not responsible to provide response , then it will update browser to contact servlet2. This can be informed by setting status code 302 & location header with servlet2.
3. Browser by seeing status code 302 creates a new request object & sends to Servlet2.
4. Servlet2 provides desired response to the browser & the browser inturn display that response to the end user .

Demo program for send Re-direct :

```

1) public class FirstServlet extends HttpServlet {
2)     public void doGet(HttpServletRequest request,HttpServletResponse response)
3)         throws ServletException, IOException {
4)         response.sendRedirect("test2");
5)         (or)
6)         response.setStatus("302");
7)         response.setHeader("location" , "scwcd/test2");
8)     }
9) }
  
```

```

1) public class SecondServlet extends HttpServlet {
2)     public void doGet(HttpServletRequest request,HttpServletResponse response)
3)         throws ServletException, IOException {
4)         PrintWriter out=response.getWriter();
5)         out.println("2nd Servlet ");
6)     }
7) }

```

- When ever we are sending request to FirstServlet , SecondServlet will provide the response through re-direction
- After committing the response we are not allow to perform re-direction. Otherwise we will get RuntimeException saying IllegalStateException(Tomcat people doesn't provide support for this).

```

1) public void doGet(HttpServletRequest request,HttpServletResponse response)
2)     throws ServletException, IOException {
3)     -----
4)     -----
5)     out.flush(); //committing the response
6)     response.sendRedirect("/test2"); //RE:IllegalStateException
7) }

```

www.durgasoftonlinetraining.com



Online Training

Pre Recorded Video

Classes Training

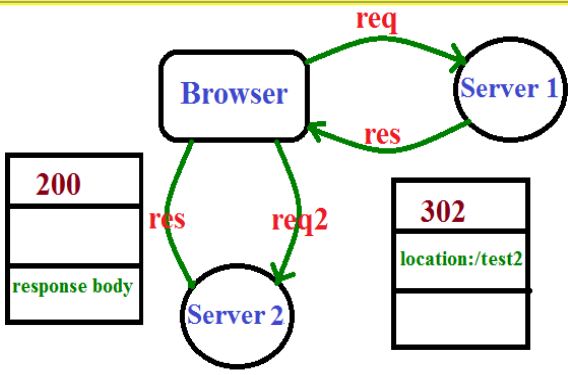
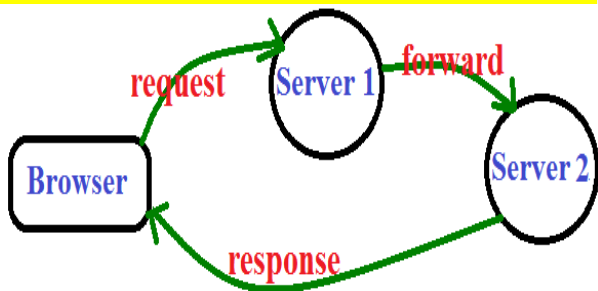
Corporate Training

Ph: +91-8885252627, 7207212427
+91-7207212428
USA Ph : 4433326786

E-mail : durgasoftonlinetraining@gmail.com

Applicable for forward also.

Differences between send-Redirection and forward mechanism :

send Re-direct	forward
 <p>Here operation takes place at browser (i.e., client-side)</p>	
This mechanism will work at client side. Hence client aware of which Servlet is providing the required response.	This mechanism will work at server side and it is not visible to the client. Hence client is not aware of which Servlet is providing the required response.
In this approach an extra trip is required to the client side . Hence network traffic increases & effects the performance of the system.	No extra trip is required to the client & hence there are no network traffic & performance problem.
This is only possible mechanism to communicate with resoueces which are present outside of web-container.	forward mechanism will work with in the web-container only and we can't use this mechanism to communicate with the resources present outside of container.
Seperate new request object will be created in send Redirection .	Same request object will be forwarded .
By using HttpServletResponse object . we can achieve send redirection. <code>response.sendRedirect("/test2");</code>	By using Request Dispatcher object we can achieve forward mechanism . <code>rd.forward(req , res);</code>

After committing the Response we are not allow to perform send Redirection .Otherwise we will get RuntimeException saying `IllegalStateException`.

After Committing Response , we can't perform forward otherwise `IllegalStateException`.

Adding Cookies to the response :

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

```
public void addCookie(Cookie c)
```

Ex:

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

NOTE :

1. We can use redirection to communicate either with in the same server or outside the server. But recommended to use communicate outside.
2. forward mechanism is applicable to communicate only with in the same server . But outside of server we can't use.
3. To communicate with in the same server we can use either forward or send redirection but recommended to use forward.





LEARN FROM EXPERTS ...

COMPLETE JAVA

CORE JAVA, ADV. JAVA, ORACLE, STRUTS, HIBERNATE, SPRING, WEB SERVICES,...

COMPLETE .NET

C#.NET, ASP.NET, SQL SERVER, MVC 5 & WCF

TESTING TOOLS

MANUAL + SELENIUM

ORACLE D2K

MSBI SHARE POINT

HADOOP ANDROID

C, C++, DS, UNIX

CRT & APTITUDE TRAINING

AN ISO 9001:2008 CERTIFIED

DURGA

Software Solutions®

202, 2nd Floor, HUDA Maitrivanam,
Ameerpet, Hyd. Ph: 040-64512786,

9246212143, 8096969696

www.durgasoft.com