Java Servlets

Applet V/S Servlet

BASIS FOR COMPARISON	APPLET	SERVLET
Execution	Applet is always executed on the client side.	Servlet is always executed on the server side.
Packages	import java.applet.*; import java.awt.*;	import javax.servlet.*; import java.servlet.http.*;
Lifecycle methods	init(), stop(), paint(), start(), destroy().	init( ), service( ), and destroy( ).
User interface	Applets use user interface classes like AWT and Swing.	No User interface required.
Requirement	Requires java compatible browser for execution.	It processes the input from client side and generates the response in terms of the HTML page, Javascript, Applets.
Bandwidth Utilization	Applets utilize more network bandwidth as it executes on the client machine.	Servlets are executed on the servers and hence require less bandwidth.
Security	More prone to risk as it is on the client machine.	It is under the server security.

What is Servlets?

- Basically, a java program that runs on the server
- Creates dynamic web pages
- Web browsers communicate with web server using **H**yper **T**ext **T**ransfer **P**rotocol When you click a link on a web page, submit a form, or run a search, an **HTTP request** is sent from your browser to the target server.

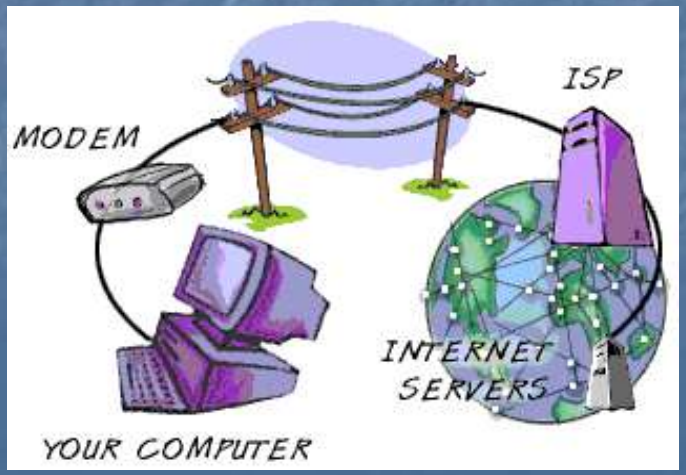
- Servlets are the Java programs that runs on the Java-enabled web server or application server. They are used to handle the request obtained from the web server, process the request, produce the response, then send response back to the web-server.

Properties of Servlets :

- Servlets work on the server-side.
- Servlets capable of handling complex request obtained from web server.

What Do They Do?

- Handle data/requests sent by users (clients)
- Create and format results
- Send results back to user

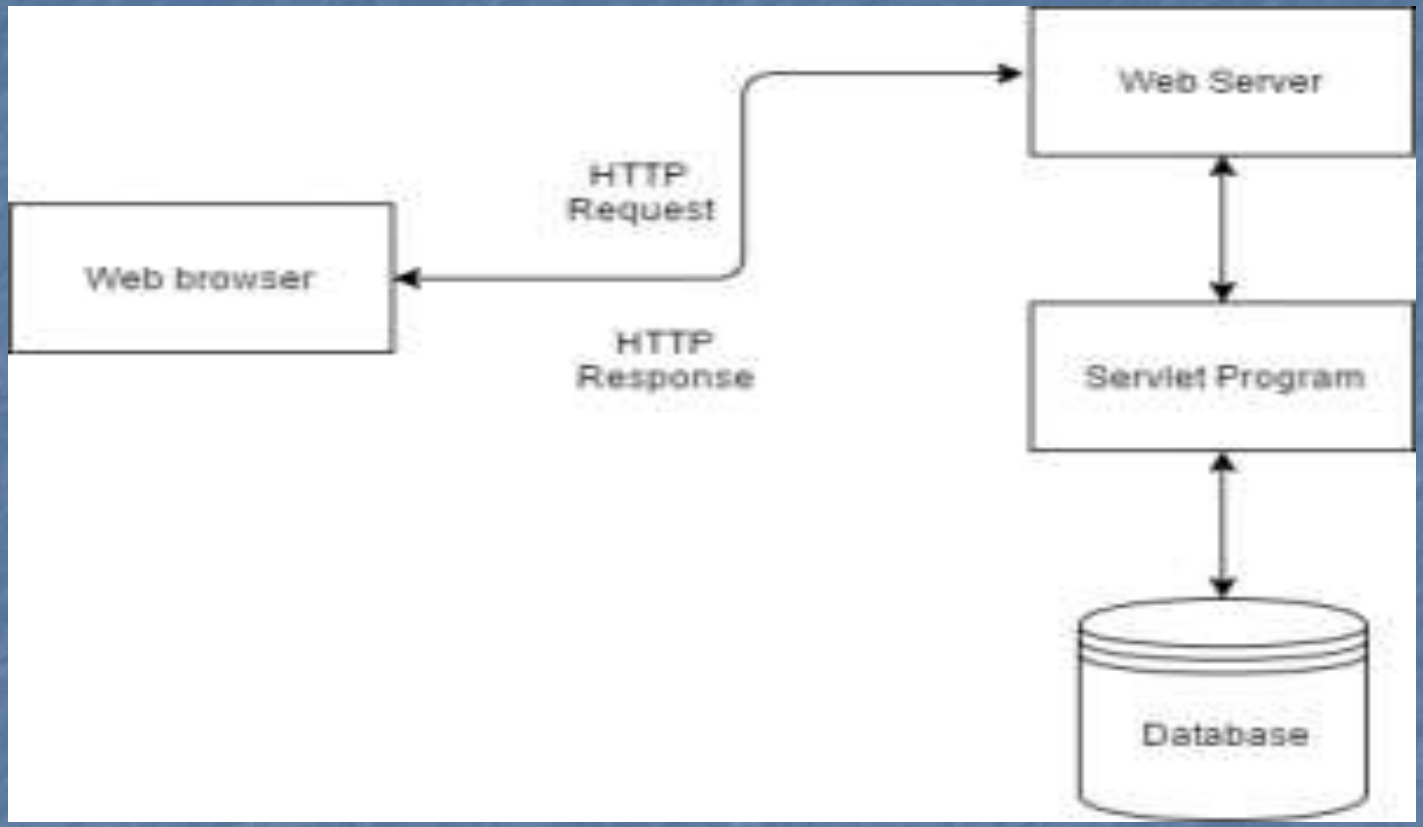


- **Execution of Servlets :**
  Execution of Servlets involves the six basic steps:

1. The clients send the request to the web server.
2. The web server receives the request.
3. The web server passes the request to the corresponding servlet.
4. The servlet processes the request and generate the response in the form of output.
5. The servlet send the response back to the web server.
6. The web server sends the response back to the client and the client browser displays it on the screen.

Servlet Architecture:

The following diagram shows the servlet architecture:



Need For Server-Side Extensions

- The **server-side extensions** are nothing but the technologies that are used to create dynamic Web pages. Actually, to provide the facility of dynamic Web pages, Web pages need a container or Web server. To meet this requirement, independent Web server providers offer some proprietary solutions in the form of **APIs**(Application Programming Interface). These **APIs** allow us to build programs that can run with a Web server. In this case , **Java Servlet** is also one of the component APIs of **Java Platform Enterprise Edition** which sets standards for creating dynamic Web applications in Java.

Who Uses Servlets?

- Servlets are useful in many business oriented websites

- … and MANY others

- **HOW WEB SERVER WORK:**
- Consider a request for a static web page.
- A user enters a URL to a browser.
- The browser generates an HTTP request to the appropriate web server.
- The web server map this request to a specific file .
- That file is returned in an HTTP response to the browser.
- The HTTP header in the response indicates the type of the content.
- The MIME are used for this purpose.
- Example: ordinary ASCII text has a MIME type of text/plain.
- HTMLL source code of a web page has a MIME type of text/html.

History

- A web server could dynamically construct a page by creating a separate process to handle each client request. The process would open connection to one or more database in order to obtain the necessary information. It communicate with the web server via an interface known as the CGI. CGI allowed separate process to read data from the HTTP request and write data to the HTTP response. C, C++ & perl Different language were used to build CGI program.
- CGI suffered serious problem like creating a separate process for each client request were expensive, in term of processor & memory. It was also expensive to open & close database connection.
- CGI programs were not platform independent.

Servlets vs. CGI

- CGI Advantages
  - CGI scripts can be written in any language
  - Does not depend on servlet-enabled serve

SERVLET	CGI(COMMON GATEWAY INTERFACE)
Servlets are portable and efficient.	CGI is not portable
In Servlets, sharing of data is possible.	In CGI, sharing of data is not possible.
Servlets can directly communicate with the web server.	CGI cannot directly communicate with the web server.
Servlets are less expensive than CGI.	CGI are more expensive than Servlets.
Servlets can handle the cookies.	CGI cannot handle the cookies.

Static vs Dynamic website

Static Website	Dynamic Website
Prebuilt content is same every time the page is loaded.	Content is generated quickly and changes regularly.
It uses the **HTML** code for developing a website.	It uses the server side languages such as **PHP,SERVLET, JSP, and ASP.NET** etc. for developing a website.
It sends exactly the same response for every request.	It may generate different HTML for each of the request.
The content is only changed when someone publishes and updates the file (sends it to the web server).	The page contains "server-side" code which allows the server to generate the unique content when the page is loaded.
Flexibility is the main advantage of static website.	Content Management System (CMS) is the main advantage of dynamic website.

Servlets

- Servlet Advantages
  - Efficient
    - Single lightweight java thread handles multiple requests
    - Optimizations such as computation caching and keeping connections to databases open
  - Convenient
    - Many programmers today already know java
  - Powerful
    - Can talk directly to the web server
    - Share data with other servlets
    - Maintain data from request to request
  - Portable/platform independent
    - Java is supported by every major web browser
  - Inexpensive
    - Adding servlet support to a server is cheap or free
  - secure

Java Server Web Development Kit

- JSWDK
  - Small, stand-alone server for testing servlets and JSP pages
- The J2EE SDK
  - Includes Java Servlets 2.4

Servlet capable server

Apache

- Popular, open-source server

- Tomcat
  - A “servlet container” used with Apache

Other servers are available

Servlet Code

- Written in standard Java
- Every servlet must Implement the javax.servlet.Servlet interface and extending one of the two classes:
- Javax.servlet.GenericServlet
- Javax.servlet.http.HttpServlet
- First class is protocol independent servlet and second one is http servlet.

GenericServlet:

if you are creating a Generic Servlet then you must extend javax.servlet.GenericServlet class.

GenericServlet class has an abstract service() method. Which means the subclass of GenericServlet should always override the service() method.

Signature of service() method:

```
public abstract void service(ServletRequest request, ServletResponse response) throws ServletException, java.io.IOException
```

The service() method accepts two arguments ServletRequest object and ServletResponse object. The request object tells the servlet about the request made by client while the response object is used to return a response back to the client.

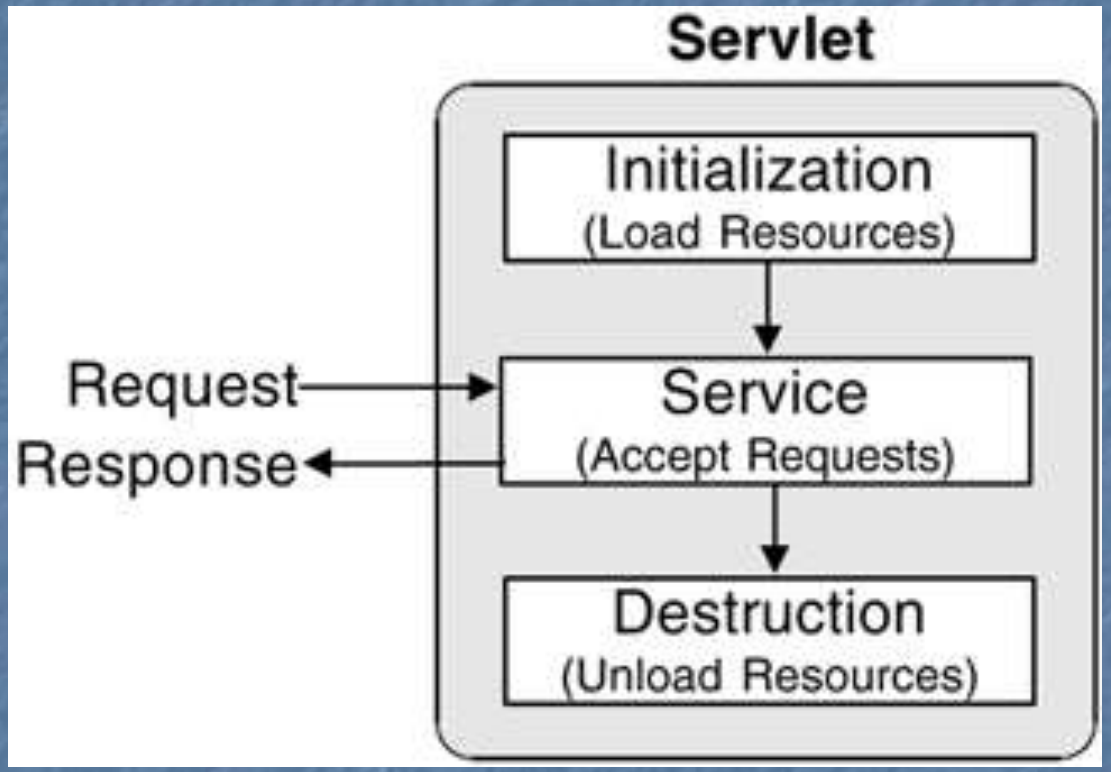
HTTP Servlet:

If you creating Http Servlet you must extend javax.servlet.http.HttpServlet class, which is an abstract class. Unlike Generic Servlet, the HTTP Servlet doesn't override the service() method. Instead it overrides one or more of the following methods. It must override at least one method from the list below:

- **doGet()** – This method is called by servlet service method to handle the HTTP GET request from client. The Get method is used for getting information from the server.
- **doPost()** – Used for posting information to the Server.
- **doPut()** – This method is similar to doPost method but unlike doPost method where we send information to the server, this method sends file to the server, this is similar to the FTP operation from client to server.
- **doDelete()** – allows a client to delete a document, webpage or information from the server.
- **init() and destroy()** – Used for managing resources that are held for the life of the servlet.
- **getServletInfo()** – Returns information about the servlet, such as author, version, and copyright.

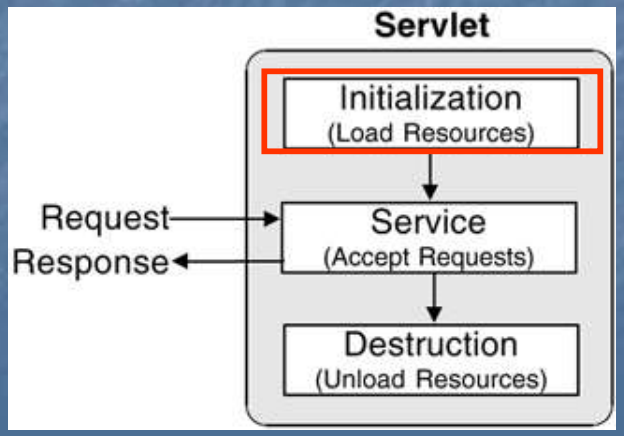
Life Cycle

- Initialize
- Service
- Destroy



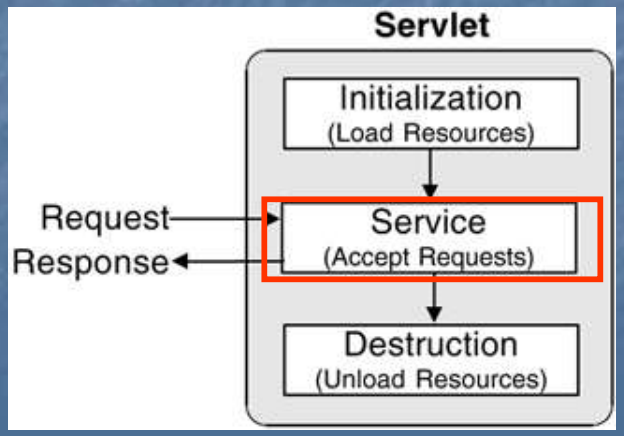
Life Cycle: Initialize

- Servlet is created when servlet container receives a request from the client
- Init() method is called only once



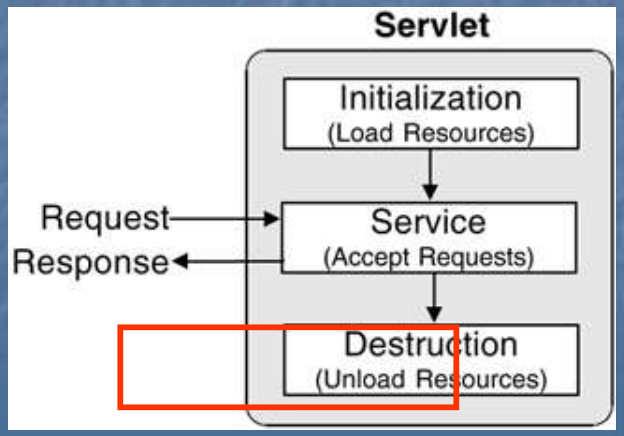
Life Cycle: Service

- Any requests will be forwarded to the service() method
  - doGet()
  - doPost()
  - doDelete()
  - doOptions()
  - doPut()
  - doTrace()



Life Cycle: Destroy

- destroy() method is called only once
- Occurs when
  - Application is stopped
  - Servlet container shuts down
- Allows resources to be freed



Servlet API

- The javax.servlet and javax.servlet.http packages represent interfaces and classes for servlet api.
- The **javax.servlet** package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.
- The **javax.servlet.http** package contains interfaces and classes that are responsible for http requests and responses.

Interfaces in javax.servlet package

RequestDispatcher	Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server.
Servlet	Defines methods that all servlets must implement.
ServletConfig	A servlet configuration object used by a servlet container to pass information to a servlet during initialization.
ServletContext	Defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file.
ServletRequest	Defines an object to provide client request information to a servlet.
ServletResponse	Defines an object to assist a servlet in sending a response to the client.

Classes in javax.servlet package

GenericServlet	Defines a generic, protocol-independent servlet.
ServletException	Defines a general exception a servlet can throw when it encounters difficulty.
UnavailableException	Defines an exception that a servlet or filter throws to indicate that it is permanently or temporarily unavailable.

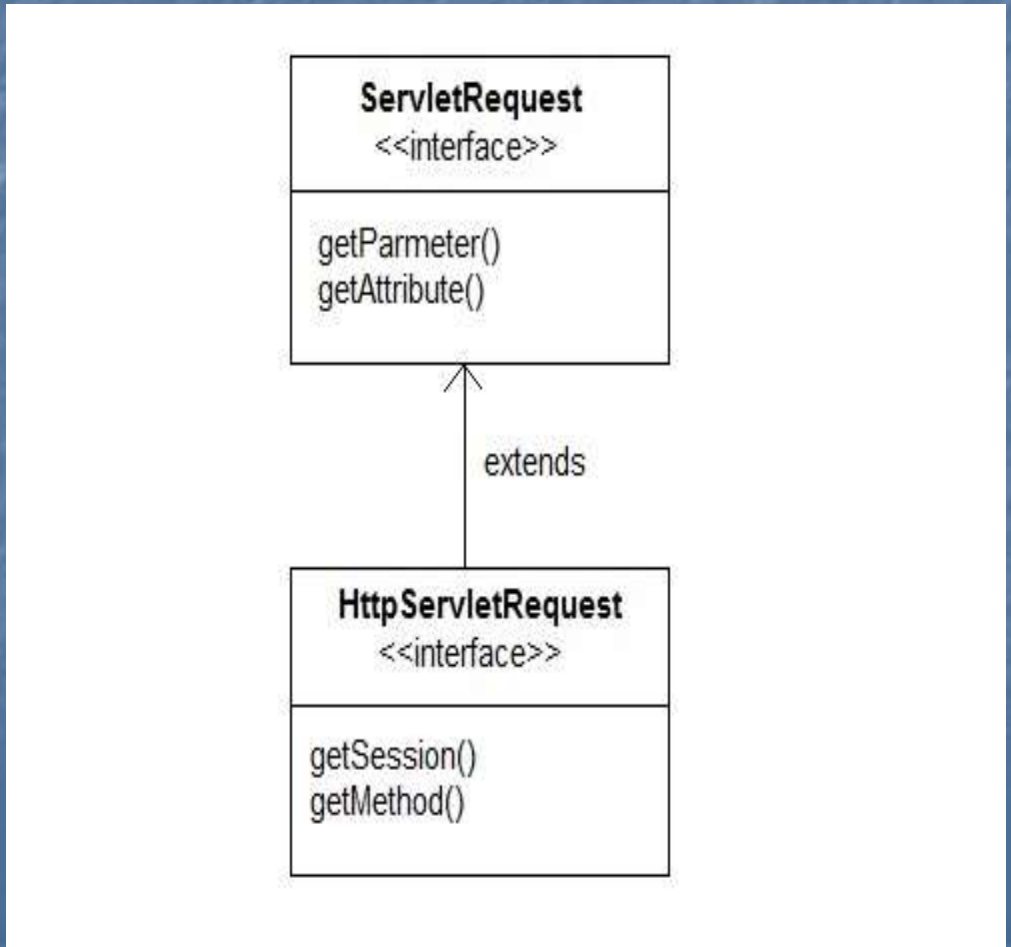
Interfaces in **javax.servlet.http**

HttpServletRequest	Extends the ServletRequest interface to provide request information for HTTP servlets.
HttpServletResponse	Extends the ServletResponse interface to provide HTTP-specific functionality in sending a response.
HttpSession	Provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

Classes in **javax.servlet.http**

Cookie	Creates a cookie, a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server.
HttpServlet	Provides an abstract class to be subclassed to create an HTTP servlet suitable for a Web site.

Servlet API provides two important interfaces **javax.servlet.ServletRequest** and **javax.servlet.http.HttpServletRequest** to encapsulate client request.



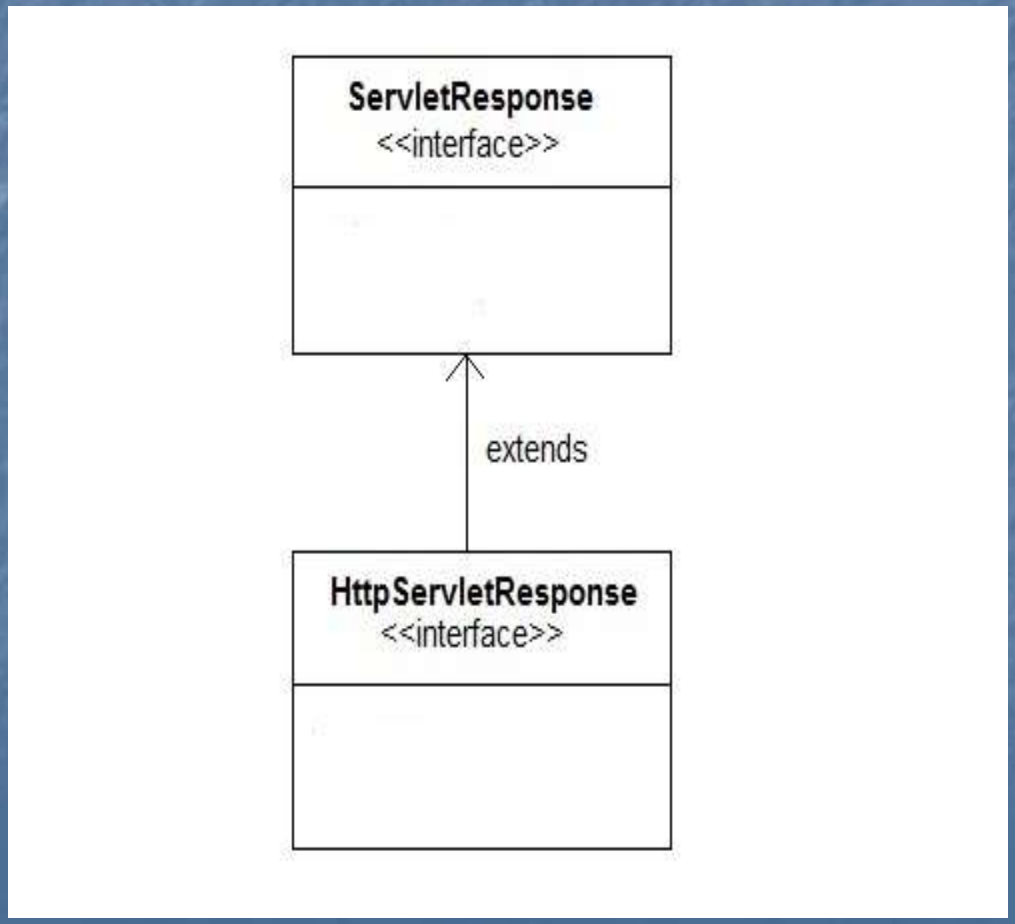
Some Important Methods of ServletRequest

Methods	Description
Object getAttribute(String name)	return attribute set on request object by name
String getLocalAddr()	returns the Internet Protocol(IP) address of the interface on which the request was received
String getParameter(String name)	returns value of parameter by name
String[] getParameterValues(String name)	returns an array of String objects containing all of the values the given request parameter has, or null if the parameter does not exist
ServletContext getServletContext()	return the servlet context of current request.
String getServerName()	returns the host name of the server to which the request was sent
int getServerPort()	returns the port number to which the request was sent

Some important methods of HttpServletRequest

Methods	Description
Cookies getCookies()	returns an array containing all of the Cookie objects the client sent with this request
HttpSession getSession( )	returns a session if session is already exist, it return existing session else create a new session
String getMethod()	Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT.

Servlet API provides two important interfaces **ServletResponse** and **HttpServletResponse** to assist in sending response to client.



Some Important Methods of ServletResponse

Methods	Description
PrintWriter getWriter( )	returns a PrintWriter object that can send character text to the client.
void setBufferSize(int size)	Sets the preferred buffer size for the body of the response
void setContentLength(int len)	Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header
void setContentType(String type)	sets the content type of the response being sent to the client before sending the respond.
boolean isCommitted( )	returns a boolean indicating if the response has been committed

Some important methods of HttpServletResponse

Methods	Description
void addCookie(Cookie cookie)	adds the specified cookie to the response.
void sendRedirect(String location)	Sends a temporary redirect response to the client using the specified redirect location URL and clears the buffer
int getStatus()	gets the current status code of this response
String getHeader(String name)	gets the value of the response header with the given name.
void setHeader(String name, String value)	sets a response header with the given name and value
void setStatus(int sc)	sets the status code for this response
void sendError(int sc, String msg)	sends an error response to the client using the specified status and clears the buffer

Get vs. Post Request

GET	POST
1) In case of Get request, only **limited amount of data** can be sent because data is sent in header.	In case of post request, **large amount of data** can be sent because data is sent in body.
2) Get request is **not secured** because data is exposed in URL bar.	Post request is **secured** because data is not exposed in URL bar.
3) Get request **can be bookmarked.**	Post request **cannot be bookmarked.**
4) Get request is **idempotent** . It means second request will be ignored until response of first request is delivered	Post request is **non-idempotent.**

Request Dispatcher Interface

- A RequestDispatcher object can be used to forward a request to the resource or to include the resource in a response. The resource can be dynamic or static.
- getRequestDispatcher() method of ServletRequest returns the object of RequestDispatcher.
- RequestDispatcher dis= request.getRequestDispatcher("index.html");

RequestDispatcher interface provides two important methods

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

sendRedirect() Method of HttpServletResponse Interface

- Sends a redirect response to the client using the specified redirect location URL. The url passed to this method should be relative.
- response.sendRedirect("relative_url");
- The main difference between a **redirection** and a **request dispatching** is that, redirection makes the client(browser) create a new request to get to the resource, the user can see the new URL while request dispatch get the resource in same request and URL does not changes.
- Also, another very important difference is that, sendRedirect() works on **response** object while request dispatch work on **request** object.

ServletConfig interface

- When the **Web Container** initializes a servlet, it creates a **ServletConfig** object for the servlet. ServletConfig object is used to pass information to a servlet during initialization by getting configuration information from **web.xml**(Deployment Descriptor).
- **getServletConfig() method** of Servlet interface returns the object of ServletConfig.
- Methods: ServletContext getServletContext() returns a reference to the ServletContext.
- String getServletName() returns the name of the servlet instance.
- String getInitParameter(String name)returns a String value initialized parameter, or NULL if the parameter does not exist.

ServletContext Interface

- For every **Web application** a **ServletContext** object is created by the web container. ServletContext object is used to get configuration information from **Deployment Descriptor**(web.xml) which will be available to any servlet or JSPs that are part of the web app.
- **getServletContext() method** of Servlet interface returns the object of ServletContext.
- Methods:**String getInitParameter(String name):**Returns the parameter value for the specified parameter name.
- **Enumeration getInitParameterNames():**Returns the names of the context's initialization parameters.
- **void setAttribute(String name,Object object):**sets the given object in the application scope.
- **Object getAttribute(String name):**Returns the attribute for the specified name.

ServletContext Interface

- **Advantages of ServletContext**
- Provides communication between servlets
- Available to all servlets and JSPs that are part of the web app
- Used to get configuration information from web.xml

Difference between Context Init Parameters and Servlet Init Parameter

Context Init parameters	Servlet Init parameter
Available to all servlets and JSPs that are part of web	Available to only servlet for which the <init-param> was configured
Context Init parameters are initialized within the <web-app> not within a specific <servlet> elements	Initialized within the <servlet> for each specific servlet.
ServletContext object is used to get Context Init parameters	ServletConfig object is used to get Servlet Init parameters
Only one ServletContext object for entire web app	Each servlet has its own ServletConfig object

Introduction to Attribute

- An **attribute** is an object that is used to share information in a web app. Attribute allows Servlets to share information among themselves.
- **How to SET an Attribute in one servlet**
- ServletContext sc1 = getServletContext();

sc1.setAttribute("user","Ramesh");

How to GET an Attribute on another servlet

ServletContext sc1 = getServletContext();

String s1=sc1.getAttribute("user");

out..println("welcome "+ s1);

Managing Session in Servlets

- We all know that **HTTP** is a stateless protocol. All requests and responses are independent. But sometimes you need to keep track of client's activity across multiple requests.
- **Session Management** is a mechanism used by the **Web container** to store session information for a particular user.
- **Cookies**
- **Hidden form field**
- **URL Rewriting**
- **HttpSession**

HttpSession Interfaces in **javax.servlet.http**

HttpSession object is used to store entire session with a specific client. We can store,
retrieve and remove attribute from **HttpSession** object.
Any servlet can have access to **HttpSession** object throughout the getSession() method of
the **HttpServletRequest** object.

Some Important Methods of HttpSession

Methods	Description
void setAttribute(s1,s2)	Assign a string s2 to s1.
String getId()	returns a string containing the unique identifier assigned to the session.
void invalidate()	destroy the session
String getAttribute(s1)	Get string from s1

Example of HttpSession

//Use to create session object in one servlet

HttpSession session = request.getSession();
session.setAttribute("user", “Ram”);

//Access user information on another servlet

- HttpSession session = request.getSession();
- String user = (String)session.getAttribute("user");

Saving State

- Session Tracking
  - A mechanism that servlets use to maintain state about a series of requests from the same user (browser) across some period of time.
- Cookies
  - A mechanism that a servlet uses to have clients hold a small amount of state-information associated with the user.

Using Cookies for Session Management

- **Cookies** are small pieces of information that are sent in response from the web server to the client. **Cookies** are the simplest technique used for storing client state.
- **Cookies** are stored on client's computer. They have a lifespan and are destroyed by the client browser at the end of that lifespan.

Difference between Session and Cookie

Cookie	Session
Cookie data are stored at client side.	Session data are stored at server side.
Not secure	Secure
Remembers info until deleted by you or expiry	Remembers info until web site time-out
Usually contains an id string	Usually contains more complex information
Specific identifier links to server	Specific identifier links to user
Cookies are stored in user's browser.	Sessions are not stored in user's browser.