Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs. Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.

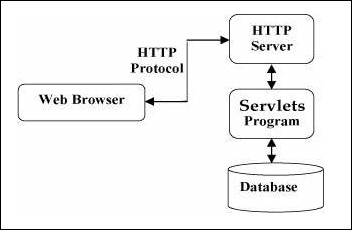
Applications of Servlet:

* Read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.
* Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.
* Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.
* Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.
* Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

Java Servlets are programs that run on a Web or Application server and act as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

Servlets Architecture:



Java Servlets are Java classes run by a web server that has an interpreter that supports the Java Servlet specification.

Servlets can be created using the javax.servlet and javax.servlet.HTTP packages, which are a standard part of Java's enterprise edition, an expanded version of the Java class library that supports large-scale development projects.

These classes implement the Java Servlet and JSP specifications.

Life Cycle of Servlet:

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

* The servlet is initialized by calling the init() method.
* The servlet calls service() method to process a client's request.
* The servlet is terminated by calling the destroy() method.
* Finally, the servlet is garbage collected by the garbage collector of the JVM.
* The init() method is called only once. It is called only when the servlet is created, and not called for any user requests afterward. So, it is used for one-time initializations, just as with the init method of applets.
* The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client( browsers) and to write the formatted response back to the client. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate

## The doGet() Method: A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

## The doPost() Method: A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by the doPost() method.

## The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

## The HttpSession Object: servlet provides HttpSession Interface which provides a way to identify a user across more than one-page request or visit a Web site and to store information about that user.

## Drawbacks of Servlet:

## We can’t write HTML code directly we can write indirectly using the println() or PrintWriter.

## If we modify the servlet code we need to restart the server.

## For every servlet class, we need to write the servlet and servlet-mapping.

## JSP:

## We write the HTML code directlyand we can write java code indirectly using the JSP tags.

## If we modify the jsp file no need to restart the server.

## For JSP file no need of web.xml or servlet or servlet mapping tags.

## In JSP we have 9 implicit objects:

## Out-> JspWriter

## Request-> HttpServletRequest

## Response->HttpServletResponse

## Config ->ServletConfig

## Application ->ServletContext

## Session-> HttpSession

## Exception ->Throwable

## Page ->object

## JSP Tags:

## Declaration tag <%! %> -- We can declare the instance variable or method or static variable or method

## Scriplet Tag <% %> -- request processing and response generation

## Expression Tag<%=>

## INTERVIEW QUESTIONS:

* How to read the name of all parameters in the servlet?

getParameterNames() method of HttpServletRequest returns complete list of all parameters in the current request. This method returns an Enumeration that contains the parameter names in an unspecified order. Once we have an Enumeration, we can loop down the Enumeration in the standard manner, using the hasMoreElements() method to determine when to stop and using the nextElement() method to get each parameter name.

* What is a session?

The session provides a way to identify a user across more than one-page request or visit a Web site and to store information about that user. The session persists for a specified time period, across more than one connection or page request from the user.