Servlet

What is Application

An application program is a computer program designed to carry out a specific task, typically to be used by end-users. For example, Web browsers, word processors, games, and utilities are all applications. The word "application" is used because each program has a specific application for the user.

What are types of Application

a. Stand Alone Applications
b. Web Applications
c. Enterprise Applications
d. Mobile Applications

What is Web Application

Any application program that is stored on a remote server and delivered over the Internet through a browser interface.
Web applications do not need to be downloaded since they are accessed through a network. Users can access a Web application through a web browser such as Google Chrome, Mozilla Firefox or Safari.
For a web app to operate, it needs a Web server and a database. Web servers manage the requests that come from a client. A database can be used to store any needed information.

What is Web

The World Wide Web, commonly known as the Web, is an information system enabling documents and other web resources to be accessed over the Internet.
Documents and downloadable media are made available to the network through web servers and can be accessed by programs such as web browsers using web applications.

URI URL URN

URI – Uniform Resource Identifier.
URL – Uniform Resource Locator.
http-hypertext transfer protocol.
(HTTP was designed to fetch only fetch html pages and send it to clients.)
URN – Uniform Resource Name.

What is Server

A server is a computer program or device that provides a service to another computer program and its user, also known as the client. In a data center, the physical computer that a server program runs on is also frequently referred to as a server. That machine might be a dedicated server or it might be used for other purposes.
Server will accept the request, process the request and give back the response to the client.

A Server response can be either static or dynamic response.
Static - A static html page is sent as response.
Dynamic – A html page is created based on request.

There are mainly two types of servers we use
1.Web Server
2.Application Server

What is web server and application server?

A web server is software and hardware that uses HTTP (Hypertext Transfer Protocol) and other protocols to respond to client requests made over the World wide web.

An application server is a server that hosts applications or software that delivers a business application through a communication protocol. An application server framework is a service layer model. It includes software components available to a software developer through an application programming interface.

Web Server	Application Server
Web server encompasses web container only.	Application server encompasses Web container as well as EJB container.
Web server is useful or fitted for static content.	Whereas application server is fitted for dynamic contents.
Web server consumes or utilizes less resources.	While application server utilize more resources.
Web servers arrange the run environment for web applications.	While application servers arrange the run environment for enterprises applications.
In web servers, multithreading is not supported.	While in application server, multithreading is supported.
Web server's capacity is lower than application server.	While application server's capacity is higher than web server.

What is CGI?

The CGI and Servlet are the programs which reside within the web or application server and assists the communication between the web server and the browser (client side) to generate the web content dynamically. CGI and servlet can be differentiated because they work in different manners and have distinct functionality and features. The CGI (Common Gateway Interface) programs can be designed in the native OS and kept in particular directory. On the other hand, the servlet is a web component which is generally written in Java and run-in java virtual machine.

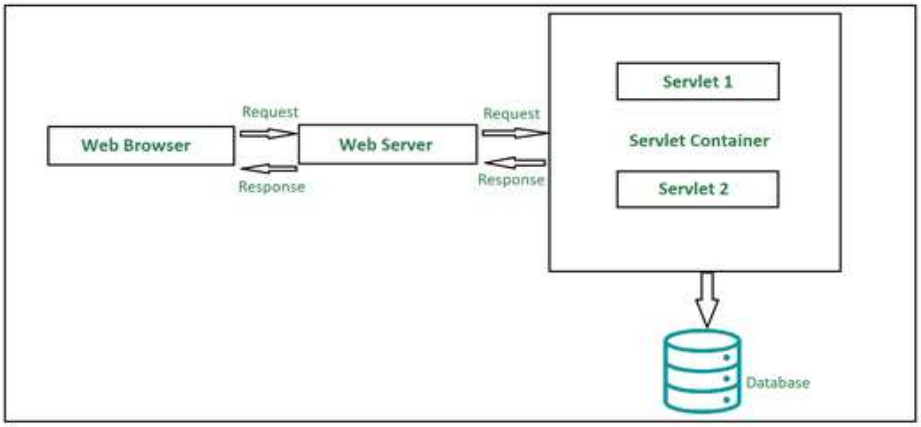
BASIS FOR COMPARISON	CGI	SERVLET
Basic	Programs are written in the native OS.	Programs employed using Java.
Platform dependency	Platform dependent	Does not rely on the platform
Security	More vulnerable to attacks.	Can resist attacks.
Speed	Slower	Faster
Portability	Cannot be ported	Portable

What is Servlet

Servlet means “Server-side component”.

Servlet are the server side programs that run on a web server or application server and acts as a middle layer between requests coming from web browser and databases of application server.

- **Servlet** technology is robust and scalable because of java language.
- Servlet is a technology which is used to create a web application.
- Servlet is an API that provides many interfaces and classes.
- Servlet is an interface that must be implemented for creating any Servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.



Web container / Servlet Container / Deployment Descriptor

A web container is the component of a web server that interacts with the java servlet.

A web container manages the life cycle of servlets and it maps a URL to a particular servlet while ensuring that the requester has relevant access rights.

Java servlet do not have main method, so a container is required to load them. The servlet gets deployed on the web container.

Servlet Life Cycle

The web container maintains the life cycle of a servlet.

1. Servlet class is loaded.

The class loader loads the servlet class. The servlet gets loaded whenever first request is received for that servlet.

2. Servlet instance is created.

The web container creates the instance for the servlet class. Servlet instance is created only once.

3. Init method is invoked.

init() method is used to initialize the servlet.

4. Service method is invoked.

The web container calls the service method whenever request for the servlet is received. If servlet is not initialized it will perform all the previous step and then service () method will be called.

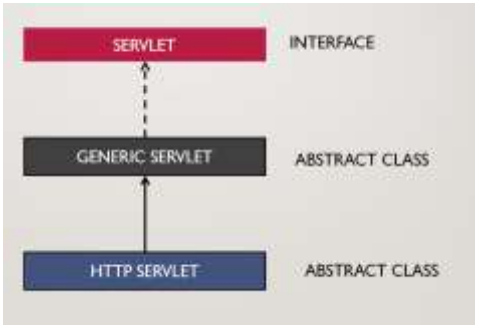
5. Destroy method is called.

The web container calls the destroy method before removing the servlet instance from the service.

Ways of Creating a Servlet Class

1. Implementing Servlet Interface.
2. Extending Generic Servlet.
3. Extending HTTP Servlet

Servlet Hierarchy



1. Servlet is an interface in javax.servlet package.
It has 5 abstract methods.
init()
service()
destroy()
getServletinfo()
getServletConfig()
You can create a servlet by implementing the Servlet interface and providing the implementation for all these methods.

2. Generic Servlet is an abstract class in javax.servlet package .
Generic servlet implements Servlet, ServletConfig and Serializable interfaces. It provides the implementation of all the methods of these interfaces except the service method.
GenericServlet class can handle any type of request so it is protocol-independent.
You can create a servlet by inheriting the GenericServlet class and providing the implementation of the service method.

3. HTTP Servlet is an abstract class in javax.servlet.http package.
The HttpServlet class extends the GenericServlet class and implements Serializable interface. It provides http specific methods such as doGet, doPost, doHead, doTrace etc.
It is an abstract class with no abstract methods.

A subclass of HttpServlet must override at least one method, usually one of these:
doGet , if the servlet supports HTTP GET requests. Used in the case of fetching data.
doPost , for HTTP POST requests. Used in the case of inserting data or sending data.
doPut , for HTTP PUT requests. Used in the case of doing update operations.
doDelete , for HTTP DELETE requests. Used when we want to perform delete operation.

NOTE:
If need to perform any database related operations ie., whenever we want to connect to database using hibernate, we

have to create persistence.xml.
but in this case i.e., in servlet there will be no src/main/resources which we were using earlier.
So now persistence.xml should be created in src->java->webapp->WEB-INF->classes->META-INF->Persistence.xml

PrintWriter

Prints formatted representations of objects to a text-output stream. This class implements all of the print methods found in PrintStream. It does not contain methods for writing raw bytes, for which a program should use unencoded byte streams.
To Get PrintWriter Object we use req.getWriter();

Request Dispatcher

The RequestDispatcher is an interface that comes under package javax.servlet. Using this interface, we get an object in servlet after receiving the request. Using the RequestDispatcher object we send a request to other resources which include (servlet, HTML file, or JSP file). 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.
The RequestDispatcher interface provides two methods. They are:
1. public void forward(ServletRequest request,ServletResponse response)
Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
2. public void include(ServletRequest request,ServletResponse response)
Includes the content of a resource (servlet, JSP page, or HTML file) in the response.

Difference between include() and forward()

1. First and foremost difference is that the include() method includes the content of a resource in the response, the resource could be another Servlet, JSP or HTML file. While the forward() method is used to forward the request to another resource.

Send Redirect

sendRedirect() method redirects the response to another resource, inside or outside the server. It makes the client/browser to create a new request to get to the resource. It sends a temporary redirect response to the client using the specified redirect location URL.
Syntax: void sendRedirect(java.lang.String location)
It accepts the respective URL to which the request is to be redirected. Can redirect the request to another resource like Servlet, HTML page, or JSP page that are inside or outside the server. It works on the HTTP response object and always sends a new request for the object. A new URL that is being redirected can be seen in the browser.

Difference between Request Dispatcher and Send Redirect

Send Redirect	Request Dispatcher
This method is declared in HttpServletResponse Interface.	This method is declared in RequestDispatcher Interface.
Signature: void sendRedirect(String url)	Signature: forward(ServletRequest request, ServletResponse response)
This method is used to redirect client request to some other location for further processing, the new location is available on different server or different context. Our web container handle's this and transfer the request using browser, and this request is visible in	This method is used to pass the request to another resource for further processing within the same server, another resource could be any servlet, jsp page or any kind of file. This process is taken care by web container when we call forward method request is sent to another resource

browser as a new request. Sometime this is also called as client-side redirect.	without the client being informed, which resource will handle the request it has been mention on requestDispatcher object which we can get by Request. This is also called server-side redirect.
In sendRedirect(), web application returns the response to client with status code 302 (redirect) with URL to send the request. The request sent is a completely new request.	RequestDispatcher forward() is used to forward the same request to another resource.
sendRedirect() is handled by browser.	RequestDispatcher forward() is handled internally by the container.
sendRedirect() is slower when compared to dispatcher servlet.forward().	We should use forward() when accessing resources in the same application because it's faster than sendRedirect() method that required an extra network call.
in sendRedirect() URL in address bar change to the forwarded resource.	In forward() browser is unaware of the actual processing resource and the URL in address bar remains same

Session Tracking

Session simply means a particular interval of time.
Session Tracking is a way to maintain state (data) of a user. It is also known as session management in servlet.

Why should a session be maintained?

When there is a series of continuous request and response from a same client to a server, the server cannot identify from which client it is getting requests. Because HTTP is a stateless protocol.
When there is a need to maintain the conversational state, session tracking is needed.
For example, in a shopping cart application a client keeps on adding items into his cart using multiple requests. When every request is made, the server should identify in which client's cart the item is to be added. So, in this scenario, there is a certain need for session tracking.
Solution is, when a client makes a request, it should introduce itself by providing unique identifier every time. There are four different methods to achieve this.

There are four techniques used in Session tracking:
1. Cookies
2. Hidden Form Field
3. URL Rewriting
4. HTTP Session

Cookies

Cookies are the mostly used technology for session tracking.
Cookie is a key value pair of information, sent by the server to the browser. This should be saved by the browser in its space in the client computer. Whenever the browser sends a request to that server it sends the cookie along with it. Then the server can identify the client using the cookie.
In java, following is the source code snippet to create a cookie: Cookie cookie = new Cookie("userID", "7456"); res.addCookie(cookie);
Session tracking is easy to implement and maintain using the cookies. Disadvantage is that, the users can opt to disable cookies using their browser preferences. In such case, the browser will not save the cookie at client computer and session tracking fails.

HttpSession

In HttpSession Session Tracking Mechanism, we will create a separate HttpSession object for each and every user at each and every request we will pick up the request parameters from the request object and we will store them in the respective HttpSession object for the sale of future reusability.
After keeping the request parameters data in the HttpSession object we have to generate the next form at the client browser by forwarding the request to the particular static page or by generating a dynamic form.
To get HttpSession object if we getSession() method then container will check whether any HttpSession object existed for the respective user or not, if any HttpSession object is existed then container will return the existed HttpSession object reference. If no object is existed for the respective user, then container will create a new HttpSession object and return its reference.
HttpSession Methods

1. **public void invalidate():** To destroy the HttpSession object we will use this method.
2. **public void setMaxInactiveInterval(int time):** If we want to destroy the HttpSession object after a particularly ideal time duration then we have to use this method.
3. **public void setAttribute(String name, Object value):** To set an attribute on to the HttpSession object we have to use this method.
4. **public Object getAttribute(String name):** To get a particular attribute value from the HttpSession object we have to use this method.
5. **public void removeAttribute(String name):** To remove an attribute from the HttpSession object we have to use the following method.

Advantages of HttpSession Session Tracking

1. There are no restrictions on the size of the object, any kind of object can be stored in a session.
2. The usage of the session is not dependent on the client's browser.
3. It is secure and transparent.

Disadvantages of HttpSession Session Tracking

1. We will create a HttpSession object for each and every user, where if we increase the number of users then automatically the number of HttpSession objects will be created at the server machine, it will reduce the overall performance of the web application.
2. We are able to identify user-specific HttpSession objects among multiple numbers of HttpSession objects by carrying Session-Id value from client to server and from server to client.

Hidden Form Field

<INPUT TYPE=" hidden" NAME=" technology" VALUE=" servlet">
Hidden fields like the above can be inserted in the webpages and information can be sent to the server for session tracking. These fields are not visible directly to the user, but can be viewed using view source option from the browsers. This type doesn't need any special configuration from the browser of server and by default available to use for session tracking. This cannot be used for session tracking when the conversation included static resources like html pages.

Url rewriting

When a request is made, additional parameter is appended with the url. In general, added additional parameter will be sessionid or sometimes the userid. It will suffice to track the session. This type of session tracking doesn't need any special support from the browser. Disadvantage is, implementing this type of session tracking is tedious. We need to keep track of the parameter as a chain link until the conversation completes and also should make sure that, the parameter doesn't clash with other application parameters.

JSP (Java Server Page)

Java Server Pages (JSP) is a technology for developing web pages that support dynamic content which helps developers insert java code in HTML pages by making use of special JSP tags.
Using JSP, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.
Advantages of JSP: vs. Pure Servlets: It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.

JSP Processing:

The following steps explain how the web server creates the web page using JSP:
- As with a normal page, your browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with .jsp instead of .html.
- The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to println( ) statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior of the page.
- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format, which the servlet engine passes to the web server inside an HTTP response.
- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally, web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page.

JSP Lifecycle

A JSP life cycle can be defined as the entire process from its creation till the destruction which is similar to a servlet life cycle with an additional step which is required to compile a JSP into servlet.

JSP Compilation:

When a browser asks for a JSP, the JSP engine first checks to see whether it needs to compile the page. If the page has never been compiled, or if the JSP has been modified since it was last compiled, the JSP engine compiles the page.
The compilation process involves three steps:
- Parsing the JSP.
- Turning the JSP into a servlet.
- Compiling the servlet.

JSP Initialization:

When a container loads a JSP it invokes the jspInit() method before servicing any requests. If you need to perform JSP-specific initialization, override the jspInit().
method: public void jspInit(){ // Initialization code... }
Typically initialization is performed only once and as with the servlet init method, you generally initialize database connections, open files, and create lookup tables in the jspInit method.

JSP Execution:

This phase of the JSP life cycle represents all interactions with requests until the JSP is destroyed. Whenever a browser requests a JSP and the page has been loaded and initialized, the JSP engine invokes the _jspService() method in the JSP.

The _jspService() method takes an HttpServletRequest and an HttpServletResponse as its parameters as follows:
void _jspService(HttpServletRequest request, HttpServletResponse response) { // Service handling code... }
The _jspService() method of a JSP is invoked once per a request and is responsible for generating the response for that request and this method is also responsible for generating responses to all seven of the HTTP methods ie. GET, POST, DELETE etc.

JSP Cleanup:

The destruction phase of the JSP life cycle represents when a JSP is being removed from use by a container. The jspDestroy() method is the JSP equivalent of the destroy method for servlets. Override jspDestroy when you need to perform any cleanup, such as releasing database connections or closing open files. The jspDestroy() method has the following form:
public void jspDestroy() { // Your cleanup code goes here. }

JSP Tags:

1. JSP Comments

Since JSP is built on top of HTML, we can write comments in JSP file like html comments as
<%-- This is HTML Comment --%> These comments are sent to the client and we can look it with view source option of browsers.
We can put comments in JSP files as: This comment is suitable for developers to provide code level comments because these are not sent in the client response.

2. JSP Scriptlets

Scriptlet tags are the easiest way to put java code in a JSP page.
A scriptlet tag starts with <% and ends with %>
Any code written inside the scriptlet tags go into the _jspService() method.

3. JSP Expression

Since most of the times we print dynamic data in JSP page using out.print() method, there is a shortcut to do this through JSP Expressions.
JSP Expression starts with <%= and ends with %> can be written using JSP Expression as Notice that anything between is sent as parameter to out.print() method. Also notice that scriptlets can contain multiple java statements and always ends with semicolon (;) but expression doesn't end with semicolon.

4. JSP Directives

JSP Directives are used to give special instructions to the container while JSP page is getting translated to servlet source code. JSP directives starts with For example, in above JSP Example, I am using page directive to to instruct container JSP translator to import the Date class.

5. JSP Declaration

JSP Declarations are used to declare member methods and variables of servlet class.
JSP Declarations starts with <%! and ends with %>.
For example we can create an int variable in JSP at class level as <%! Int a=10; %>.

JSP Implicit Objects

JSP supports nine automatically defined variables, which are also called implicit objects.
These variables are:

1. Request:
   This is the HttpServletRequest object associated with the request.
2. Response:
   This is the HttpServletResponse object associated with the response to the client.

3. Out:
   This is the PrintWriter object used to send output to the client.
4. session:
   This is the HttpSession object associated with the request.
5. Application:
   This is the ServletContext object associated with application context.
6. Config:
   This is the ServletConfig object associated with the page.
7. pageContext:
   This encapsulates use of server-specific features like higher performance JspWriters.
8. Page:
   This is simply a synonym for this, and is used to call the methods defined by the translated servlet class.
9. Exception:
   The Exception object allows the exception data to be accessed by designated JSP.

Servlet Context

An object of ServletContext is created by the web container at time of deploying the project. This object can be used to get configuration information from web.xml file. There is only one ServletContext object per web application.
If any information is shared to many servlets, it is better to provide it from the web.xml file using the **<context-param>** element.

Advantage of ServletContext

Easy to maintain if any information is shared to all the servlet, it is better to make it available for all the servlet. We provide this information from the web.xml file, so if the information is changed, we don't need to modify the servlet. Thus it removes maintenance problem.

Servlet Config

An object of ServletConfig is created by the web container for each servlet. This object can be used to get configuration information from web.xml file.
If the configuration information is modified from the web.xml file, we don't need to change the servlet. So it is easier to manage the web application if any specific content is modified from time to time.

Advantage of ServletConfig

The core advantage of ServletConfig is that you don't need to edit the servlet file if information is modified from the web.xml file

Welcome file list

The **welcome-file-list** element of **web-app**, is used to define a list of welcome files. Its sub element is **welcome-file** that is used to define the welcome file.
A **welcome file** is the file that is invoked automatically by the server, if you don't specify any file name.