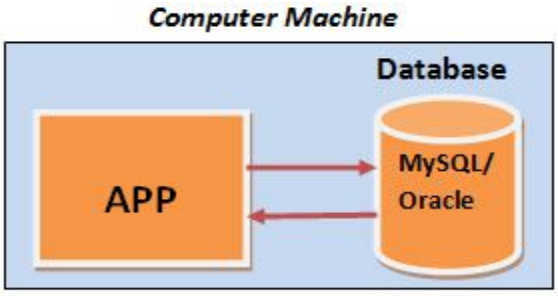
Intro to JEE (Java Enterprise Edition)

Types of Applications: - We can develop various types of applications based on the requirement.

1. Stand alone applications.
2. Client Server applications.
3. Web based applications.
4. Enterprise/ Distributed applications.

1. **Stand Alone:** - Application which is running in a single machine and can be accessed by a single user at a time is called standalone application.
   Eg:- MSWord, Access, etc.

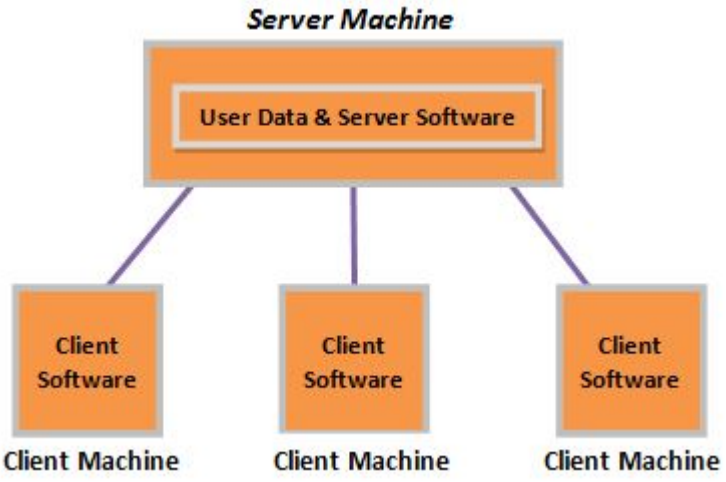


Drawbacks:-

a) Application should be installed in each and every machine.
b) Other users from other machines are unable to share this application.
c) If application migration is required then you need to reinstall the software in all the machines which is maintenance problem.
d) Data sharing is not possible with these kinds of applications.

Technologies required to develop:- C,C++, VB, Java (Swing) etc.
Note: No special server is required)

2. **Client-Server Applications:-**
   Eg: gtalk, yahoo messenger



- Fat client and Thin Server.
- To solve the above problem of data sharing in standalone applications, client-server architecture was designed.
- In client-server application, the application will be divided into 2 parts. Some part of the application will be installed in server machine and some part will be installed in multiple client machines.
- Server is a centralized place to store the data and some part of the application. Data in the server can be accessed by multiple clients who are connected to the server.
- Client-Server architecture solves data sharing problem.

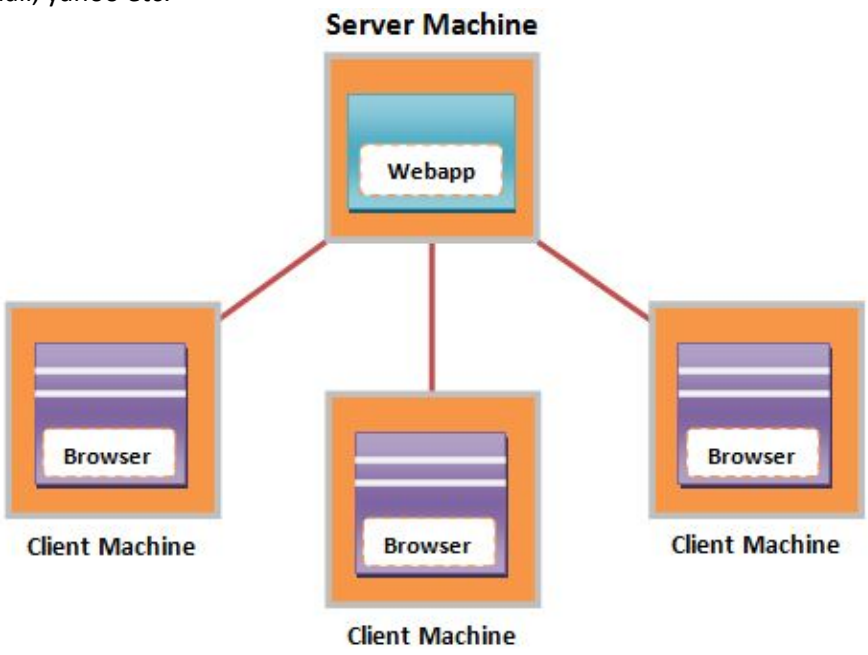
Drawbacks:-

a) Client-Server architecture gives you maintenance issue because we are installing the client software in all the machines. When client software is enhanced, then all the client machines should be updated with new software.

Technologies required developing: - socket programming with C, C++, VC++, and Java.
In java we can implement client server application using java.net package.
(Note: No special server is required)

3. <u>Web based applications :-</u>

- To solve the problems of client-server architecture, web-based applications was designed.
- In this complete web application will be placed in the centralized place called as a web server.
- Web application placed in web server can be accessed by any web client in the world with internet facility.
- Thin client and Fat Server.
- Eg: gmail, yahoo etc.

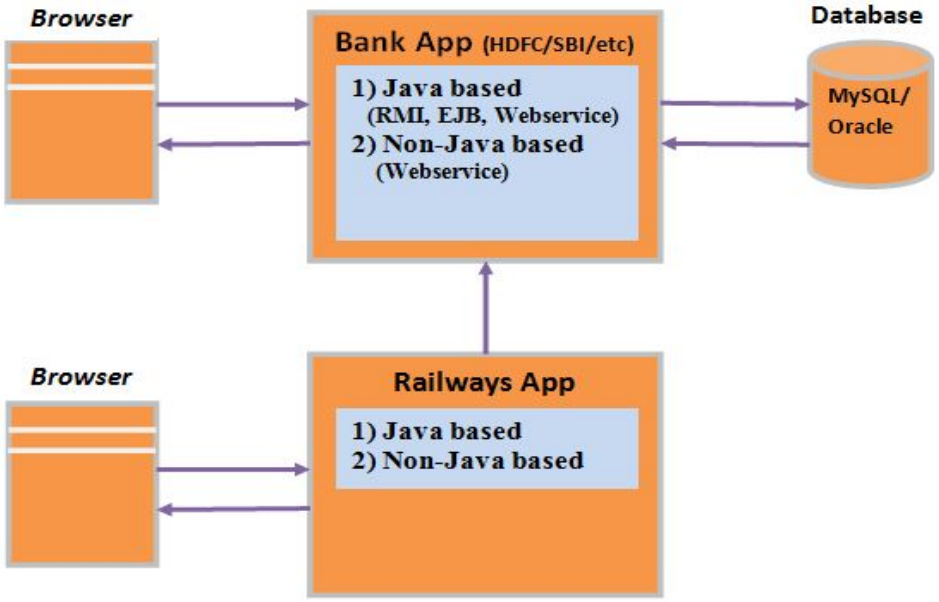


Drawbacks:-

a) This kind of application is limited to a single organization.
b) Technologies required developing: - Any web technologies (Servlet/JSP, PHP, and ASP) To run the web applications you need some 3rd party web server.

   a. For java applications: - Tomcat from Apache, Jboss from Red hat, Weblogic from Oracle, Websphere from IBM, and Glassfish from SUN etc.
   b. For ASP .Net applications: - IIS (Internet Information Server) from Microsoft.

4. Distributed applications/ Enterprise Applications :-



- In web-based applications, functionality will be exposed to those organizational clients only.
- It is not suitable to communicate with another organizational application.
- This gap can be filled by a distributed/enterprise application.
- Distributing computing is the current trend in the market which allows the business partners to share the information among each other.
- Technologies required to develop enterprise applications:-
  1) In Java (SUN/ Oracle):- RMI/ EJB/Web Services
  2) In C# (Microsoft):- DCOM.
  3) In C,C++ :- CORBA
- Server required running java application: - Application server.
  Eg: - JBOSS, Weblogic, Websphere, Glassfish etc

When communication is happening between two applications then there are two possible chances.

1) Homogeneous applications: - Both applications are using the same language and same technology.
   - ✓ For two java based applications:- RMI,EJB, Webservices.
   - ✓ For two .Net based applications:- DCOM.
   - ✓ For two C,C++ based applications :- CORBA.

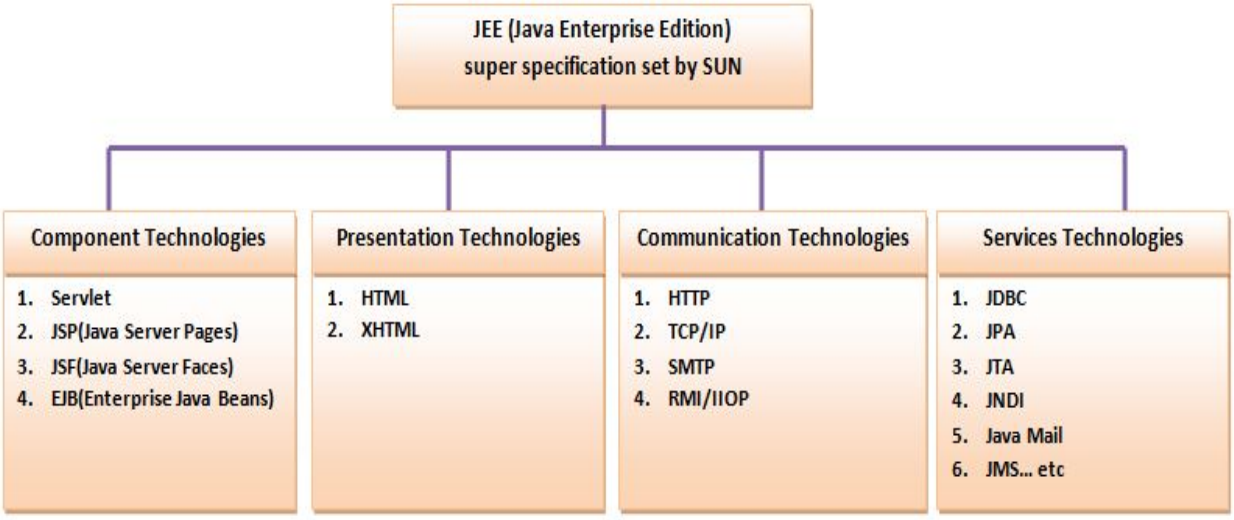
2) Heterogeneous applications:- Both applications are using different language and different technology.
   - ✓ One java and one .NET application:- Web Services technology from Java.

JEE (Java Enterprise Edition)

- SUN has provided three platforms or super specifications for the development of different types of applications. Java is the language used for development of all these type of applications.
  1) JSE (Java Standard Edition) earlier known as J2SE:- platform for building wide variety of applications like standalone (Swing) applications, client server applications or applet applications. It is usually supplied with JDK (Java Development Kit).
  2) JME (Java Micro Edition) earlier known as J2ME:-platform for building java applications for micro devices like mobile phones, PDA etc
  3) JEE (Java Enterprise Edition) earlier known as J2EE:-is a platform for building server side applications. It is usually supplied by various server vendors.

Java and JEE (Java Enterprise Edition)

- Java is the programming language used for developing enterprise application.
- JEE is a platform for building server side applications for enterprises or organizations.
- An enterprise means a business organization. Enterprise applications are software that facilitates various business processes in an enterprise.



Full form of the abbreviations

1) JDBC: - Java Data Base Connectivity.
2) JPA:- Java Persistence API
3) JTA:- Java Transaction API
4) JNDI: - Java Naming and Directory Interface.
5) JMS: - Java Messaging Service.
6) JAAS:- Java Authentication and Authorization Service

- JEE is super specification set by SUN for two people.
  1) All server vendors for implementing the specification.
  2) We java application developers for using the implementation.

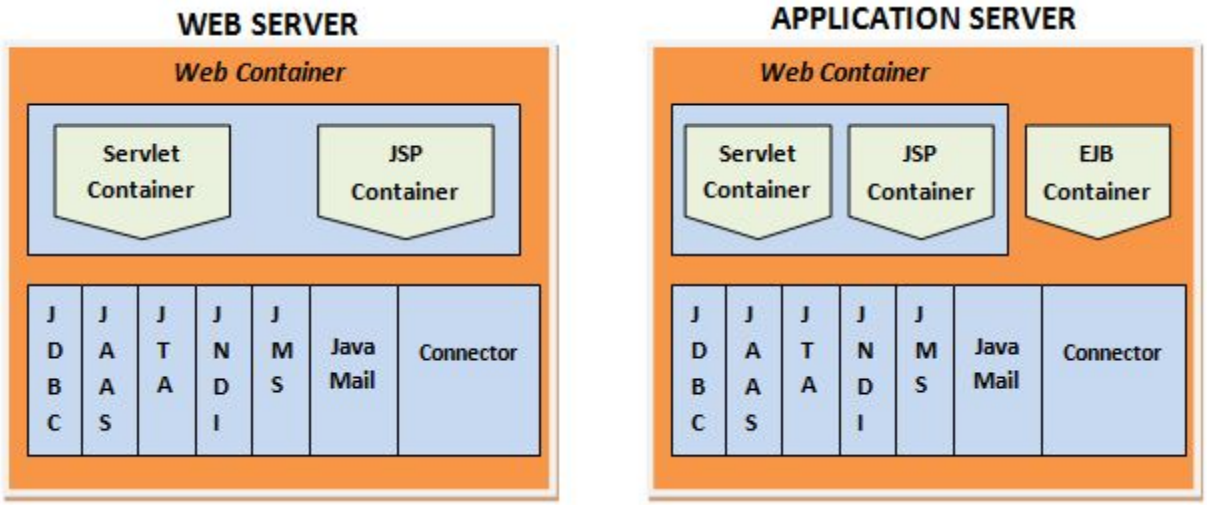
- Various server vendors like Apache (Tomcat), Red hat (JBOSS), IBM (Websphere), Oracle (Weblogic) implement these specifications to provide the support to java application developers.
- There are many sub specifications under JEE like Servlet, JSP (Java Server Page), JSF (Java Server Faces), EJB (Enterprise Java Bean), etc. All these sub specifications solve the various problems which might arise while developing a web or distributed application.
- We application developers use this JEE server or platform for deploying and running our web or distributed application. We application developers develop our application code using Servlet API, JSP API, JSF API, EJB API and deploy or host the application into this JEE server.
- We developers are free to choose any server from any vendor to deploy our web or distributed application.

JEE Server

- JEE server is having the implementation of JEE specification set by SUN. Many server vendors do this implementation.
- Server side applications have additional requirements for development and execution.
- JEE platform or server provides the infrastructure for meeting these needs.
- JEE platform provides the following.
  1. A set of APIs to build applications (Servlets, JSP, JSF and EJB).
  2. A run time infrastructure for hosting and managing applications (Servlet, JSP and EJB container).
  3. A set of service APIs (JDBC for database interaction, Java mail for sending mail, JTA for transaction, JMS for messaging etc) needed for all applications.
  4. JEE servers also provide container services that include threads, life cycle management, resource pooling, memory management etc, which are commonly needed for all applications.

JEE Architecture

- A JEE server is a runtime to manage application components (Servlet, JSP or EJB) and to provide access to the JEE service APIs.
- There are two types of JEE server.
  1) Web server:- contains only web container
  2) Application Server: - contains both web and EJB container.



- A web container is used for hosting Java Servlets and JSP.
- An EJB container is used for hosting Enterprise Java Beans (EJB).
- Each of these containers provides a run time environment for respective components (Servlet, JSP, and EJB).
- JEE components are also called managed objects, as these objects are created and managed within the container runtime.

Web server and Application server

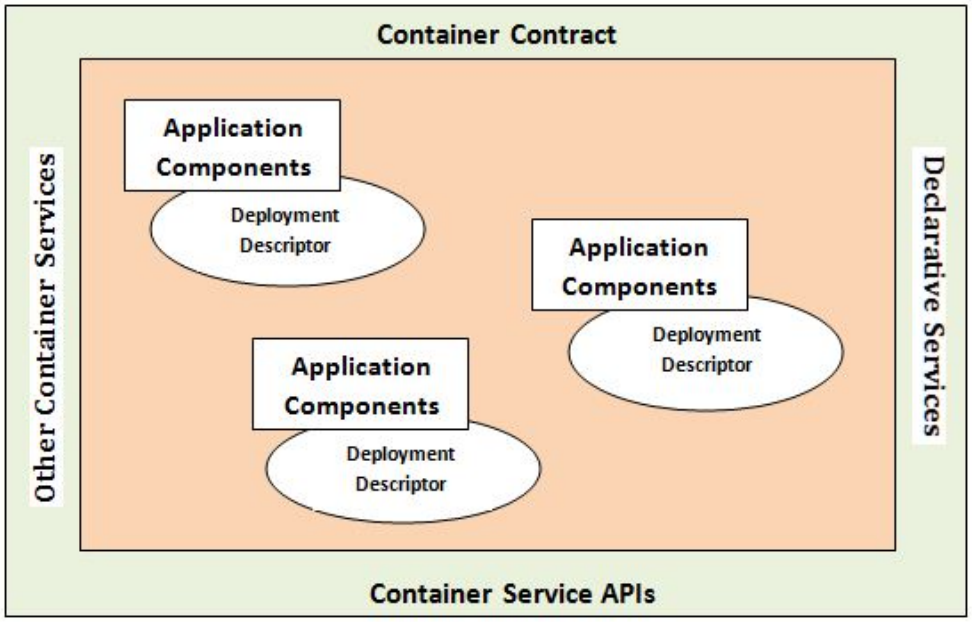
After development of web application or enterprise/distributed application we need to run those applications in a web server or application server.

1) Web Server:- used for hosting and running web applications.
2) Application Server:- can be used for hosting and running web and distributed application.

Difference between Web server and Application server

Web Server	Application Server
1. Web server has web container(servlet and jsp container)	1. Application server has both web container and EJB container.
2. Web server supports only web based applications.	2. Application server supports web based applications and distributed applications.
3. Web container provides support for web components which are implemented with web technologies like servlets and jsps.	3. In application server web container supports the web component (servlet and JSP) and EJB container supports EJB components.
4. Web container provides following services: 1) Networking. 2) I/O Streams. 3) Multithreading. 4) SecURIty management (declaratively). 5) Resource management. 6) Life cycle management. 7) Memory management. 8) etc.	4. In application server, web container will provide all these 7 services. Besides that EJB container provides following services. 1) Networking. 2) I/O Streams. 3) Multithreading. 4) SecURIty management (declaratively). 5) Resource management. 6) Life cycle management. 7) Memory management. 8) Transaction management. 9) Remoting service. 10) Scalability (of business logic). 11) Web service support. 12) Messaging service. 13) Timer service….etc
5. Eg:- Tomcat	5. Eg:- Jboss, Weblogic, Websphere, Glassfish, etc

JEE Container Architecture

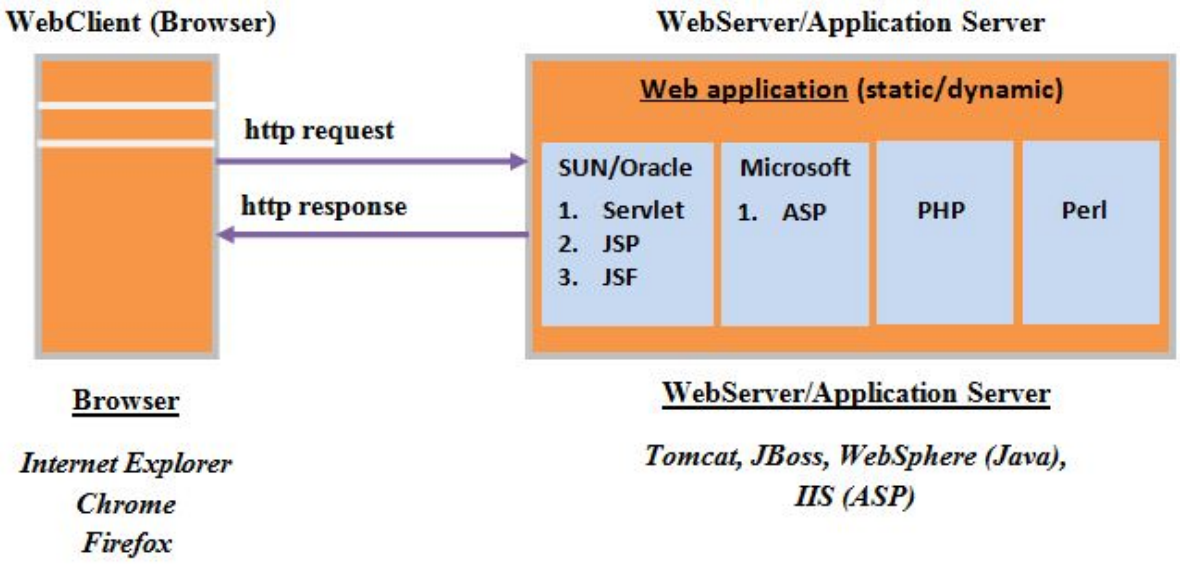


- In this architecture, as developers we are required to provide the following.
  1. Application components:- They include Servlets, JSP, EJB. In JEE application components can be packaged into archive files.
  2. Deployment Descriptors:- It is an XML file that describes the application components. It also includes additional information required by containers for effectively managing application components.

- The rest of the figure forms the container. The architecture of the container can be divided into four parts.
  1. Component contract:- A set of APIs specified by the container, that your application components are required to extend or implement.
  2. Container Service APIs:- Additional services provided by the container, which are commonly required for all applications in the container.(JDBC, JMS, JNDI, JTA, JAAS).
  3. Declarative Services:- Services that the container interposes on your application, based on the deployment description provided for each component, such as secURIty, transactions etc.
     For EJB containers, the declarative services include transaction and secURIty (JTA and JAAS for EJB) and for web containers, the declarative services include only secURIty (JAAS).
  4. Other container services: - Other runtime services, related to component life cycle, resource pooling, garbage collection, etc.

Servlet

- Servlet is a web-technology from SUN Microsystems which will be used to develop web-based applications.
- Servlet is a sub specification of JEE platform which is set by SUN, implemented by various server vendors and used by java developers.
- Servlets will be used to develop server side components.



- Here two participants are involving:
  1. Web client.
  2. Web server/ Application server

1. **Web-Client: -** is an application which sends the request to the server and also responsible to receive the response and display it to the client. Any browser can be used as web-client. Eg:- IE, Mozilla, Safari, etc.
2. **Web Server: -** is an application which receives the request from the web-client, processes the request and sends the response to the web-client.
   - ☞ Web server is used to run web applications.
   - ☞ Web Server has a web container (servlet and jsp container).
     Eg: - Apache Tomcat.

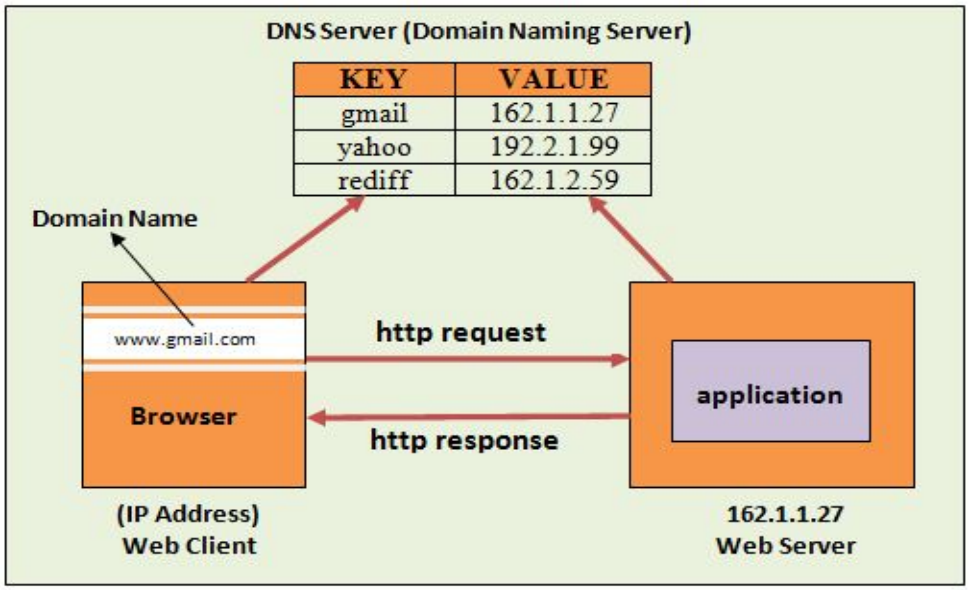
Application Server: - is also a software/application which is responsible to receive the request from the client, process the request and send the response to the client.

- ☞ Application Server has a web container and EJB container.
- ☞ Application server is also used to run web applications which are implemented using Servlets and JSPs.
- ☞ It is also used to run enterprise applications which are implemented using EJB, Servlets & JSP.
  Eg:- Weblogic(Oracle), JBOSS(Redhat),Websphere (IBM), Glassfish (SUN) etc.

Web Technologies: - Technologies which you are going to use to develop web-based applications.

- SUN provided web technologies are Servlet and JSP.
- Other Eg. : - ASP (.NET), PHP, Perl script.
- **The main features and advantages of Servlet technology are:-**
  1) Web support (Can accept web requests)
  2) Standard API for developing web application.
  3) I/O.
  4) Networking.
  5) Collecting input values.
  6) Life cycle management.
  7) Resource Management.
  8) Scalability.
  9) Memory management and garbage collection.
  10) Multithreading.
  11) Session management.
  12) Events and Listeners.
  13) Filter (AOP - Aspect Oriented Programming).

Hosting an application in internet or World Wide Web (WWW)

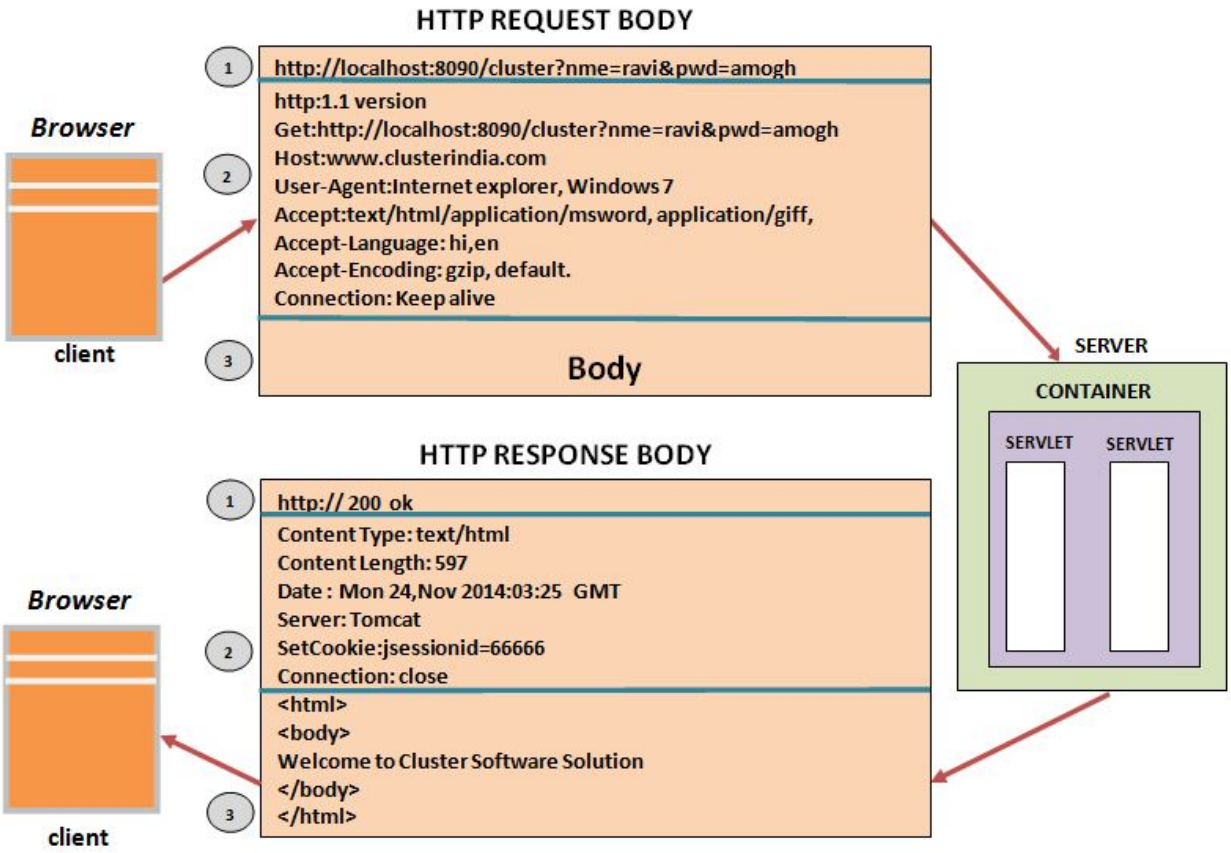


- When you hit the browser with some URL, first request will be given to DNS(Domain Naming Server).
- In DNS server, domain will be reserved for IP address and then request would go to the server where the application is running.
- Once server receives the request it processes the request and sends the response to the client.
- Server identifies the client IP address from the HttpRequest.

- In the World Wide Web (www) the communication between the web client and web server is happening with HTTP (Hyper Text Transfer Protocol).

HTTP (Hyper Text Transfer Protocol)

- HTTP request and HTTP response has 3 parts.
  1) URL or status line.
  2) Header information.
  3) Body



- Http request body contains URL, the HTTP method (GET, POST) and the request values.
- Http response body contains status code, content type (MIME) and the actual content to be displayed.
- Http response body contains html code which can be understood by any web browser. Sometimes Http response body contains images, pdf, word, excel document, etc.
- Http response body allows various content types. These types are called as MIME(Multipurpose Internet Mail Extension) like html/text, img/giff, application/word, application/pdf, video/mpeg etc.

HTTP:-

- Http is the communication protocol used between the client (browser) and the server.
- Http is a stateless protocol.
- Http is responsible to carry the hyper text from client to server and server to client.
- Http sits on top of TCP/IP.
- IP is the protocol which ensures the data packets reach the correct destinations.

- TCP just controls the data transmission, i.e. TCP checks whether all the data packets started from the server have reached the client or not and vice versa.
- When a request is being sent from a browser to the server we can use one of the following http request methods:
  1) get.
  2) post.
  3) head.
  4) put.
  5) trace.
  6) options.
  7) delete.
  8) connect.

- Following is the way to specify the http method.

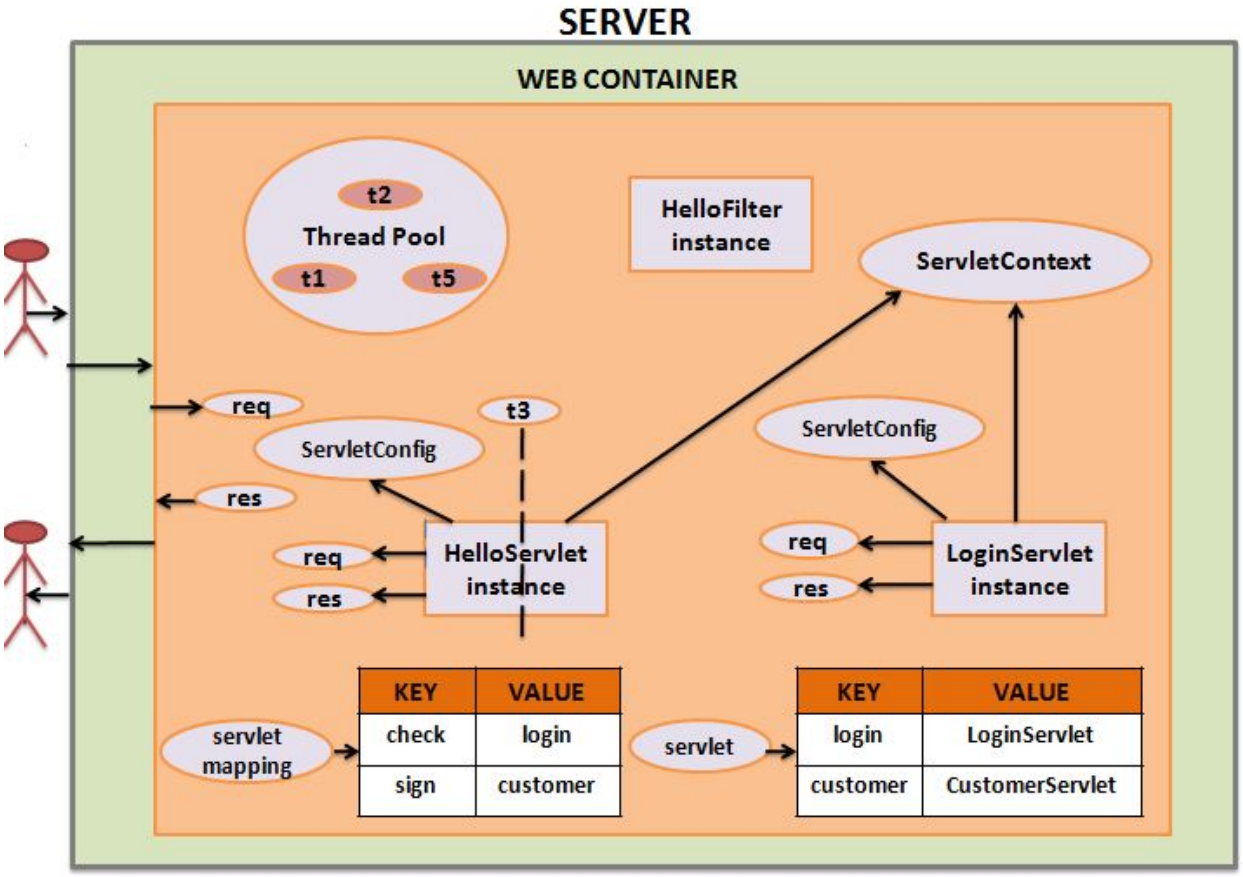
```
<form action="../cluster" method="post" >
```

- If you don't specify `method="post"` default value is "*get*".

Difference between GET and POST

GET	POST
1) When a request is sent using GET method, data will be attached to the URL as a query string.	When a request is sent using POST method, data will be sent to the server along with the Http request body.
2) We can send limited amount of data using GET method.	Using POST we can send unlimited amount of data, size is not fixed because data is carried in the body
3) GET method is not secure because of attaching the data in the URL which can be seen by the user.	POST method is secure because it carries the data in the request body and does not expose the data in the URL.
4) GET cannot be used for carrying sensitive data like password.	POST is used for carrying sensitive and secure data.
5) GET is used for bookmarking.	POST cannot be used for bookmarking.
6) GET request remains in the browser history.	POST requests do not remain in the browser history.

Server Life Cycle and Servlet Life Cycle



Server Life Cycle:-

At server start up following things will happen:-

1. Container initializes the web.xml file i.e. using the SAX parser container reads the data from web.xml document and stores in java objects. If any errors are found while parsing the document then SAXParserException will be thrown. If SAXParserException comes at server start up , then it means that application has not been deployed properly.
2. Container creates ServletContext object and initializes the ServletContext object, with the context parameters as specified in the web.xml.
3. All the Filter instances in the application will be created by calling the init() method on each filter.
4. Thread pool will be created by the container. Threads inside this pool will be used by the container whenever client is sending the request. Based on the client usage container can increase the pool size and can also decrease the pool size i.e. in the peak hours when more clients are coming container will increase the thread pool size and in the non peak hours when client usage is less, container can decrease the pool size.
5. Servlet instances will be created for all the servlets for which <load-on-startup> tag has been used.

Note: There are 2 ways a servlet instance gets created.

a) When server starts up. (<load-on-startup> should be used in web.xml).
b) When client makes a first request to the servlet. (if <load-on-startup> tag is not used)

6. Server is now ready for accepting the first request from the client.

Servlet Life Cycle:-

In servlet life container calls the following 3 methods on the servlet instance.

1. init() (executes only once when the servlet instance gets created)
2. service() (executes every time the request comes to the servlet)
3. destroy() (executes only once before the servlet instance gets destroyed)

Once the server is ready, and client is sending a first request to a servlet following things will happen:

1) Web server receives the request from the web client and delegates it to the web container.
2) Web container collects the request URI (value of action attribute in form tag) from the incoming request and verifies any servlet is configured with URL pattern similar to this.
3) If the request URI is not found with any servlet then error message "request URI not available status code: 404" will be sent to the client.
4) With the URL pattern servlet name will be identified and then servlet class also will be identified.
5) Container verifies whether servlet class is available or not. If servlet class is not available in the specified location (WEB-INF/classes folder) then an exception will be thrown called ClassNotFoundException.
6) If the servlet class is found then it will be loaded into the main memory.
7) Container creates the servlet instance by calling the default constructor. (No chance of calling any argumented constructor).
8) Container creates ServletConfig object and config object will be initialized with init or config parameters as specified in web.xml related to the particular servlet.
9) Container calls init() method by passing ServletConfig as parameter.
10) Container takes one thread from the thread pool which was created at server startup and hands over the request processing to that thread.
11) Thread is responsible to start the processing. First it will create ServletRequest and ServletResponse objects. Request object will be initialized with the incoming data (request parameters, request headers etc). The thread then calls service() method by passing request and response as arguments.
12) service() method processes the request and provides the response to the client.
13) Once the response is handed over to the client, request and response objects will be destroyed and the thread will be returned back to the pool.
14) The servlet instance is not destroyed and waits for further requests from the client.

Note: If the servlet instance gets created at server startup because of <load-on-startup> tag then its constructor will be called, ServletConfig object will be created, init() method will be called and then the servlet instance waits for requests.

If a second or more requests comes to the same servlet or if the request comes to a servlet which got instantiated dURIng the server startup following things will happen:

1) A thread is taken from the pool, creates new request and response objects and calls the service method.

2) After service() method completes execution and provides response, Request and Response objects are destroyed and the thread is returned back to the pool.

Server Life Cycle contd...

At server shutdown time following things will happen:

1. Container destroys all the servlet instances by calling destroy() method on each instance.
2. Container destroys all the ServletConfig objects related to all the servlets.
3. Container destroys all the filter instances by calling destroy() method on each instance.
4. ServletContext object will be destroyed.
5. Thread pool will be destroyed.
6. web.xml is unloaded. (The objects which contain the data belonging to web.xml will be destroyed).