Application Server Vs Web Server

-Gagandeep Singh

Application Server and Web Server in Java both are used to host Java web application.

On Java J2EE perspective main difference between **Web Server and Application Server** is support of EJB.

In order to run EJB or host Enterprise Java Application (.ear) file, you need an application server like JBoss, WebLogic, WebSphere or Glassfish, while you can still run your servlet and JSP or Java Web Application (.war) file inside any web server like Tomcat or Jetty.

Basic differences

1. **Application Server** supports **distributed transaction and EJB**, while **Web Server** only supports Servlets and JSP.

2. **Application Server** can contain **Web Server** in them, most of App server e.g. JBoss or WAS has Servlet and JSP container.

3. Though its not limited to **Application Server** but they used to provide services like **Connection pooling**, **Transaction management**, messaging, clustering, load balancing and persistence. Now Apache tomcat also provides connection pooling.

4. In terms of logical difference between web server and application server-**Web Server** is supposed to provide http protocol level service while **Application Server** provides support to web service and expose business level service e.g. EJB.

5. **Application Server** are more heavy than web server in terms of resource utilization.

A **Web Server** can be either a computer program or a computer running a program that is responsible for accepting HTTP requests from clients, serving back HTTP responses along with optional data contents, which usually are web pages such as HTML documents and linked objects on it.

An **Application Server** is the kind of software engine that will deliver various applications to another device. It is the kind of computer found in an office or university network that allows everyone in the network to run software off of the same machine.

Comparison Chart

	Application Server	Web Server
What is it?	A server that exposes business logic to client applications through various protocols including HTTP.	A server that handles HTTP protocol.
Job	Application server is used to serve web based applications and enterprise based applications(i.e servlets, jsps and ejbs...). Application servers may contain a web server internally.	Web server is used to serve web based applications.(i.e servlets and jsps)
Functions	To deliver various applications to another device, it allows everyone in the network to run software off of the same machine.	Keeping HTML, PHP, ASP etc files available for the web browsers to view when a user accesses the site on the web, handles HTTP requests from clients.
Supports	distributed transaction and EJB's	Servlets and JSP
Resource utilization	High	Low

The Web Server

A Web server handles the HTTP protocol. When the Web server receives an HTTP request, it responds with an HTTP response, such as sending back an HTML page. To process a request, a Web server may respond with a static HTML page or image, send a redirect, or delegate the dynamic response generation to some other program such as CGI scripts, JSPs (JavaServer Pages), Servlets, ASPs (Active Server Pages), server-side JavaScripts, or some other server-side technology. Whatever their purpose, such server-side programs generate a response, most often in HTML, for viewing in a Web browser.

Understand that a Web server's delegation model is fairly simple. When a request comes into the Web server, the Web server simply passes the request to the program best able to handle it. The Web server doesn't provide any functionality beyond simply providing an environment in which the server-side program can execute and pass back the generated responses. The server-side program usually provides for itself such functions as transaction processing, database connectivity, and messaging.

While a Web server may not itself support transactions or database connection pooling, it may employ various strategies for fault tolerance and scalability such as load balancing, caching, and clustering—features oftentimes erroneously assigned as features reserved only for application servers.

The Application Server

As for the application server, according to our definition, an application server exposes business logic to client applications through various protocols, possibly including HTTP. While a Web server mainly deals with sending HTML for display in a Web browser, an application server provides access to business logic for use by client application programs. The application program can use this logic just as it would call a method on an object (or a function in the procedural world).

Such application server clients can include GUIs (graphical user interface) running on a PC, a Web server, or even other application servers. The information traveling back and forth between an application server and its client is not restricted to simple display markup. Instead, the information is program logic. Since the logic takes the form of data and method calls and not static HTML, the client can employ the exposed business logic however it wants.

In most cases, the server exposes this business logic through a component API, such as the EJB (Enterprise JavaBean) component model found on J2EE (Java 2 Platform, Enterprise Edition) application servers. Moreover, the application server manages its own resources. Such gate-keeping duties include security, transaction processing, resource pooling, and messaging. Like a Web server, an application server may also employ various scalability and fault-tolerance techniques.

An example

As an example, consider an online store that provides real-time pricing and availability information. Most likely, the site will provide a form with which you can choose a product. When you submit your query, the site performs a lookup and returns the results embedded within an HTML page.

The site may implement this functionality in numerous ways. I'll show you one scenario that doesn't use an application server and another that does. Seeing how these scenarios differ will help you to see the application server's function.

Scenario 1: Web server without an application server

In the first scenario, a Web server alone provides the online store's functionality. The Web server takes your request, then passes it to a server-side program able to handle the request. The server-side program looks up the pricing information from a database or a flat file. Once retrieved, the server-side program uses the information to formulate the HTML response, then the Web server sends it back to your Web browser.

To summarize, a Web server simply processes HTTP requests by responding with HTML pages.

Scenario 2: Web server with an application server

Scenario 2 resembles Scenario 1 in that the Web server still delegates the response generation to a script. However, you can now put the business logic for the pricing lookup onto an application server. With that change, instead of the script knowing how to look up the data and formulate a response, the script can simply call the application server's lookup service. The script can then use the service's result when the script generates its HTML response.

In this scenario, the application server serves the business logic for looking up a product's pricing information. That functionality doesn't say anything about display or how the client must use the information. Instead, the client and application server send data back and forth. When a client calls the application server's lookup service, the service simply looks up the information and returns it to the client.

By separating the pricing logic from the HTML response-generating code, the pricing logic becomes far more reusable between applications. A second client, such as a cash register, could also call the same service as a clerk checks out a customer. In contrast, in Scenario 1 the pricing lookup service is not reusable because the information is embedded within the HTML page. To summarize, in Scenario 2's model, the Web server handles HTTP requests by replying with an HTML page while the application server serves application logic by processing pricing and availability requests

Caveats

Recently, XML Web services have blurred the line between application servers and Web servers. By passing an XML payload to a Web server, the Web server can now process the data and respond much as application servers have in the past.

Additionally, most application servers also contain a Web server, meaning you can consider a Web server a subset of an application server. While application servers contain Web server functionality, developers rarely deploy application servers in that capacity. Instead, when needed, they often deploy standalone Web servers in tandem with application servers. Such a separation of functionality aids performance (simple Web requests won't impact application server performance), deployment configuration (dedicated Web servers, clustering, and so on), and allows for best-of-breed product selection.

Function

The main function of a web server is keeping files active for web site browsing, twenty-four hours a day, seven days a week. Any time lost is known as down time which means that at that point, the website and its pages will not be viewable. Any good web hosting company tries to keep their downtime to less than a fraction of a second to be successful. An Application server facilitates this process and tries to make for easy data access of an application.

Multi Threading

The Web Server does not support the concept of multi-threading. In Application Server we have features like connection pooling, isolation pooling, multi-threading, and majorly the Transaction feature which is not there in Web Server. Web servers (programs) are supposed to serve requests quickly from more than one TCP/IP connection at a time.

Consider that Internet Explorer or Firefox Web Browser is a local program on the user's hard drive, whereas the web pages themselves are not. The web pages are actually stored on the hard drives of other computers, and these are known as web servers.

Application server products typically bundle middleware to enable applications to intercommunicate with dependent applications, like Web servers, database management systems, and chart programs.

Load Limit

A web server (program) has defined load limits, because it can handle only a limited number of concurrent client connections (usually between 2 and 60,000, by default between 500 and 1,000) per IP address (and IP port) and it can serve only a certain maximum number of requests per second. On the other hand, an application server has a much higher capacity.

Model

Webserver delegation model is fairly simple, when the request comes into the webserver, it simply passes the request to the program best able to handle it (Server side program). It may not support transactions and database connection pooling. Web servers support to deploy .war files only while Application servers support to deploy .war and .ear files.

Application server is more capable of dynamic behaviour than webserver. An application server can be configured to work as a webserver

History

The first web server owes its origin to Tim Berners-Lee when as part of a new project to his employer CERN (European Organization for Nuclear Research). In 1989 he wrote two programs which led to the implementation of the first web server. The Application server first came up in the 1990's.

It can be said that a Web server is a subset of an application server. Application servers and web servers are beginning to blur into each other with the expansion of the Internet and Web 2.0 technologies. In most instances currently, software is hosted on web servers, and then downloaded to the local hard drive, where it is installed on the local computer. In the new model that fuses the web server and application server, the software would be hosted online and the user could access it and use it as needed, generally, at a lower rate than if he or she were to purchase the software new.

Why is Tomcat a Webserver and not an Application Server

Many application developers do not focus much on the infrastructure on which their code runs. When it comes to web applications there are common confusions like what is the difference between webserver and applications server or when to go for a EAR vs WAR file deployment etc.

There are many good answers that differentiate between web servers and applications servers like this one. Most of the times the terms Web Server and Application server are used interchangeably. This article explains the working of a typical web server. Typically we get confused with the example of Tomcat Server (an example for a web server) having the capability to run the enterprise applications. So, tomcat is a web server or an application server? Let me tell you how I convinced my self regarding this.

Some time back I was struck with the question What's the difference between JPA and Hibernate on stack overflow. I did answer it, but one of the comment lead me to a more detailed understanding of the JavaEE spec and certified servers. If you can understand this then differentiating between the web server and application server is easy. During my investigations I got this article, which discusses the advantages of both.

A more detailed look in to the meaning JavaEE specification will throw some light in to our discussions. As we know specifications are set of rules. Simply put they contain the interface. Any JavaEE servers which needs to comply to spec needs to have the implementation of these interfaces. You can find the certified JavaEE servers list here. If you are deploying your enterprise applications (means you have JPA, EJB or some technology which is part of Java EE) to the a server which comply to JavaEE then the lib need not contain the API implementation jars. But these are needed if you are using a web server like tomcat for deployment.

For example, if you use JPA in your applications and deploying it to the Jboss AS 7, then you need any additional jars in the lib. But the same application you want to deploy to the tomcat server then you need to have additional jars to lib that implements the JPA spec may be eclipselink or Hibernate. This is what makes JBoss AS 7 an application server and tomcat a web server. Another key difference is that we can not deploy an EAR file to tomcat, it could only handle WAR files.