1. All Servlet Containers **are** Web Servers
2. Not all Web Servers **are** Servlet Containers. I.E. Apache HTTP Server cannot run Java Servlet directly.

**HTTP is the Hypertext Transport Protocol. Both Apache and Tomcat will accept HTTP requests.**

**The difference is that Apache is JUST an HTTP server. It serves up static HTML pages.**

**Tomcat has an HTTP listener inside it, but in addition to that it has a servlet/JSP engine. It can serve up both static and dynamic HTML pages.**

**You can use Tomcat without Apache.**

**Apache cannot handle servlet/JSP requests without having a servlet/JSP engine bolted on top of it**

While Tomcat is capable of serving both dynamic and static content, it is not as fast or feature-rich as the Apache web server with regard to static content. While it would be possible for Tomcat to be extended to support the same features that Apache does for serving up static content, it would take a great deal of time;.

The earliest versions of Tomcat included a connector that enabled Tomcat and Apache to work together. In this arrangement, Apache receives all of the HTTP requests made to the web application. Apache then recognizes which requests are intended for servlets/JSP pages, and passes these requests to Tomcat. Tomcat fulfils the request and passes the response back to Apache, which then returns the response to the requestor.

--------------------------------------------------------------------------------------------------------------------

The difference between WebServer and web container are as under   
\* The WebServer just receives the request from the client and forward to the WebContainer, and do the Response vice versa.  
  
\* The WebContainer Creates HTTP Request and Response for the Servlet, calls the appropriate servlet service method (doGet or doPost) to service the client request.  
  
3) The Web Container also gives Communication support(socket creation), Servlet life cycle management (Init(), Service() and Destroy()), Multithreading support, Declarative security(thru deployment descriptor) and JSP suppot

What is the benefit of using Application server over WebServer and web container?

Application Server is mainly Used when You Required some extra feature from your Application Like Automatic Transaction management, and Security,   
So If you only want to Deliver some Static Content from Web Application or Want your Servlets, JSPs, JSF to handle the Client Request Just Stick with Web Server which is having Servlet and JSP container,   
But If you want to deploy EJBs which is a Java EE component, and having some different Life cycle methods over Servlets and JSP, so in this Case you have to choose an Application Server.

Web server basically accepts request and gives response for that request in different format like text,jpg,pdf etc.

Some well known JEE servers are JBOSS,WebLogic,WebSphere etc.

A fully compliant JEE application server contains both Web container and EJB container.

Web container is for components like Servlet, JSP etc and EJB container is for EJB.

Web Container is there to support the web components. It provides support like communication support, servlet lifecycle management, multithreading support, declarative security, JSP Support(translation of JSP to real java).

EJB container is there to support the business components like enterprise java beans. It provides support like bean life cycle management, transaction support etc.

Actually only for the container we can concentrate on business logic rather thinking much about the code for threading,security,JSP processing,networking, transaction, bean life cycle etc. These are the things that container does for us . :-)

The Webserver: Just Receive the request from the client and forward to Webcontainer, and do the Response vice versa.  
  
Web Container: Creates HTTP Request and Response for the Servlet, calls the appropriate servlet [service](http://www.geekinterview.com/question_details/64460) method (doGet or doPost) to service the client request.  
  
Container also gives: [Communication](http://www.geekinterview.com/question_details/64460) support(socket creation), Servlet life cycle management (Init(), Service() and Destroy()), Multithreading support, Declarative security(thru deployment descriptor) and  JSP support.

An **Application Server** is a complete server, which provides an environment for running the business components (EJBs, ADF BCs, etc.) in addition to providing the capabilities of a Web Container as well as of a Web Server

A **Web Container** is a implementation which provides an environment for the Servlets and JSPs to run. Putting it differently we can say that a Web Container is combination of a Servlet Engine and a JSP Engine. If an HTTP Request refers to a Web Component (typically a Servlet or a JSP) then the request is forwarded to the Web Container and the result of the request is sent back to Web Server, which uses that result to prepare the HTTP Response for the particular HTTP Request. **Example:** Tomcat is a typical Web Container. A typical setup would be to have Apache HTTP Server as the Web Server and Tomcat as the Web Container.

A **Web Server** is a server capable of receiving HTTP requests, interpreting them, processing the corresponding HTTP Responses and sending them to the appropriate clients (Web Browsers).

**Tomcat is a**[**web server**](http://en.wikipedia.org/wiki/Web_server)**(can handle HTTP requests/responses) and**[**web container**](http://en.wikipedia.org/wiki/Web_container)**(implements**[**Java Servlet API**](http://jcp.org/aboutJava/communityprocess/mrel/jsr154/index2.html)**, also called servletcontainer) in one. Some may call it an**[**application server**](http://en.wikipedia.org/wiki/Application_server)**, but it is definitely not an fullfledged Java EE application server (it does not implement**[**the whole Java EE API**](http://java.sun.com/javaee/5/docs/api/overview-summary.html)**.**

Tomcat is a web server and a Servlet/JavaServer Pages container. It is often used as an application server for strictly web-based applications but does not include the entire suite of capabilities that a Java EE application server would supply.

Application server also contains the web server.

A Web application runs within a Web container of a Web server. The Web container provides the runtime environment through components that provide naming context and life cycle management. Some Web servers may also provide additional services such as security and concurrency control. A Web server may work with an EJB server to provide some of those services. A Web server, however, does not need to be located on the same machine as an EJB server.

Web applications are composed of web components and other data such as HTML pages. Web components can be servlets, JSP pages created with the JavaServer Pages™ technology, web filters, and web event listeners. These components typically execute in a web server and may respond to HTTP requests from web clients. Servlets, JSP pages, and filters may be used to generate HTML pages that are an application’s user interface. They may also be used to generate XML or other format data that is consumed by other application components.

***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.

**What is the difference between Application Server and Web Server**

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.

1. Application server are more heavy than web server in terms of resource utilization.

WEB SERVER

1. There are no domains in web server

2. we can not ron two projects with diff server settings in a web server .

3. A web server only provides web container .

4.  We can only deploy war file in a web server .

5. A web server only accepts http or https protocol requests  .

6. A web server can provide less middle ware services to the application .

 APPLICATION SERVER

1. there are domains in application server

2. we can run two projects with dif server settings at a time in a application server

3.  An application server provides web container and EJB container also .

4.  We can deploy jar, war, ear files in an application server .

5.  Application server accepts multiple protocol requests .

6.  An app server can provide more middle ware services to the application

most of the production environments have web server acting as reverse proxy to app server. That means while service a page request, static contents such as images/Static html is served by web server that interprets the request. Using some kind of filtering technique (mostly extension of requested resource) web server identifies dynamic content request and transparently forwards to app server

Example of such configuration is Apache HTTP Server and BEA WebLogic Server. Apache HTTP Server is Web Server and BEA WebLogic is Application Server.

* An application server is typically a super-set of a web server, which is to say that an application server can do what a web server does (i.e. answer HTTP socket requests on port 80 and HTTPS socket requests on port 443).

Typically a web server is software which listens to some TCP port (typically 80). It can receive an HTTP request and is capable of responding with an HTTP response. The HTTP response could be generated from a static file, or it could be generated by a script or program.  
  
Within the context of Java, a web server is typically a Java process which is capable of delegating the processing of a request to Servlets and JSP's. An application server on the other hand does everything a web server can do and more. Among the other things an application server does is support the JavaEE specification by being able to support EJB's, JPA, etc.  
  
For example Tomcat is a web server, and JBoss is an application server.