(Web App)
A web application (web app) is an application that users run through their browsers. Common examples of web applications are webmails (e.g. Google Mail, Yahoo Mail), online stores (e.g. Lazada) and social media sites (e.g. Facebook, Instagram, Twitter).

Users run the web app by entering the domain name (e.g. facebook.com , gmail.com). When this happens, the browser connects to the server that is hosting the application and display whatever information it has received from the server.

Web application grew in popularity because of the benefits it provides to both customers and service providers. For service providers, web applications allows central management of the application, easier and much cheaper upgrades, hardware independent among others. For customers, they can check out the services and avail it even when they are at their home.

Most web apps, if not all, focuses on a web application that provides the best possible user interaction and user experience which eventually leads to more revenue. To make this happen, service providers thought that web apps must have a dynamic content.

(History)
Applets, one of the earliest attempts toward this goal, focused on using the client platform to deliver dynamic user experiences. At the same time, developers also investigated using the server platform for this purpose. Initially, Common Gateway Interface (CGI) scripts were the main technology used to generate dynamic content. Although widely used, CGI scripting technology has a number of shortcomings, including platform dependence and lack of scalability. To address these limitations, Java Servlet technology was created as a portable way to provide dynamic, user-oriented content. (https://docs.oracle.com/javaee/5/tutorial/doc/bnafd.html)

Servlet (version 4.0)

A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes. (https://docs.oracle.com/javaee/5/tutorial/doc/bnafe.html)

The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. All servlets must implement the Servlet interface, which defines life-cycle methods. When implementing a generic service, you can use or extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet and doPost, for handling HTTP-specific services. (https://docs.oracle.com/javaee/5/tutorial/doc/bnafe.html)

// web app structure

//sample servlet code

download file functionality :
Java Servelet Specification : http://download.oracle.com/otndocs/jcp/servlet-4-final-spec/index.html)

Servlet Life Cycle

Servlets, which are managed by a container, are used as web components to generate dynamic content. One function of containers (a.k.a servlet engines) is to manage the life cycle of a servlet.

> *//Example : better if may actual code : may be improved . master web servlets first*
>
> Reference : Java™ Servlet Specification version 4.0 (Version 4.0, Shing Wai Chan, Ed Burns, July 2017)
> When a browser sends an HTTP request to the Web Server, the Web Server will send the request to the servlet container. From there, the servlet container will determine which servlet to invoke. When the servlet is done in processing the request, the servlet container ensures that response is properly flushed and then returns the control back to the Web server.

Servlet Life Cycle Java Servlet 3.1 (Java EE 7)

The life cycle of a servlet has 4 stages:

1. ~ instantiation >> constructor call
    the constructor call
        -creates an instance of the servlet to service client requests
        -invoked implicitly by the servlet container when the servlet is called upon to service client request and no instance currently exists.
2. ~ initialization >> init() method
    invoked only once intended for any startup init code
3. ~ request handling >> service() method
    invoked for each client request
    performs whatever logic to serve request and generate response
    multi-threaded servlet instance; 'thread safe'
4. ~ destruction >> destroy() method
    invoked before servlet instance is 'unloaded', housekeeping cleanup code.

Java Servlets

- handles client requests by generating responses to such requests.
- applications implemented on web servers.
- part of the Java Enterprise Edition (EE), which are used for transaction-based applications.
- hosted in a servlet container that provides the environment which the servlet runs, as well as controls the 'servlet lifecycle'.

HTTP Servlets
- handles HTTP request and generates responses
- hosted in a web container, that takes cares of HTTP Requests conversion to HTTP Responses
- service() method call is routed to a doXXX() call
    doXXX calls are: doGet,doPost,doPut,doDelete,doTrace,doOptions,doHead
    the doXXX call requires two arguments
        1. HTTPServletRequest
        2. HTTPServletResponse