WHAT ARE SERVLETS?

Java Servlets are programs that run on a Web or Application server and act as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

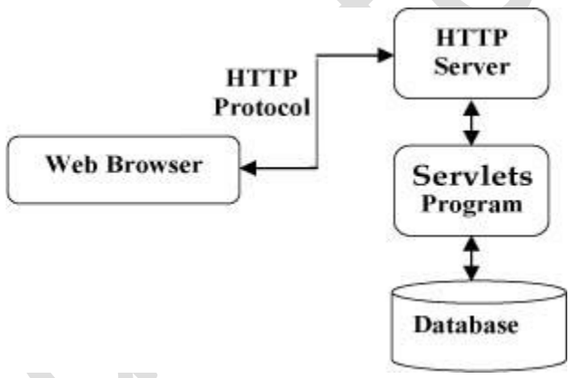
Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

Java Servlets often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But Servlets offer several advantages in comparison with the CGI.

- Performance is significantly better.
- Servlets execute within the address space of a Web server. It is not necessary to create a separate process to handle each client request.
- Servlets are platform-independent because they are written in Java.
- Java security manager on the server enforces a set of restrictions to protect the resources on a server machine. So servlets are trusted.
- The full functionality of the Java class libraries is available to a servlet. It can communicate with applets, databases, or other software via the sockets and RMI mechanisms that you have seen already.

SERVLETS ARCHITECTURE:

Following diagram shows the position of Servelts in a Web Application.



SERVLETS TASKS:

Servlets perform the following major tasks:

- Read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.
- Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.
- Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.
- Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.
- Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

SERVLETS PACKAGES:

Java Servlets are Java classes run by a web server that has an interpreter that supports the Java Servlet specification.

Servlets can be created using the **javax.servlet** and **javax.servlet.http** packages, which are a standard part of the Java's enterprise edition, an expanded version of the Java class library that supports large-scale development projects.

A development environment is where you would develop your Servlet, test them and finally run them.

Like any other Java program, you need to compile a Servlet by using the Java compiler **javac** and after compilation the Servlet application, it would be deployed in a configured environment to test and run.

This development environment setup involves following steps:

SETTING UP JAVA DEVELOPMENT KIT

This step involves downloading an implementation of the Java Software Development Kit (SDK) and setting up PATH environment variable appropriately.

You can download SDK from Oracle's Java site: Java SE Downloads.

Once you download your Java implementation, follow the given instructions to install and configure the setup. Finally set PATH and JAVA_HOME environment variables to refer to the directory that contains java and javac, typically java_install_dir/bin and java_install_dir respectively.

If you are running Windows and installed the SDK in C:\jdk1.5.0_20, you would put the following line in your C:\autoexec.bat file.

set PATH=C:\jdk1.5.0_20\bin;%PATH%

set JAVA_HOME=C:\jdk1.5.0_20

Alternatively, on Windows NT/2000/XP, you could also right-click on My Computer, select Properties, then Advanced, then Environment Variables. Then, you would update the PATH value and press the OK button.

SETTING UP WEB SERVER: TOMCAT

A number of Web Servers that support Servlets are available in the market. Some web servers are freely downloadable and Tomcat is one of them.

Apache Tomcat is an open source software implementation of the Java Servlet and Java Server Pages technologies and can act as a standalone server for testing Servlets and can be integrated with the Apache Web Server. Here are the steps to setup Tomcat on your machine:

Download latest version of Tomcat from http://tomcat.apache.org/.

Once you downloaded the installation, unpack the binary distribution into a convenient location. For example in C:\apache-tomcat-5.5.29 on windows and create CATALINA_HOME environment variable pointing to these locations.

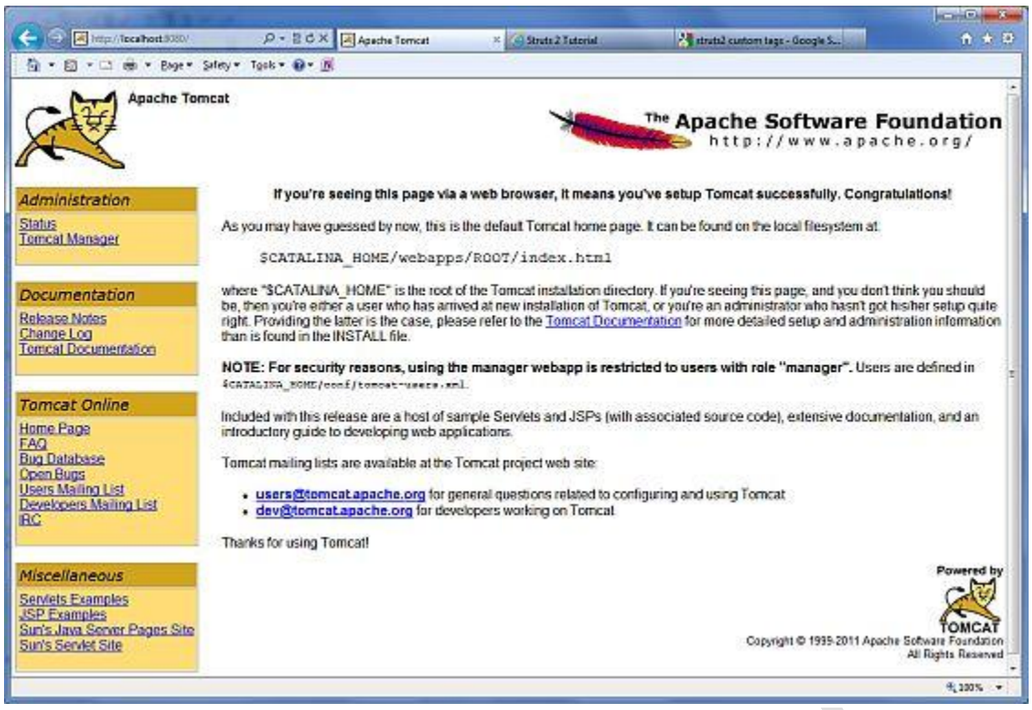
Tomcat can be started by executing the following commands on windows machine:

```
%CATALINA_HOME%\bin\startup.bat

or

C:\apache-tomcat-5.5.29\bin\startup.bat
```

After startup, the default web applications included with Tomcat will be available by visitinghttp://localhost:8080/. If everything is fine then it should display following result:



Further information about configuring and running Tomcat can be found in the documentation included here, as well as on the Tomcat web site: *http://tomcat.apache.org*

Tomcat can be stopped by executing the following commands on windows machine:

```
C:\apache-tomcat-5.5.29\bin\shutdown
```

SETTING UP CLASSPATH

Since Servlets are not part of the Java Platform, Standard Edition, you must identify the Servlet classes to the compiler. If you are running Windows, you need to put the following lines in your C:\autoexec.bat file.

```
set CATALINA=C:\apache-tomcat-5.5.29
set CLASSPATH=%CATALINA%\common\lib\servlet-api.jar;%CLASSPATH%
```

Alternatively, on Windows NT/2000/XP, you could also right-click on My Computer, select Properties, then Advanced, then Environment Variables. Then, you would update the CLASSPATH value and press the OK button.

A Servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a Servlet

- The Servlet is initialized by calling the **init ()** method.
- The Servlet calls **service()** method to process a client's request.
- The Servlet is terminated by calling the **destroy()** method.
- Finally, Servlet is garbage collected by the garbage collector of the JVM.

DRAWBACKS OF SERVLETS

1. *In many Java Servlet-based applications, processing the request and generating the response are both handled by a single Servlet class.*
2. *Thorough Java programming knowledge is needed to develop and maintain all aspects of the application, since the processing code and the HTML elements are lumped together.*
3. *Changing the look and feel of the application, or adding support for a new type of client, requires the Servlet code to be updated and recompiled*
4. *To develop the web page layout, the generated HTML must be manually embedded into the Servlet code, a process which is time consuming, error prone, and extremely boring.*