Servlet

Need of Web Application

The application that is specific for one computer and allows one user at a time to operate the application is called **STANDALONE APPLICATION.** *E.g. All Core java application*

In standalone application, data & logics are specific to one computer.

To provide global visibility, accessibility to data & logics of application, we need to develop web application/websites and run in the internet.

WEB APPLICATION:

Web application is a collection web component having the capability to generate web pages. There are three dominating environment to develop web sites:

- PHP **(Small Scale)**
- Asp.net **(Medium Scale)**
- Java [JEE] **(Large Scale)**

Based on content there are two types of web pages:

- **STATIC WEB PAGE:** display fixed content for all requests e.g. *About Us, Terms & Conditions, Contact Us etc.*
- **DYNAMIC WEB PAGE:** The content of web page change based on time of request generation or based on input values of the request e.g. *Live Game Score Page, gmail inbox page etc.*

Based on content of the web pages there are two types of web resources components

- **STATIC WEB COMPONENT:** Generate static web pages e.g. *HTML Pages*
- **DYNAMIC WEB COMPONENT:** Generate the dynamic web pages e.g. servlet, JSP, asp.net, php component.

Note: Every web application is the collection of multiple web components and other files e.g. *images, audio, video, js, files etc*.

Need of Web Application

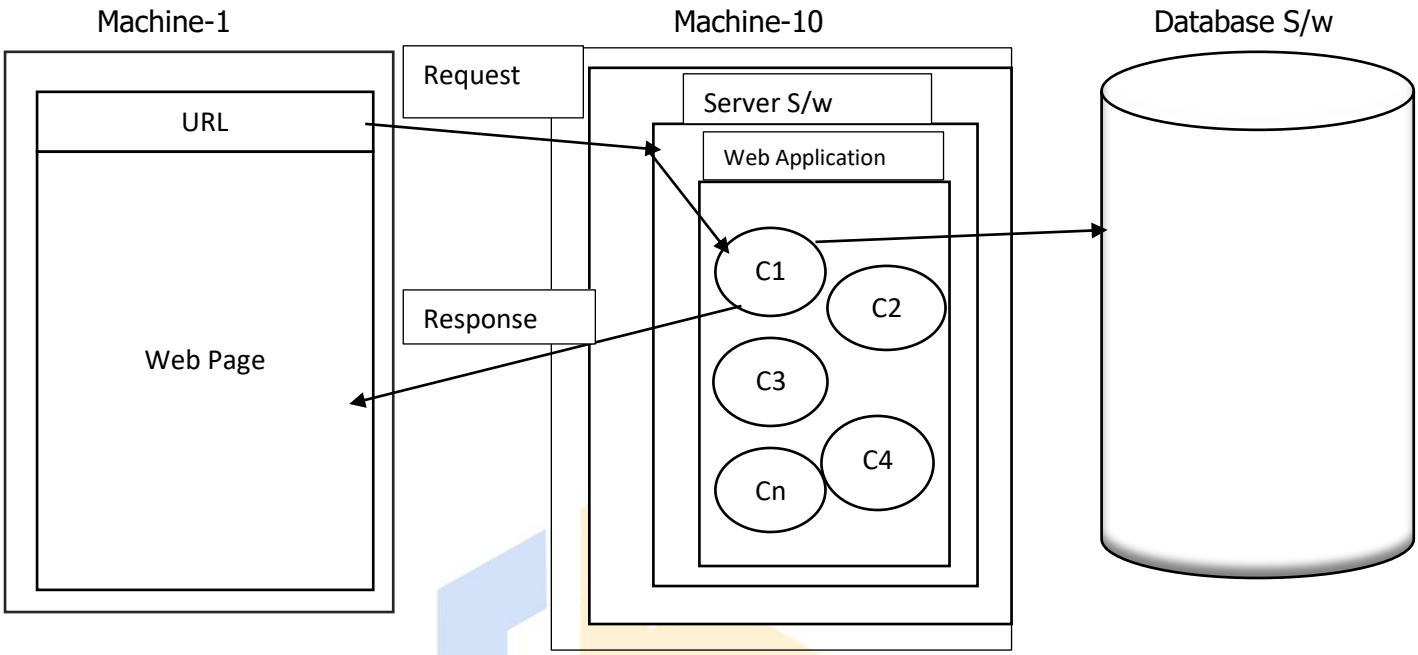
The execution of standalone application takes place only when we initiate the execution.

Then web component of web application will be executed dynamically the moment they are requests them, so we can think about human environment in the execution. Web component, web component should be executed **AUTOMATICALLY & DYNAMICALLY IN AUTOMATED ENVIRONMENT**, the moment they are requested from browser for this automation we need special software called **Web Server Software.**

Web Server:

It is a special software that listen the client request continuously, takes the request from clients, passes the request to the appropriate web component, executes the web component dynamically and sends the output the browser as response in form of web pages. E.g. *Apache Tomcat, IIS etc.*

Servlet



Based on place where the web component executes there are two types of web components:

- **Client Side Component:** This web component goes to browser from the web application of web server for execution when requested e.g. HTML components.
- **Server Side Component:** This web component executes in web server itself when requested e.g. servlet, jsp, php etc.

****

Note:

Beside weather web component is client side or server side base on the place where it executes not based on where it resides

Developing web server software is not the responsibility of programmers, vendor companies will supply them, but developing web application having web components is the responsibility of programmers.

All java based web server and application server provide built-in **Servlet Container (Catlina), JSP Container (JASPER).**

Container is software application or software that take care the whole life cycle of given resources/components (Birth-to-Death)/ object creation to object destruction. It is like an accureium for fishes.

Servlet

Understanding the Tomcat Server:

Type: Java Based

Version: 9.x (Compatible with JDK 1.8 and above)

Vendor: Apache Foundation (Open Source)

Default Port: 8080

To download server software: www.apache.org

Servlet Container: Catalina

JSP Container: JASPER

While installing Tomcat software:

1. Choose <java_home>/jre location
2. Tomcat installation folder
3. Default port no. 8080
4. Configure username and password
5. To start Tomcat server use <tomcat_home>\bin\Tomcat9.exe file
6. To see the home page open browser and type url http://<server-address>:portNo (http://localhost:8080)

To change the http port no of tomcat after installation:

<tomcat_home>\conf\server.xml and modify port attribute value of first <connector port= "8080"> and restart the server.

We can deploy java web application either in form of directories or in the form of war files in <tomcat_home>\webapps folder.

Servlet-api.jar: Represents servlet api.

Jsp-api.jar: represents jsp api.

Catalina.jar: represents servlet container.

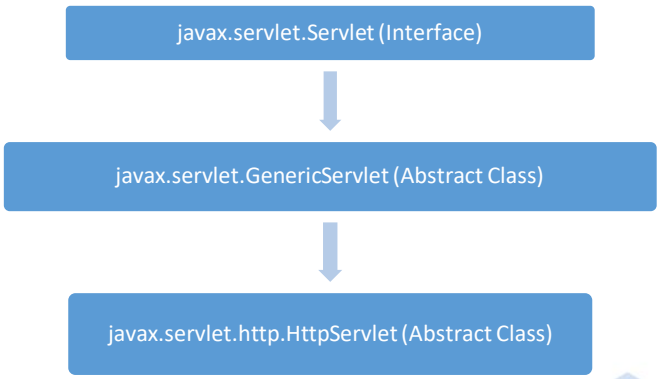
Jasper.jar: represents jsp container.

Definitions of servlet:

- Servlet is jee module server-side web technology that can be used to develop dynamic component having the capability to generate dynamic web pages.
- Servlet is jee technology which can be used enhance the functionalities of *WEB SERVER* or *HTTP SERVER* or *APPLICATION SERVER* .
- Servlet is jee technology/specification that provides companies to develop servlet container and also provides api for programmers to develop java server-side web components of web application.

Servlet

Three important resources of servlet-api:



javax.servlet.Servlet contains five prototypes that are following:

1. public void init(ServletConfig sc) throws ServletException;
2. public ServletConfig getServletConfig();
3. public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException;
4. public String getServletInfo();
5. public void destroy();

javax.servlet.GenericServlet have following functions:

1. public void destroy() {}
2. public String getInitParameter(String name) { }
3. public Enumeration getInitParameterNames() { }
4. public ServletConfig getServletConfig() {}
5. public ServletContext getServletContext() {}
6. public String getServletInfo() {}
7. public void init(ServletConfig config) throws ServletException { }
8. public void init() throws ServletException {}
9. public void log(String msg) {}
10. public void log(String message, Throwable t) {}
11. public abstract void service(ServletRequest req, ServletResponse res) throws ServletException, IOException;
12. public String getServletName() {}

javax.servlet.http.HttpServlet have following functions and varibales:

```
private static final String METHOD_DELETE = "DELETE";

private static final String METHOD_HEAD = "HEAD";

private static final String METHOD_GET = "GET";

private static final String METHOD_OPTIONS = "OPTIONS";

private static final String METHOD_POST = "POST";

private static final String METHOD_PUT = "PUT";

private static final String METHOD_TRACE = "TRACE";

private static final String HEADER_IFMODSINCE = "If-Modified-Since";

private static final String HEADER_LASTMOD = "Last-Modified";

private static final String LSTRING_FILE = "javax.servlet.http.LocalStrings";
```

Servlet

private static ResourceBundle lStrings;

1. public HttpServlet() {}
2. protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {}
3. protected long getLastModified(HttpServletRequest req) {}
4. protected void doHead(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {}
5. protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {}
6. protected void doPut(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {}
7. protected void doDelete(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {}
8. private static Method[] getAllDeclaredMethods(Class c) {}
9. protected void doOptions(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {}
10. protected void doTrace(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {}
11. protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {}
12. private void maybeSetLastModified(HttpServletResponse resp, long lastModified) {}
13. public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {}

Note:

every servlet component is a java class that uses servlet-api. This must be implement **javax.servlet.Servlet** interface directly or indirectly. There are three approaches to develop servlet component.

Take a class which implement javax.servlet.Servlet Interface and provide definition for all five methods.

Example:

```
Import javax.servlet.Servlet;
public class <class_name> implements Servlet{
}
```

Take a class extending from javax.servlet.GenericServlet and provide implementation for service(req, res).

```
public class <class_name> implements Servlet{
    public void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    }
}
```

Note: Not a recommended approach because it does not allow programming to work with protocol http features like ending auto-refresh on web page.

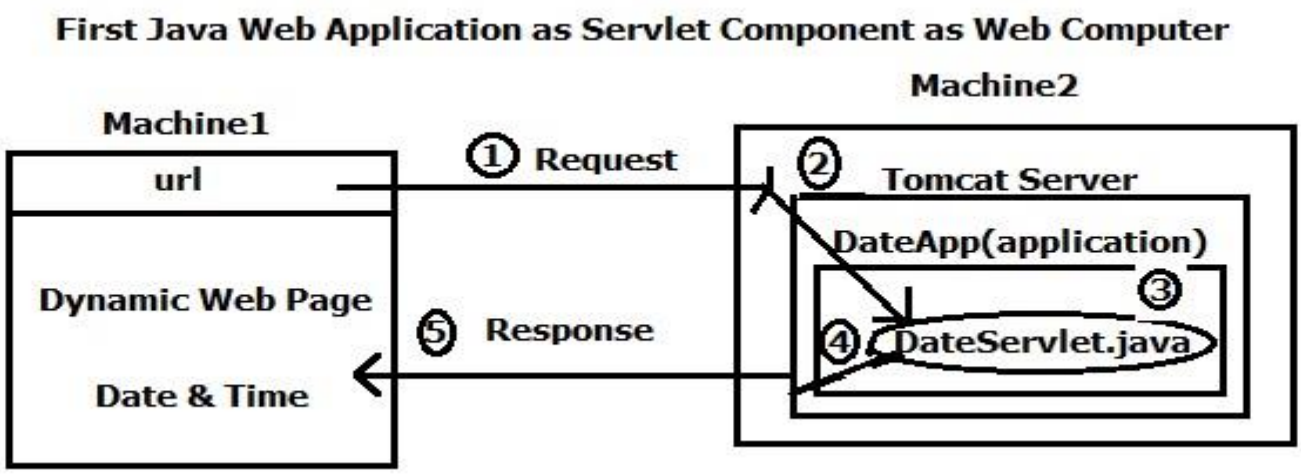
Take a class extending from javax.servlet.http.HttpServlet and override one of 2 service(-,-) or orverride one the doxxx(-,-)

```
public class <class_name> implements HttpServlet{
    public void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    }
```

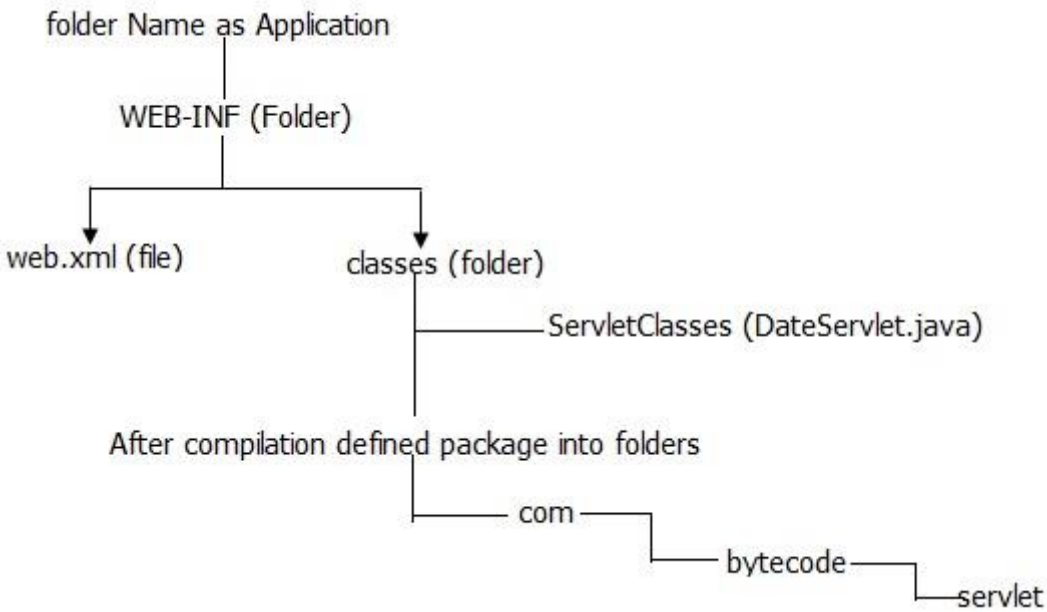
Servlet

}

Note: A recommended approach because it does not allow programming to work with protocol http features like ending auto-refresh on web page.



Step 1: Create deployment directory structure



The above directory structure represent the web application that is deployable in the server so it is called **DEPLOYMENT DIRECTORY STRUCTURE.** This structure is common for all servers. It is designed by **Sun Microsystem (M.S.)** and followed by all vendor companies who creates web server software. This directory structure specifies the arrangement and packing of web component into web application.

WEB-INF/classes folder given to place java classes like servlet components. We place html/jsp component outside the **WEB-INF** folder and web.xml file contains various configuration like servlet component configuration, homepage configuration etc...

Providing detail about creation file or web component to make underlying server or container recognizing that file or component as special component is called **CONFIGURATION FILE.** Such configuration in web.xml file. Therefore, web.xml is called **WEB APPLICATION CONFIGURATION FILE**. This file contains servlet configuration, jsp configuration etc…

Step 2: Develop Servlet Component

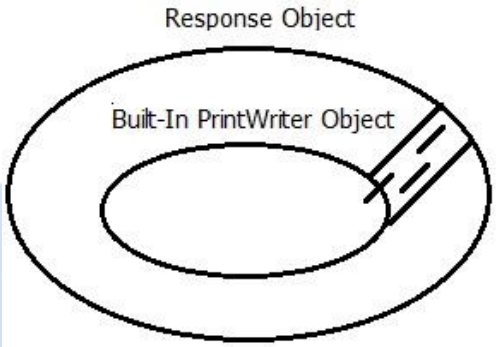
Servlet

To make our servlet component classes visible to servlet container for instantiating we need to take our servlet component class as ***public class.*** Similarly service (-,-) should be taken as public method to make that method invoke- able from servlet container.

res.setContentType("text/html");

gives instruction to browser through web server to display received output content as html code base text document (Web Page) on browser window.

PrintWriter pw = res.getWriter(); Response object contains one built-in stream **printWriter.** res.getWriter() returns that stream to servlet component. This stream points to response object as the destination object to write data.



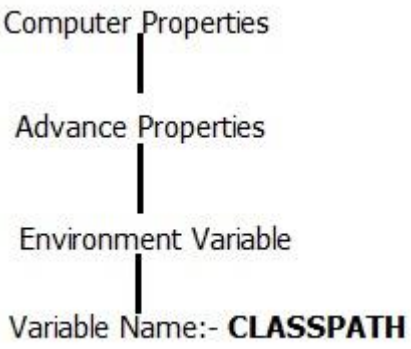
Stream is container flow of data, which can have file, object, or array as destination to write data and read data.

pw.println("Hello World"); This println() writes given message to response object. The response object writes given message to web server through servlet container web server writes given message to browser window as the web page content.

pw.close(); Closes the stream that is associated with response object that is does not allow to write future content to response object using that stream

Note: For every request coming to servlet component the servlet container automatically call service(-,-) having request, response object as arguments.

Step 3: Add <Tomcat_Home>/lib/servlet-api.jar file to classpath environment variable.



Note: servlet-api is not the part of **jdk**, but it is available in <Tomcat_Home>/lib folder. So we need to add that jar file to classpath.

Step 4: Compile the servlet component

Command in cmd: javac –d . className.java

Servlet

For run this command go to drive:/applicationName/WEB-INF/classes

Step 5: Configuration component in web.xml file

Servlet component configuration is nothing but placing the details of servlet component in web.xml file to make servlet container recognizing the given java class as servlet component. Web server reads web.xml file the moment web deploy the web application so web.xml file can be called as **DEPLOYMENT DESCRIPTOR**. Web.xml contains lot of configuration related to web application so this file is also called as **WEB APPLICATION CONFIGURATION FILE**. Servlet component is identified with its **<url-pattern>,** not with its servlet name and logical name.

Note: to test weather web.xml file is correct file or not press **Ctrl+B** in Editplus application.

Step 1 to 5 development of java web application

Step 6: Start Tomcat Server use <Tomcat_Home>/bin/Tomcat9.exe.

Step 7: Deploy the web application <Tomcat_Home>/webapps/(To deploy web application)

Step 8: Test the web application Request url in the web browser

http://serverName:port_Number/applicationName/logicalName

for reloading in browser <Tomcat_home> (http://serverName:prot_Number)

Manager app (authentication window open)

Reload/Deploy

Knowing username and password: After installation in **Tomcat Server** goto <Tomcat_Home>/conf/Tomcat-User.xml and observe first user tab.

- Using url-pattern, we can hide servlet class name, technology name from visitor of the websites. This helps to improve the security of web application towards getting protection from hackers.
- Underlying server will recognize the modifiers done in web.xml automatically because the server internally reloads the web application.
- System.out.println("") message placed in servlet component will be display on server console as debug message.
- pw.println() sends the message to servlet component but those messages will be displayed on browser window as web page content.
- Servlet container uses 0(zero) parameter constructor while instantiating to our servlet class (Object Creation). So make sure that 0(zero) parameter constructor is available in our servlet component directly or indirectly.
- For every request given to servlet component, the servlet container creates one thread and one set of request, response object & this object will be destroyed automatically once the request related response goes to browser window.

Understanding Flow of Execution:

From request arrival to response generation with support of above example. If xml documentation satisfied syntax and rule, then it is call **Well Formed Document.** If xml documentation satisfying .dtd/.xsd rules (about tags and attributes) then it is called valid document.

.dtd (Document Type Definition) **.xsd (XML Schema Definition)**

When web application deploy the server/container locate web.xml file and checks weather it is well formed, valid document or not and also reads web.xml file. Web.xml file to contain to maintain that content as in memory metadata, container uses this in memory metadata to get configuration detail of web.xml file without opening that files multiple times.

a) Programmers deploy application to web application in **Tomcat Server.**
b) Server or container locate web.xml file and checks weather web.xml file is well-formed, valid document or not.

c) Creates in memory metadata having web.xml file content in the memory where server runs.
d) End user type request url in the web browser window to generate request to servlet component that is DateServlet.java
    a. http://localhost:8080/DateApp/test
e) Based on localhost:8080 of request url the tomcat web server takes request.
f) Tomcat server passed the request to DateApp web application on request url
g) Servlet component can give instruction to browser through web server to display web page in different format by specifying the MIME-Type. As response content type and display the output as response in the form of web page.

MIME- Multipurpose Internet Mail Instruction

We can specify this MIME Type by using **res.setContentType("text/html");** Various MIME type are following:

- **ms-word:** application/msword
- **ms-excel:** application/vnd.ms-excel
- **xml:** text/html

To know the **MIME Types** in windows Operating System we can use regedit tool