

HTML Language with Java

Module Name: **Programming with Java & HTML**

Module Code: **CST 06102**

Instructors: **Jamila L.**



Contents

1. HTML and its use in creation and visualization of web sites
2. Steps to use HTML to create web page(s)
3. Use HTML to create simple web pages and web applications
4. Design and implement server side programs

HTML By definition

HTML is the standard markup language for creating Web pages.

- A markup language is a collection of symbols and special terms used to specify the contents of a document.
- HTML is one of several different types of markup languages
- HTML stands for Hyper Text Markup Language
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML editor can be a Note pad, Eclipse, VSCode, Visual Studio etc

HTML Example

```
<!DOCTYPE html>
<html>
   <head>
      <title>Page Title</title>
   </head>
   <body>

      <h1>My First Heading</h1>
      <p>My first paragraph.</p>

   </body>
</html>
```

HTML Example Description

- The <!DOCTYPE html> declaration defines that this document is an HTML5 document
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the HTML page
- The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The <h1> element defines a large heading
- The <p> element defines a paragraph

Steps to use HTML to create web page(s)

Step 1: Basic HTML Page

Step 2: Creating Page Content.

Step 3: Create major elements in the layout. ...

Step 4: Create the HTML content. ...

Step 5: Create CSS for the layout. ...

Step 6: Create CSS to style individual elements. ...

Step 7: Add background color and style.

Use HTML to create simple web pages and web applications

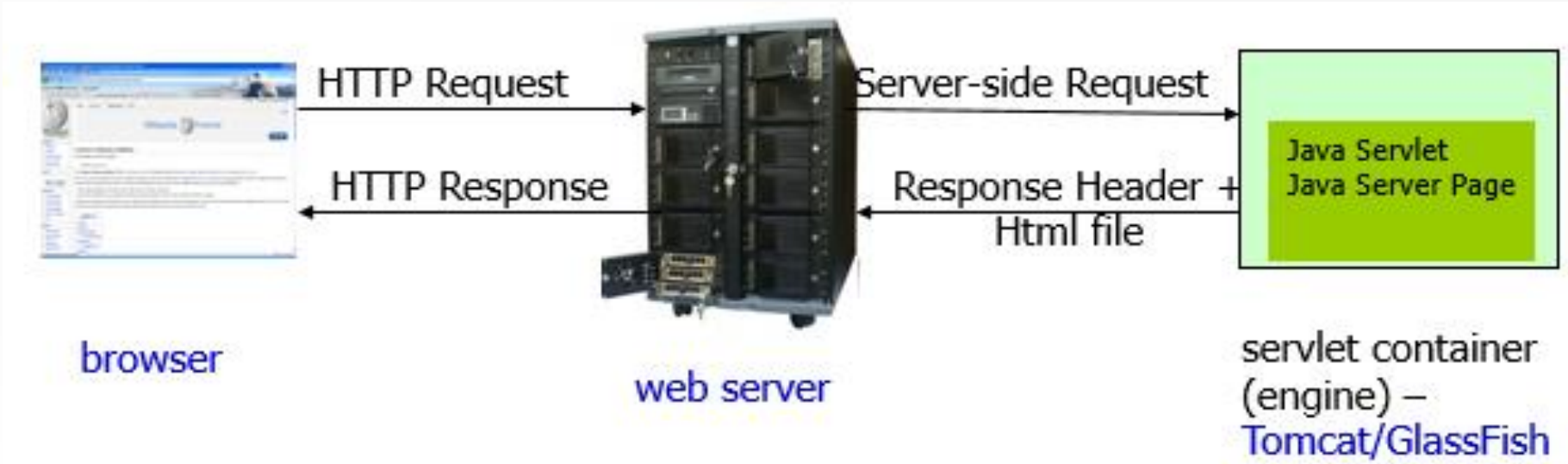
*** Assignment

Design and implement server side programs

Servlet

A servlet is a Java program which outputs an html page; it is a server-side technology.

Servlet technology is used to create a web application (resides at server side and generates a dynamic web page). Servlet technology is robust and scalable because of java language.



Servlet

Servlet Terminology	Description
Website: static vs dynamic	It is a collection of related web pages that may contain text, images, audio and video.
HTTP	It is the data communication protocol used to establish communication between client and server.
HTTP Requests	It is the request send by the computer to a web server that contains all sorts of potentially interesting information.
Get vs Post	It gives the difference between GET and POST request.
Container	It is used in java for dynamically generating the web pages on the server side.
Server: Web vs Application	It is used to manage the network resources and for running the program or software that provides services.
Content Type	It is HTTP header that provides the description about what are you sending to the browser.

Servlet

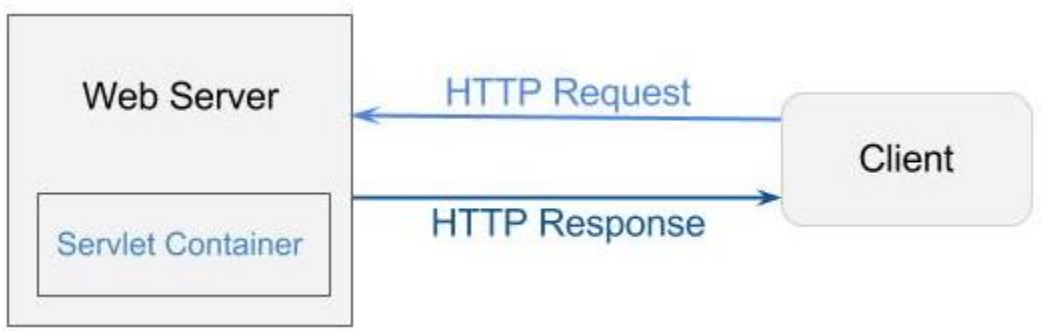
- Servlet is a Java class developed by Sun Microsystems.
- It extends the capability of a server and provides a dynamic content.
- In order to run and deploy Servlet and JSP, a compatible web server and web container (Servlet container) is required like Apache Tomcat, GlassFish, Jetty, etc.
- A Java servlet is a Java class that extends HTTPServlet class.
- A Java servlet class implements the doGet(), doPost() or other equivalent HTTP methods and (usually) prints at the standard output an HTML file.

Servlet cont...

- A Java servlet class can contain any kind of Java code the JDK can compile.
- A Java Servlet class needs to be compiled prior to using it.
- The purpose of a servlet is to create a Web page in response to a client request.
- Servlets are written in Java, with a little HTML mixed in.
- Servlet = Java containing HTML
  - The HTML is enclosed in out.println( ) statements

Servlet Container

- It provides the runtime environment for JavaEE (j2ee) applications. The client/user can request only a static WebPages from the server. If the user wants to read the web pages as per input then the servlet container is used in java.
- The servlet container is the part of web server which can be run in a separate process.



Server: Web vs. Application

- Server is a device or a computer program that accepts and responds to the request made by other program, known as client. It is used to manage the network resources and for running the program or software that provides services.

There are two types of servers:

1. Web Server:
2. Application Server

Server: Web vs. Application

There are two types of servers:

i. Web Server:

It contains only web or servlet container. It can be used for servlet, jsp, struts, jsf etc. It is a computer where the web content can be stored. In general web server can be used to host the web sites but there also used some other web servers also such as FTP, email, storage, gaming

ii. Application Server

It can be used for servlet, jsp, struts, jsf, ejb etc. It is a component based product that lies in the middle-tier of a server centric architecture.

The Example of Application Servers are:

- JBoss: Open-source server from JBoss community.
- Glassfish: Provided by Sun Microsystem. Now acquired by Oracle.
- Weblogic: Provided by Oracle. It more secured.
- Websphere: Provided by IBM.

Servlet API

- The javax.servlet and javax.servlet.http packages represent interfaces and classes for servlet api.
- The javax.servlet package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.
- The javax.servlet.http package contains interfaces and classes that are responsible for http requests only.
- Let's see what are the interfaces of javax.servlet package

javax.servlet Interface

There are many interfaces in javax.servlet package. They are as follows:

i. Servlet
ii. ServletRequest
iii. ServletResponse
iv. RequestDispatcher
v. ServletConfig
vi. ServletContext
vii. SingleThreadModel
viii. Filter
ix. FilterConfig
x. FilterChain
xi. ServletRequestListener
xii. ServletRequestAttributeListener
xiii. ServletContextListener
xiv. ServletContextAttributeListener

Classes in javax.servlet package

There are many classes in javax.servlet package. They are as follows:

i. GenericServlet
ii. ServletInputStream
iii. ServletOutputStream
iv. ServletRequestWrapper
v. ServletResponseWrapper
vi. ServletRequestEvent
vii. ServletContextEvent
viii. ServletRequestAttributeEvent
ix. ServletContextAttributeEvent
x. ServletException
xi. UnavailableException

Servlet Example

```
//first Java Servlet
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class FirstServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>First servlet</title>");
        out.println("</head>");
         out.println("<body>");
          out.println("It works...");
          out.println("</hr>");
                out.println("</body>");
                out.println("</html>");
    }
}
```

Apache Tomcat

- Apache Tomcat is the most well known Servlet/JSP container.
- It is a web server + implementation of Java Servlet and JSP APIs
- It is developed by Apache Software Foundation.
- It is available at http://tomcat.apache.org/ under Apache Software License
- Download the latest binary installer distribution (i.e., 32-bit/64-bit Windows Service installer) from download page into a local folder.
- Double click the Tomcat installer and proceed with dialogs that pop up by clicking yes/next until the installation process completes.

Apache Tomcat Configuration

- Open the file named tomcat-users.xml which is contained in C:\Program Files (x86)\Apache Software Foundation\Tomcat 8.0\conf
- The path depends on the location you selected during the installation process.
- After opening the file, add the user and password at the end of the root tag </tomcat-users>.
- E.g., <user roles="manager-script,admin" username="kmbise" password="Thehill2004" />
- Save the file and close it.

Tomcat standard folders

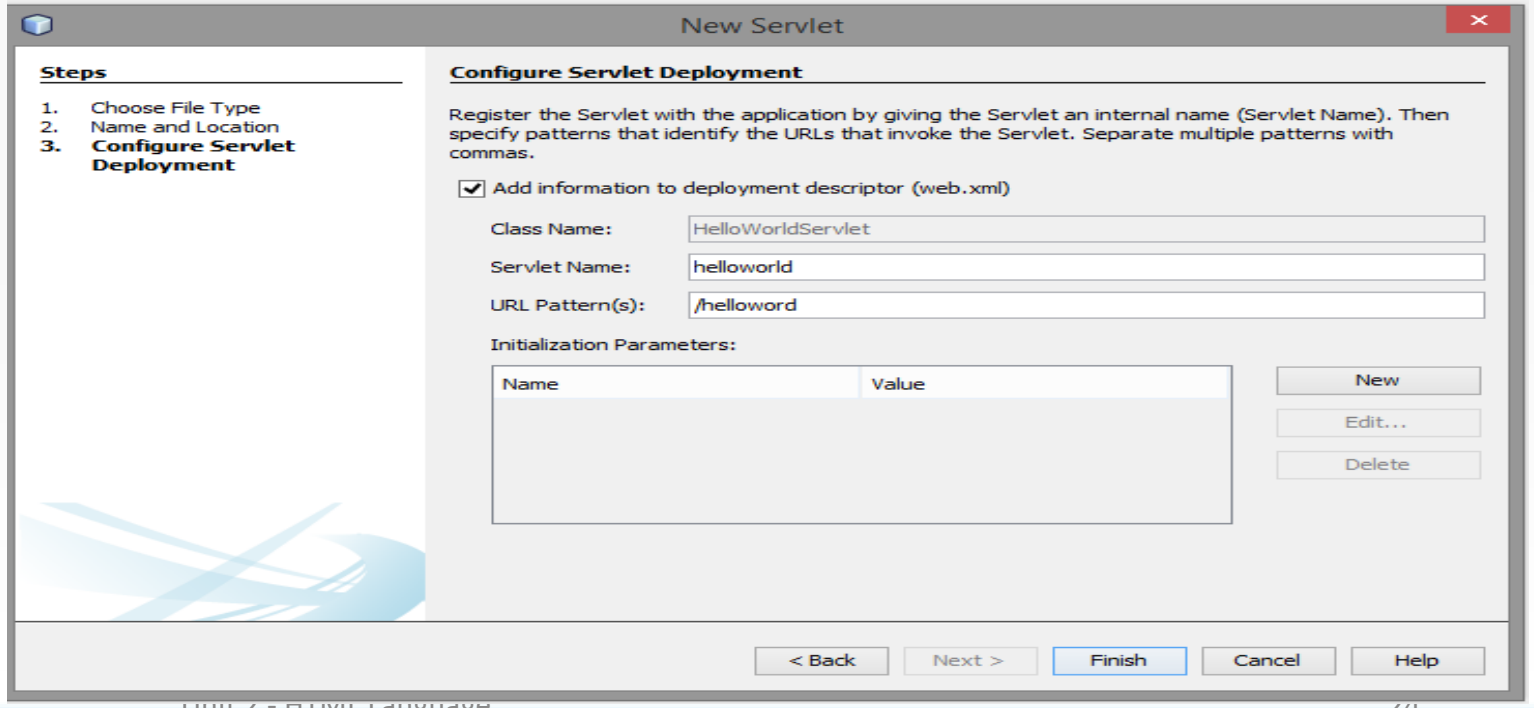
- bin – contains executable files for controlling the server (start, shut down etc.)
- conf – contains configuration files; most important server.xml for configuring the server and web.xml a general configuration file for web applications
- lib – libraries (jars) used by tomcat and deployed web applications
- logs – log files
- temp – temporary files
- webapps – contains the web applications deployed
- work – contains files created by tomcat during running (e.g. it crates a servlet from each jsp file)

Format of a web Application

- Web applications are stored in the webapps folder, either as a folder or as a .war archive
- A web application (either a folder or a .war archive) must contain the following files:
  - WEB-INF folder
  - WEB-INF\web.xml: a configuration file
- Optionally the WEB-INF folder can contain the following subfolders:
  - classes: which contain servlets
  - lib: which contains jars used by the web application
- html, jsp and resource files can be placed anywhere in the web application home folder
- Servlets must be placed in the folder or subfolders of WEB-INF\classes

Servlet Configuration

- For JSPs no additional configuration needs to be done
- For Java Servlets additional lines must be placed in the web.xml file:
- NetBeans has a check box that will add the lines automatically by checking 'Add information to deployment descriptor (web.xml)' box.



Servlet Configuration cont…

```
<servlet>
        <servlet-name>helloworld</servlet-name>
        <servlet-class>HelloWorldServlet</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>helloworld</servlet-name>
        <url-pattern>/helloworld</url-pattern>
</servlet-mapping>
```

Java Server Page (JSP)

- JSP is an html file containing parts of Java code.
- Tomcat will create a Servlet from the jsp file (which will be saved in the work folder)
- When the jsp is requested, the servlet is executed and the output of the server is sent back to the client.
- JSP is an alternate way of creating servlets
  - JSP is written as ordinary HTML, with a little Java mixed in
  - The Java is enclosed in special tags, such as <% ... %>
  - The HTML is known as the template text
- JSP files must have the extension .jsp
  - JSP is translated into a Java servlet, which is then compiled.
  - Servlets are run in the usual way
  - The browser or other client sees only the resultant HTML, as usual
- Tomcat knows how to handle servlets and JSP pages

JSP Scripting Elements

- There is more than one type of JSP "tag," depending on what you want done with the Java
- <%= expression %>
  - The expression is evaluated and the result is inserted into the HTML page
- <% code %>
  - The code is inserted into the servlet's service method
  - This construction is called a scriptlet
- <%! declarations %>
  - The declarations are inserted into the servlet *class*, not into a method

JSP Example

```
<html>
<body>
<!- First JSP file-->
<%
  out.println(“First JSP… It works<br/>");
%>
</body>
</html>
```

JSP Variables

- You can declare your own variables, as usual
- JSP provides several predefined variables
  - request : The HttpServletRequest parameter
  - response : The HttpServletResponse parameter
  - session : The HttpSession associated with the request, or null if there is none
  - out : A JspWriter (like a PrintWriter) used to send output to the client
- Example:
  - Your hostname: <%= request.getRemoteHost() %>

JSP Scriptlets

- Scriptlets are enclosed in <% ... %> tags
  - Scriptlets *do not* produce a value that is inserted directly into the HTML (as is done with <%= ... %>)
  - Scriptlets are Java code that you *may* write into the HTML

    Example:

```
<%
 out.println("Hello JSP");
   %>
```

- Scriptlets are inserted into the servlet exactly as written, and are not compiled until the entire servlet is compiled

  Example:

```
<% if (Math.random() < 0.5) { %>
            Have a <b>nice</b> day!
<% } else { %>
            Have a <b>lousy</b> day!
<% } %>
```

JSP Variable declaration

- Use <%! ... %> for declarations to be added to your servlet class, not to any particular method
  - Caution: Servlets are multithreaded, so nonlocal variables must be handled with extreme care
  - If declared with <% ... %>, variables are local and OK
  - Data can also safely be put in the request or session objects

Example:

```
<%! private int accessCount = 0; %>
      Accesses to page since server          reboot:
<%= ++accessCount %>
```

- You can use <%! ... %> to declare methods as easily as to declare variables.

JSP Directives

- Directives are messages to the JSP container and do not produce output into the current output stream
- Directives affect the servlet class itself
- Syntax:

  <%@ directive attribute="value" %>

  <%@ directive attribute1="value1"

  attribute2 ="value2" … %>

- There are three types of directives:
  i. page - defines attributes that apply to an entire JSP page
  ii. include - inserts another file into the file being parsed
  iii. Taglib - defines a tag library and prefix for the custom tags used in the JSP page.

JSP Directives: Page

- Defines attributes that apply to an entire JSP page.

```
<%@ page
            [ language="java" ]
            [ extends="package.class" ]
            [ import="{package.class |
    package.*}, ..." ]
            [ session="true|false" ]
            [ buffer="none|8kb|sizekb" ]
[ autoFlush="true|false" ]
   [ isThreadSafe="true|false" ]
            [ info="text" ]
          [ errorPage="relativeURL" ]
       [ contentType="mimeType [
    ;charset=characterSet ]"   [
    isErrorPage="true|false" ]
```

JSP Directives: Include

- The include directive inserts another file into the file being parsed
- The included file is treated as just more JSP, hence it can include static HTML, scripting elements, actions, and directives.
- Syntax: <%@ include file="URL" %>
  - The URL is treated as relative to the JSP page
  - If the URL begins with a slash, it is treated as relative to the home directory of the Web server
- The include directive is especially useful for inserting things like navigation bars.

JSP Directives: taglib

- Defines a tag library and prefix for the custom tags used in the JSP page.
- Syntax

  <%@ taglib uri="URIToTagLibrary" prefix="tagPrefix" %>

  <%@ taglib uri="http://thathost/tags" prefix = "public" %>

  The <%@ taglib %> directive declares that the JSP file uses custom tags, names the tag library that defines them, and specifies their tag prefix.

JSP Example

```
<%@page contentType="text/html" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html> <head>
        <meta http-equiv="Content-Type" content="text/html;
  charset=UTF-8">
        <title>Java Server Page</title> </head>
    <body>
<%! int number1, number2, sum;%>
<%   number1=19;
         number2=4;
         sum=number1+number2;
%>
        <%-- Comments in JSP-- %>
  The sum of the two numbers is <%= sum %>
       </body>
</html>
```

JSP with MySQL Database

MySQL Connection to JSP

The database is used for storing various types of data which are huge and has storing capacity in gigabytes. JSP can connect with such databases to create and manage the records.

JSP can be connected to database by using JDBC

Steps to connect database to JSP file:

Create Database: You can create database by using phpMyAdmin of XAMPP server.

Create Tables: In MYSQL database, we can create a table in the database with any MYSQL client.

Create JSP project: create a JSP project with required files.

MySQL Connection to JSP

In JSP file add the following lines of codes for connection:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.sql.*"%>
<%@page import="javax.servlet.ServletException" %>
<%@page import ="java.util.Properties" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
       <%
            String url,user,password;
        url="jdbc:mysql://localhost/autoshop";
        user="root";
        password="";
```

MySQL Connection to JSP cont…

```
try {
    java.sql.Connection con;
     Class.forName("com.mysql.jdbc.Driver");
            Connection
   connection=DriverManager.getConnection(url,user,password);
    out.println (user+ "database successfully opened.");
      Statement statement=connection.createStatement();
  }
  catch(SQLException e) {
    out.println("SQLException caught: " +e.getMessage());
  }
%>
    </body>
</html>
```

JSP in XML

- JSP can be embedded in HTML as well as in XML.
- Due to XML's syntax rules, the tags must be different (but they do the same things).

HTML: <%= expression %>
XML: <jsp:expression>expression</jsp:expression>
HTML: <% code %>
XML: <jsp:scriptlet>code</jsp:scriptlet>
HTML: <%! declarations %>
XML: <jsp:declaration>declarations</jsp:declaration>
HTML: <%@ include file=URL %>
XML: <jsp:directive.include file="URL"/>

Thank you!!