Java Servlets

Web App

A web application, or 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 web apps by entering its domain name (e.g. facebook.com, gmail.com) into their browser URL bar. When this happens, the browser connects to the server hosting the application, and displays whatever information it has received from the server.

Web applications grew in popularity because of the benefits it provides to both customers and service providers. Service providers now have central management of their applications, allowing for easier, cheaper upgrades, among other things, while customers can check out their services and avail it no matter where they are. Most web apps, if not all, focuses on an application that provides the best possible user interaction and user experience, which may eventually lead to more revenue. To make this happen, it is recommended for web apps to have dynamic content.

Applets

Applets are one of the earliest attempts toward this goal, focusing on using the client platform to deliver dynamic user experiences. At the same time, developers also used the server platform for the same 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 that includes 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.

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.

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 by the Java Servlet API. The *HttpServlet* class provides methods, such as *doGet* and *doPost*, for handling HTTP-specific services.

Java Servlet 3.1 (Java EE 7)

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

When a browser sends an HTTP request to the web server, the server will send the request to the servlet container. From there, it will determine which servlet to invoke. When the servlet is done 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

The life cycle of a servlet has 4 stages: instantiation, initialization, request handling, and destruction.

1. Instantiation

- 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.

HTTP Servlets

HTTP Servlets handles HTTP requests and generates responses. Hosted in a web container, it takes care of converting requests to responses. The service() method call is routed to a doXXX() handler method

doXXX

List of doXXX calls:

- doGet
- doPost
- doPut
- doDelete
- doTrace
- doOptions
- doHead

The doXXX call requires two arguments:

1. HTTPServletRequest
2. HTTPServletResponse

As shown in the diagram above, the client or the web browser makes requests of any type, which the server sends off to the servlet's service() method. This method routes the requests to the doXXX() handler method for each HTTP request type.

Methods of HTTP Servlet class

1. protected void doDelete (HttpServletRequest req, HttpServletResponse resp)
2. protected void doGet (HttpServletRequest req, HttpServletResponse resp)
3. protected void doHead (HttpServletRequest req, HttpServletResponse resp)
4. protected void doOptions (HttpServletRequest req, HttpServletResponse resp)
5. protected void doPost (HttpServletRequest req, HttpServletResponse resp)
6. protected void doPut (HttpServletRequest req, HttpServletResponse resp)
7. protected void doTrace (HttpServletRequest req, HttpServletResponse resp)
8. protected long getLastModified (HttpServletRequest req)
9. protected void service (HttpServletRequest req, HttpServletResponse resp)
    - mReceived the Http request from clients
10. Forwards client request to the protected service methodvoid service (ServletRequest req, ServletResponse resp)

Java Server Pages, or JSP, is a Java view technology under Oracle's Java EE (Enterprise Edition) for web development. Compared to Java Servlets, which is comprised of HTML code within a Java program, JSP allows Java code to be inserted into HTML or XML pages. The codes can then be run to generate dynamic content embedded within web pages, as well as connect to external databases.

Apache Tomcat and JDK (Java Software Development Kit) should be installed as they are required in order to run JSP.

When a client sends a request over the internet, the web server compiles files with the *.jsp* extension, converting them into executable Java Servlets before serving or processing client HTTP requests. (Note: it first checks the page; if the page has already been compiled and has no modification from its last compilation, then conversion/compilation is not necessary.) This produces a document, like a HTML page, which is then handled like any other page or servlet and sent back to the client with a HTTP response.

The **`jspInit()`** method is invoked to initialize external connections, like with databases or files, before running the servlet. The **`_jspService()`** method is invoked when there is a need for responding to HTTP requests. It dies with **`jspDestroy()`**.

Elements

Actions
Uses: inserts a file, reuses JavaBeans, forwards, generates html
Syntax: `<jsp:actionName attrib="value" />`

actionName	**Notes**
include	called when page is requested
useBean	(looks for or creates object) (can use setProperty/getProperty only after this has been called)
setProperty	
getProperty	
forward	can only forward to another page or servlet
plugin	inserts java components
element	
attrib	
body	
text	write template text

Directives
Uses: Instructions for how to translate and execute JSP
Syntax: `<%@ directiveName attrib=”value” %>`

List of directiveNames
- page (for container)
- include (called when translated into servlet - usually java packages)
- taglib

ex:
handling jsp page errors
```
      <%@ page errorPage="filename" %>
```
static html
```
      <%@ page contentType="text/html" &>
      <%@ page pageEncoding="utf-8" %>
```

Expressions
Syntax: `<%= expression %>`
ex:
```
    <%= new java.util.Date() %>
```

Scriptlets
Uses: inserts any Java code within the JSP file
Syntax: `<% code %>`

Declarations
Syntax: `<%! code %>`
ex:
```
    <%! int i = 0; %>
```

Comments
Syntax: `<%-- comment --%>`

Pre-defined Objects
ex.
- request (HTTPServletRequest)
- response (HTTPServletResponse)
- out (PrintWriter)
- session (HTTPSession)
- application (ServletContext)
- config (ServletConfig)
- page (this)
- exception

References

Java™ Servlet Specification version 4.0 (Version 4.0, Shing Wai Chan, Ed Burns, July 2017)
https://docs.oracle.com/javaee/5/tutorial/doc/bnafe.html
https://docs.oracle.com/javaee/5/tutorial/doc/bnafd.html
https://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServlet.html
tutorialspoint
beginnersbook (singh, c.)
docs.oracle.com/javaee/5
http://condor.depaul.edu/mwright1/j2ee/index.html