ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Java Applets vs. Java Servlets

Anjali Rajput[1], Dipti Bhardwaj[2], Sakshi Ahuja[3]
Department of Computer Science, Dronacharya College of Engineering

Abstract: ***An applet is a Java program that can be included in an HTML page and executed by a Java Technology-enabled browser. It can be used in web-page enhancement and on enterprise intranets for sharing resource and data. Applets allow local validation of data entered by the user, can Using database to perform list of values lookups and data validation, and have Complex GUI widgets. Servlets are protocol- and platform-independent server side components, which runs as part of a network service and dynamically extend Java-enabled servers. With applets, servlets can provide interactivity/dynamic Web content generation. Servlets can accept form input and generate HTML Web pages dynamically like CGI, support Collaborative applications by synchronizing request, partition a single logical service between several servers, act as active agents sharing data. Servlets support different protocol, can be part of middle tiers in enterprise networks. Servlet are more appropriated when involves loading large pieces of code over a slow communication channel; when a large part of the computation for generating the Web page can be done on the server side or when processing involvesoperationsthatappletscannotperformduetosecurityrestrictions. If a sophisticated user interface is desired, applets are appropriate. Applets and servlets can share data and communicate, so the processing can be split between them. Client-side Java is a glorious vision that does and will not change the way most people use the Internet anytime soon. Sun has taken some dramatic steps to insure that Java remains ubiquitous by creating a newset of APIs that allow developers to use Java as a server-side development tool.***

I. INTRODUCTION

Web applications in the Java programming language can be implemented using two technologies: Java Servlets and Java Server Pages (JSP) [11]. The optimal result is achieved by a combination of both technologies, as described below. Servlets are Java objects that dynamically process requests and generate responses. Generated response is usually in the form of HTML (Hyper Text Markup Language), but the data can be in another form, for example XML (Extensible Markup Language) [3, 8].Servlets run on the server and the program code is never disclosed outside the server, only derived results are submitted. This way, the author code is fully protected, as opposed to applications that run in a Web browser using the Java Applet technology [1]. Another more advantage of using Servlets is their portability. They can run on different servers without changing the code [23].After the client sends a request to a server; it loads the servlet and creates a separate thread. Each servlet has the methods Do Get () or do Post () which are used to process the request. For each request a separate thread is created, which makes execution significantly faster than any other technology, in which, operating system must create a new process for each request, initialize it, and run it and cleanup after it, which is extremely slow. One example of such technology is the application logic is "hardcoded" in the user interface. These problems are solved by combining these two technologies in order to split the Web application into layers. A typical division into three layers: the business layer, presentation layer and data layer. Business layer (business logic layer) is a layer responsible for the application logic, and is run through servlets.

Java applets were included in the initial release of the Java Software Development Kit (JDK) in 1996. An applet is a Java program that is run in the context of another application, typically a web browser with the Java plug-in. The Java plug-in is part of the Java runtime environment (JRE) and allows Java code to be interpreted inside a web browser. The JRE can be freely downloaded (http://www.java.com) and installed on computers running any operating system. Applets are downloaded to the client machine and run using the JRE. Although applets have access to server resources and can communicate with servlets running on the host server, security restrictions prevent applets from reading or writing files, running programs, or accessing information on the local client system.[5]

Servlets are Java programs that run on a web server to translate dynamic content into hypertext markup language (HTML) pages. They act as a middle layer between client requests and server resources such as data or other applications. Servlet functions are processed on the server; however, they can be invoked through a network (like the Internet) by a web browser. Only servlet output, formatted as HTML, is returned to the web browser. Java server pages (JSP) are an extension of servlets that allow dynamic servlet code to be interlaced in static HTML. JSP documents are translated into regular servlets by the web server the first time the page is

requested. A single instance of the Java interpreter, called the Java virtual machine, remains active in memory on the server and creates a light-weight Java thread to handle new client requests. SWS system work as a web-based application by adopting Java Applet and Servlet technologies. Applet [1] is a small Java program that can be embedded in an HTML page. Its common rule is to make an Internet connection to the computer from which the Applet was sent. Servlets [3] are modules of Java code running in server application (hence the name "Servlets" on the server side is similar to "Applets" on the client side). Their task is to answer client requests. Servlets are not tied to some specific client-server protocol though normally they are used with HTTP. The word "Servlet" is often explained as "HTTP Servlet"

A. Applets

Java was chosen as the baseline programming language for this project. The entire GUI (Graphical User Interface) for this system has been designed using the Java SWING packages. Java is the language of the internet. In addition to the programming language itself, Java has a rich library that makes it possible to write portable programs that are operating system independent. Java also has a rich set of security features that help in learning the language faster. The java virtual machine can catch many kinds of mistakes and report them accurately. Instead of producing executable code, a Java compiler produces byte code. The Java run-time system is an interpreter for byte code. Once the Java run-time package exists for a given system, the byte code version of any Java program can run on it. Therefore, Java produces truly portable programs.

The combination of the World Wide Web and Java increases the prominence of Internet in information systems planning and implementation. The World Wide Web makes tremendous amounts of information available to anyone with a web browser. Applets are java programs that can be embedded in HTML documents and can run inside a web browser. The applet code resides on the web server and gets downloaded in the browser whenever the web page containing the applet is requested by the browser.

1) Capabilities and Limitations of Applets

Java applets obviously have many potential capabilities because of their tight security model, their generally small size, there and network awareness. Java applets can be used to build full-featured graphical user interfaces, communicate over the Internet to a host server, and even communicate with other applets on a form. All of this can be done in an operating-environment-neutral manner, which is what makes this such a great technology. For Java to be truly successful, however, the client security has to be completely assured. Because of this, security measures place some limitations on Java applets. By default, applets cannot communicate with any server other than the originating server. Applets also cannot read or write les to the local le system. The growth of technologies such as Web-based client/server application development and electronic commerce has been severely limited by the lack of "industrial-strength" security. Because the underlying Internet was never designed to handle secure transactions (the Department of Defense has a separate net for this purpose), the entire infrastructure of the InternetwassomewhatunpreparedforthephenomenalgrowthoftheWorldWideWeb over the last few years. The concept of applets (or related technologies such as software agents) has been discussed in academic circles for years, yet most theoretician serialized the security

2) Applet vs. HTML

Applets allow local validation of data entered by the user. Local validation of data is possible using HTML combined with JavaScript but variances in JavaScript implementations make Java Script difficult to generally use. An applet can use the database to perform list of values lookups and data validation.HTML (even if combined with JavaScript) cannot do that without invoking a CGI or servlet program and drawing a new HTML page. Once an applet is downloaded, the amount of data transferred between the Web browser and the server is reduced. HTML requires that the server transfer the presentation of the data (the HTML tags) along with the data itself. The HTML tags can easily be 1/4 to ½ of the data transferred from the server to the client. Applets allow the designer to use complex GUI widgets such as grids, spin controls, and scrollbars. These widgets are not available to HTML.

B. Servlets

Servlets are the server side counterparts of applets. They run in a special environment of the server which supports the servlet API. Servlets do not have a user interface but can access all server resources. Servlets are easy to use and understand. Their performance is good, because once a servlet is loaded, the server only makes simple method calls and does not need to create a new process for each request. A servlet is a server-side software program, written in Java code that handles messaging between a client and server. The Java Servlet API defines a standard interface for the request and response messages so your servlets can be portable across platforms and across different Web application servers. Servlets can respond to client requests by dynamically constructing a response that is sent back to the client. For example, in this tutorial you'll write a servlet to respond to an HTTP request. Because servlets are written in the Java programming language, they have access to the full set of Java APIs. This makes them ideal for implementing complex business application logic and especially for accessing data elsewhere in the enterprise. The Java Database Connectivity (JDBC) API, which allows Java programs to access relational databases (and is beyond the scope of this tutorial), is one example. Because there are no graphics associated with servlets, access to the GUI Java APIs (the AWT) is not relevant. A single servlet can be invoked multiple times to serve requests from multiple clients. A servlet can handle multiple requests concurrently and can synchronize requests. Servlets can forward requests to other servers and servlets. A servlet runs inside a Java-enabled application server. Application servers are a special kind of Web server; they extend the capabilities of a Web server to handle requests for servlets, enterprise beans, and Web applications. There is a distinct difference between a Web server and an application server. While both can run in the same machine, the Web server runs client code such as applets and the application server runs the servlet code. The server itself loads, executes, and manages servlets. The server uses a Java bytecode interpreter to run Java programs; this is called the Java Virtual Machine (JVM).Servlets are either invoked as an explicit URL reference or are embedded in HTML and invoked from a Web application. You don't execute a Java command to run a servlet (as you would an applet); instead, you issue a URL command pointing to the location of the servlet. Servlets are located in any directory on a machine where an application server is running. Typically, there is a servlet directory to which you will ftp or copy your class files so the application server can find them. This location can be configured differently, depending on the administrator and the environment. Servlets are loaded when they are invoked for the first time from a client. Alternatively, they can be loaded when the application server is started; this is a configuration decision.

1) A Servlet That Generates HTML

Most servlets generate HTML, not plain text as in the previous example. To generate HTML, you add three steps to the process just shown:

1. Tell the browser that you’re sending it HTML. 2. Modify the println statements to build a legal Web page. 3. Check your HTML with a formal syntax validator.

You accomplish the first step by setting the HTTP Content-Type response header to text/html. In general, headers are set by the set Header method of Http Servlet Response, but setting the content type is such a common task that there is also a special set Content Type method just for this purpose. The way to designate HTML is with a type of text/html, so the code would look like this:

response.setContentType("text/html");

Although HTML is the most common kind of document that servlets create, it is not unusual for servlets to create other document types. For example, it is quite com- mon to use servlets to generate Excel spreadsheets (content type application/ vnd.ms-excel—see Section 7.3), JPEG images (content type image/jpeg—see Section 7.5), and XML documents (content type text/xml). Also, you rarely use servlets to generate HTML pages that have relatively fixed formats (i.e., who’s lay- out changes little for each request); JSP is usually more convenient in such a case. JSP is discussed in Part II of this book (starting in Chapter 10). Don’t be concerned if you are not yet familiar with HTTP response headers; they are discussed in Chapter 7. However, you should note now that you need to set response headers before actually returning any of the content with the PrintWriter. That’s because an HTTP response consists of the status line, one or more headers, a blank line, and the actual document, in that order. The headers can appear in any order, and servlets buffer the headers and send them all at once, so it is legal to set the status code (part of the first line returned) even after setting headers. But servlets do not necessarily buffer the document itself, since users might want to see partial results for long

pages. Servlet engines are permitted to partially buffer the output, but the size of the buffer is left unspecified. You can use the get Buffer Size method of Http Servlet Response to determine the size, or you can use setBufferSize to specify it. You can set headers until the buffer fills up and is actually sent to the client. If you aren't sure whether the buffer has been sent, you can use the is Committed method to check. Even so, the best approach is to simply put the set Content Type line before any of the lines that use the Print Writer.

2) The Servlet Life Cycle

In Section 1.4 (The Advantages of Servlets Over "Traditional" CGI) we referred to the fact that only a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. We'll now be more specific about how servlets are created and destroyed, and how and when the various methods are invoked. We summarize here, then elaborate in the following subsections. When the servlet is first created, its init method is invoked, so init is where you put one-time setup code. After this, each user request results in a thread that calls the service method of the previously created instance. Multiple concurrent requests normally result in multiple threads calling service simultaneously, although your servlet can implement a special interface (SingleThreadModel) that stipulates that only a single thread is permitted to run at any one time. The service method then calls doGet, doPost, or another doXxx method, depending on the type of HTTP request it received. Finally, if the server decides to unload a servlet, it first calls the servlet's destroy method.

C. The service Method

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc., as appropriate. A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified. A POST request results from an HTML form that specifically lists POST as the METHOD. Other HTTP requests are generated only by custom clients if you aren't familiar with HTML forms (Creating and Processing HTML Forms).

1) The doGet, doPost, and doXxx Methods

These methods contain the real meat of your servlet. Ninety-nine percent of the time, you only care about GET or POST requests, so you override doGet and/or doPost. However, if you want to, you can also override doDelete for DELETE requests, doPut for PUT, do Options for OPTIONS, and doTrace for TRACE. Recall, however, that you have automatic support for OPTIONS and TRACE. Normally, you do not need to implement doHead in order to handle HEAD requests (HEAD requests stipulate that the server should return the normal HTTP headers, but no associated document). You don't normally need to implement doHead because the system automatically calls doGet and uses the resultant status line and header settings to answer HEAD requests. However, it is occasionally usefulto implement doHead so that you can generate responses to HEAD requests (i.e., requests from custom clients that want just the HTTP headers, not the actual docu- ment) more quickly—without building the actual document output.

2) The init Method

Most of the time, your servlets deal only with per-request data, and doGet or doPost are the only life-cycle methods you need. Occasionally, however, you want to perform complex setup tasks when the servlet is first loaded, but not repeat those tasks for each request. The init method is designed for this case; it is called when the servlet is first created, and not called again for each user request. So, it is used for one-time initializations, just as with the init method of applets. The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started. The init method definition looks like this:

public void init() throws ServletException { // Initialization code... }

The init method performs two varieties of initializations: general initializations and initializations controlled by initialization parameters.

3) The destroy Method

The server may decide to remove a previously loaded servlet instance, perhaps because it is explicitly asked to do so by the server administrator or perhaps because the servlet is idle for a long time. Before it does, however, it calls the servlet's destroy method. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities. Be aware, however, that it is possible for the Web server to crash (remember those California power outages?). So, don't count on destroy as the only mechanism for saving state to disk. If your servlet performs activities like counting hits or accumulating lists of cookie values that indicate special access, you should also proactively write the data to disk periodically.

4) Possible Usage

As already shown, servlets can be used for processing HTML forms. They can dynamically generate HTML pages, e.g. presenting results of database queries or data analyses. Moreover, they are able to manage sessions which is important for successive requests. For example, based on the result of a cluster analysis it is possible to select one cluster for a subsequent analysis.

D. Java Applets vs. Java Servlets

Servlets are named after applets which are also written in Java but which run inside the JVM of a HTML browser on the client. Servlets and applets allow the server and client to be extended in a modular way by dynamically loading code which communicates with the main program via a standard programming interface. Basically, a servlet is the opposite end of an applet. A servlet can almost be thought of as a server-side applet. Servlets run inside the Web server in the way that applets run inside the Web browser. The browser can submit a request to execute a servlet directly; it can be stand-alone in terms of its actions {as a browser can request an applet directly. Since servlets run inside servers, they do not need a graphical user interface, often referred to as faceless applets. Servlets are free of the security restrictions that apply to applets. This is because they run within a Web server on the server-side. Thus, they are trusted programs, Java application components which are downloaded, on demand, to the part of the system which needs them. Like applets, servlets may be called from HTML les dynamically and there are several cases in which the two could be used interchangeably. So when should we design servlets and, when should we design applets? The answer to this question goes back to the basic issue of load distribution between the client and the server. The distributed client/server paradigm has shifted over the past few years from fat clients to thin clients and subsequently from thin servers to fat servers. Applets are representative of the client side of the architecture and servlets represent the server side. Some scenarios in which servlets are more appropriate are given below:

1. Applet classes are downloaded over the Internet to the client and then executed in a JVM running on the client. If this involves loading large pieces of code over slow modem lines, applets are not the appropriate choice.

2. If a large part of the computation for generating the Web page can be done on the server side, it is pointless to load the part of the code that does the computation to the client. The computation should be done on the server and the results passed back to the client.

3. If processing involves operations that applets cannot perform due to security restrictions, then a local servlet may be used.

Applets are more appropriate in the following scenarios:

1. Applets basically have a well-dined user interface (remember that they derive from Panel). In servlets, on the other hand, a user interface would have to be built from scratch. Therefore, when a sophisticated user interface is desired applets are appropriate.

2. If the speed of the communication channel is adequate, then the overhead involved in downloading applets may not be an issue.

Applets and servlets can also share data and communicate. Therefore, the processing can be split between them.

II. CONCLUSION

It's a peculiar moment in the industry's history. The Java buzz is intense. And yet when you look at the Web applications that people actually use every day to do their work, you invariably and that there are no Java applets in the mix. The universal client today is still the HTML browser. The universal client of tomorrow will be the HTML/Java Script browser. Client-side Java is a glorious vision that will not change the way most people usethe Internet anytime soon. It's just more than what the majority of today's computers and networks can readily push. So what are millions of people running every day? Server- based applications that feed the universal HTML client.

Server-side scripting is still very popular for several reasons:

Totally independent of the browser since everything takes place on the server;

Complex requests may execute faster on the server;

Can be made safer since the programs run under direct control of the server administrator

Java servlets give you the capability to develop complex server-side applications. Servlets leverage Java's object-oriented features to build reusable and modular components. You can easily create servlets to replace CGI applications, access databases and communicate with remote computers.

As promising as applet development is, developers' frequent lack of control over the internal appearance and performance of their products is often frustrating. The limitations on applet capabilities and the changes between Java versions make it difficult to apply the full strength of Java's incredibly powerful programming structures, restricting the real-world use of Java fairly severely. As applet development becomes a specialized held for multimedia toys and Carefully-built applications, Sun has taken some dramatic steps to insure that Java remains ubiquitous {and useful {by creating a new set of APIs that allow developers to use Java as a server-side development tool.

REFERENCES

[1] http://jserv.javasoft.com:80/products/java-server/documentation/webserver1.0.2/servlets/api.html
[2] http://www.sys-con.com/java/feature/3-1/servletsfriends/
[3] http://www.sys-con.com/java/feature/3-1/cgi-scripts/index.html
[4] http://www.servletcentral.com/common/articlelist.dchtml
[5] http://webreview.com/wr/pub/97/10/10/feature/colton.html
[6] http://www.developer.com/news/techfocus/021698 servlet.html
[7] http://www.sys-con.com/java/feature/3-2/3-tier/index.html
[8] http://www.servletcentral.com/
[9] http://javaboutique.internet.com/
[10] http://www.javasoft.com/applets/index.htm
[11] http://www.javasoft.com/features/1997/oct/applets.html
[12] http://www.javasoft.com/features/1997/dec/applets2.html
[13] http://www.javasoft.com/features/1998/07/applet.power.iii.html
[14] HUNTER, J. (1998): Java Servlet Programming, O'Reilly & Associates Inc., Se- bastopol.
[15] KUHLINS, S., SCHADER, M. (1999): Remote Data Analysis Using Java, in: Studies in Classification, Data Analysis, and Knowledge Organization, Gaul, W., Locarek- Junge, H. (Eds.), Classification in the Information Age, Springer-Verlag, June 1999, pp. 359–367.
[16] APACHE GROUP (1999): Apache Web Server, http://www.apache.org/.
[17] JAVA APACHE PROJECT (1999): Apache JServ, http://java.apache.org/jserv/index.html.
[18] SUN MICROSYSTEMS INC. (1999a): Java Development Kit, http://java.sun.com/products/jdk/.
[19] SUN MICROSYSTEMS INC. (1999b): Java Servlet Development Kit, http://www.javasoft.com/products/servlet/index.html.
[20] SUN MICROSYSTEMS INC. (1999c): Java Web Server, http://www.sun.com/software/jwebserver/index.html.

[21] Java Server Pages - Apache Tomcat. Website: http://java.sun.com/products/jsp/tomcat/
[22] Frank Naudé, (2006, March). Glossary of Terms: R. Website: http://www.orafaq.com/glossary/faqglosr.htm 23. SOAP Version 1.2 Part 1: Messaging Framework. Website: http://www.w3.org/TR/soap12-part1/#intro