Articles & White Papers

JAVA OBSERVER ***By Max Dolgicer and Peter Fischer***

The difference between Java applets, applications

No one can deny the unprecedented fervor that surrounds Java as the ultimate programming language. It has been highly touted as the language to be using to program all your Internet/Intranet applications today.

The biggest manifestation of this is the tremendous number of Web pages -- characterized by dazzling, scrolling graphics -- activated by Java applets.

Is this the only way in which to leverage Java as a programming language? The answer is no. There are two different ways to run a Java executable: either as a standalone application or as an applet that is embedded in a Web page. Significant differences between the two executable models exist and will factor in how Java is used to build systems, now and in the future.

What is an applet?

Let's start the discussion with by far the biggest use of Java to-date -- the applet. Applets were originally intended to supply small, finite units of functionality for activating Web pages and have held true to that promise to this day. However, as Java has matured, applets are now seen as one way of delivering component-based functionality to Web pages. The original intent is that applets would be small applications -- hence the name. As Java matures and its use for creating active Internet and Intranet applications increases, this distinction will become irrelevant. Instead the distinction will be based solely on technical characteristics.

Java applets leverage and take advantage of features that are built into Web browsers. This enables developers to write applets that can contain a rich amount of functionality with a minimal amount of code. One of the major uses of applets is to display graphics and images. To incorporate GIF or JPEG files into a Java applet, it is not necessary to write any special decoding methods to interpret the image files. Instead applets can use the browser's built-in decoder to display the images.

Using the functionality built into the browser offers an applet a good degree of extensibility. If a new image file format becomes the hot format and is incorporated into the browser, the Java applet will automatically be able to handle the new format without any custom code having to be written.

An applet has no life outside an HTML page that is being processed by a browser. Thus the browser is the "container" of the applet and depends on the browser to call its methods. There are a lot of restrictions on the functionality of an applet. If you think about this, it is not unreasonable when you consider that an applet is an invited or uninvited guest that borrows your machine to execute and display.

Because applets run inside the browser container, the distinction about whether such restrictions derive from the browser or are inherent in the language often get blurred. This has a significant impact on the functionality that an applet can have. In a sense, a specification between the browser and the applet exists, which is driven by both the browser and the Java language. This is dictated by the Java Security Manager component that is part of the browser. When you select the Options Security menu item in Navigator, the Java selections in the dialog are controlling the level of security enforced by the browser. By the way, no real security specification for browsers exist, just a de facto acceptance of what security really means in the context of a browser running foreign code.

Four major types of restrictions are imposed on applets. First, an applet has limited access to the local and remote file systems. This is due in part to both the security features of Java as well as the browser. Both Netscape and Internet Explorer simply forbid all access.

Second, an applet class that was loaded from the Internet is not allowed to make any native calls. These remote classes cannot execute any local commands. This restriction is lifted for classes that are used by the applet but are loaded from the local machine. Some examples of such local classes are base classes such as Object and Component. Third, an applet cannot be a network socket server and may only open a socket connection to the server machine that served the applet's HTML page.

Lastly, there are restrictions imposed on inter-applet communication that are also driven by the browser. Netscape Navigator, versions 2.x, limits all inter-applet communications to applets on the same HTML page. Navigator 3.0 further restricts these applets to come from the same directory on the Web server and to have the same level of support for JavaScript.

An applet's life cycle begins when the user visits the Web page that contains the applet. After constructing the applet, the browser allocates a place for it within its viewing area on the screen.

At this juncture an important event occurs -- the applet is assigned a peer. What is this thing called a peer? It is the essence of the platform independence that makes Java such a powerful programming language. In the nutshell, a peer is an applet's connection to the local, platform-specific underlying Windows system. This peer is an inherent part of the Java virtual machine for that platform. Java components, which include applets as well as applications, receive much of their functionality by using the facilities of the local Windows system.

Next the browser sends several startup method calls to the applet. The first applet method that is called is init. This method is only called once and all one-time initialization for the applet is done here. Such functionality includes creating the applet's GUI and reading in the parameters. The browser then follows this with a call to the start method which basically puts the applet into the "active" state. This means that the page that houses this applet is now the active page in the browser.

When the user navigates away from this page, the browser calls the stop method, putting the applet into the "inactive" state. The applet is running and is not destroyed because many browsers cache Web pages.

When the applet needs to render its representation in the page, the paint method is called. As applets tend to be graphical in nature, this is where most of the processing action occurs. This method is called the first time the applet is displayed and any time that its needs to be repainted.

The destroy method is called when the browser no longer needs the applet. In reality, because most browsers cache the applet when the Web pages gets cached, this method is rarely called.

Whew, there is a lot of stuff going on here! In reality, an applet is simply a finite state machine that provides methods that are called on demand by the browser. It is a reactive component completely controlled by the browser and whose main responsibility is to implement the correct methods for it to function correctly.

What is an application?

An application is basically a program that just happens to be written in the Java language. At a high level, it is just like any application that is written in another programming language. From the Java perspective, an application runs without a browser, a security manager and without applet context.

An application is invoked from the command line by passing the class name of the Java application to the Java runtime and when execution starts and stops. It is the proactive voice in the execution context.

In general, applications have an easier lifestyle compared to applets. They have complete access to the file system and network, as you would expect from any typical application. They can also invoke native methods. Native methods are basically code that is written in another language such as C that is called from Java. In a way, it permits access to "legacy code" (from the Java perspective, of course) from Java. This gives them the ability to be very functional as compared to applets.

If the picture seems rosy for an application, it is for the most part. The one drawback to being an application is that the peer is not created automatically. Fortunately, most initialization activities do not require the presence of a peer. Usually this is not a big issue except in one instance, which is when dealing with images. In this case the peer must be created by the Java application programmer and assigned to the application before any image creation routines can be called. This is just one of the basic rules of Java programming. All languages have them.

As applets are called upon to perform more functionality, the support for interactions between applets, browsers and network resources can be expected to mature and become more refined. Some of the more recent advances have been signed applets, which provide the basis for secure electronic commerce as the Java Wallet technology. As new uses for the Internet and Intranet emerge, Java and applets will be sure to evolve with them.

Most of the use of Java is directed towards building applets, which enhance and activate Web pages. As more and more businesses and corporations adopt Java as an enterprise language, the use of Java applications will become more prevalent. Certainly, the recent release of the Java Enterprise technologies (Java RMI, JDBC) positions Java applications as distributed-object applications for the enterprise. Applets are the shining star of Java currently, but Java applications are just starting to glow.

*Max Dolgicer is a director at International Systems Group Inc. (**isg@isg-inc.com**), a New York City-based consulting and research firm that specializes in the design, development and implementation of complex distributed applications using middleware. Mr. Dolgicer can be reached by E-mail at **mdolgicer@isg-inc.com** (corporate) or **mdolgicer@aol.com** (personal).*

*Peter Fischer is a senior consultant at International Systems Group Inc. With over 10 years experience in developing distributed systems, he is a recognized authority on implementing distributed object systems using DCOM and Corba technologies. Mr. Fischer can be reached by E-mail at **pfischer@isg-inc.com.***

Summary of differences between Javaapplets and applications

Feature	*Applet*	*Application*
Security	Security Manager provided by browser, Remote code is untrusted	No default Security Manager. All code is trusted by default
Execution Privileges	Runs in the context of a browser	Runs standalone
Container	Browser	None
Communication	Inter-applet communication limited by browser	No restrictions
Lifecycle	Controlled by browser	
Peer	Created and assigned by the Browser	No default. Must be created by the application.
Network Access	May only open a socket connection to the host it was loaded from	No restrictions
Invoke Native Methods	No	Yes

illustration by chris spollin