

WAKE UP to Server-Side Java

Cheryl Walton



Not long ago, you may have associated the Java programming language almost exclusively with applets and dynamic web pages. You may have also thought Java Virtual Machines (JVMs) were included only with web browsers, such as Netscape Navigator. In other words, you probably thought Java was used primarily to write programs that jazz up web pages and run on web browsers.

Times have changed, however, and Java isn't just for browsers anymore. Java can do a lot more than make a snappy web page. In the past few years, increasingly sophisticated JVMs have been included with server operating systems, such as NetWare 5. These server-side JVMs have spawned a growing number of Java programs that run on servers rather than on browsers.

With these Java programs, you can add interest to your company's web site and extend the usefulness of resources that run on the server. For example, Java servlets, like applets, can activate otherwise static web pages. (A *servlet* is a Java program that executes on an HTTP web server on which a servlet-enabled JVM is running.) Servlets can also access information from other programs that are running on the server and then dynamically create web pages that contain this information.

Other server-side Java applications can also make servers more useful. For example, you can use server-side Java applications to host Internet or intranet chat rooms in which your company's employees, customers, or business partners can conduct online meetings in real time. You can also use these server-side Java applications to access databases and to write reports based upon the data accessed. (For more information about some interesting server-side Java applications, see "A Java Sampler" on p. 18.)

Before you begin to explore the available server-side Java applications, you may find it helpful to know how these applications work. You should also know that although Java is largely a platform-independent programming language, most installation programs for Java applications are not platform independent. Because many installation programs are programmed to install the Java application in a Windows NT or 95 directory, you may need to learn a few techniques for installing and running Java applications on the NetWare 5 JVM.

NOW SERVING JAVA

One of Java's inherent benefits is its ease of use. According to Steve Holbrook, a Java product specialist for Novell, Java also

allows programmers to be more productive than they would be if they were using other programming languages. "It's a great language," Holbrook states, adding that programmers who "articulate their solutions" in Java get their products to market sooner than they otherwise would. In addition, Java-based products have fewer bugs than products written in other languages.

What makes Java easy to use? Java classes, for one thing.

Java in the Class, Java in the Library

Java classes encompass the categories to which specific types of objects (or variables) belong and the methods (or procedures) that you can perform on those objects. In other words, an object is a particular instance (or constituent) of the class to which that object belongs.

For example, a programmer may define a Java class called *Branches*, which includes objects that consist of corporate branch offices. Furthermore, this programmer may define two methods that operate on objects of the *Branches* class: *List* and *int*.

The *List* method allows the programmer to write code that lists all of the objects that he or she has instantiated (or created) in a particular class. In this case, the list of objects that belong to the *Branches* class might include *HoustonBranch*, *NYBranch*, and *LondonBranch*.

The *int* method allows the programmer to write code that adds the total number of objects he or she instantiates for a particular class. For example, the sum of the *Branches* class objects would equal the number three. (For more information about Java objects and classes, complete the tutorial at <http://www.java.sun.com/docs/books/tutorial/java/javaOO>.)

Java classes make programming easier because programmers can use previously created classes to write new programs. For example, if a programmer wants to write a program that will, among other things, allow users within an organization to request a list of the organization's branch offices, the programmer can use the hypothetical *Branches* class described above.

Classes and Java Beans

A standard set of classes is included with the NetWare 5 Java Virtual Machine (JVM). (These classes are stored in the SYS:\JAVA\LIB\classes.zip file on the NetWare 5 server.) For example, the Frame class, as shown below, defines a frame in which Java objects (text strings, for example) can be displayed.

```
public class Frame {

    int THE_WIDTH ()
    int THE_HEIGHT ()

}
```

Because the Frame class is a standard class, a programmer would not ordinarily define this class. Rather, the programmer would import the Frame class. For example, this class has been imported into the following Java bean:

```
import java.awt.Frame;
import java.awt.ActionListener;
import java.awt.ButtonListener;

public class VolAdmin extends Frame
    implements ActionListener,
        NwBrowserChangeListener,
        ButtonListener
{

    //
    //omitted lines
    //
```

The more classes Java programmers create, the more previously created classes are available to other programmers. Consequently, it takes less time for programmers to write new programs, making Java programmers more efficient than other types of programmers. (For an example of Java classes, see "Classes and Java Beans.")

Because Java classes are reusable, they also tend to yield programs that have fewer bugs than programs written in other languages. When a programmer uses previously created classes in a program, the code that comprises those classes has already been debugged. Obviously, the more previously created (and, therefore, previously debugged)

```
NWBrowser      nwBrowser = null
NWEntry        parentEntry = null

//more omitted lines

setSize (THE_WIDTH, THE_HEIGHT);
setTitle ("VolAdmin");
setBackground (Color.lightGray);

//more omitted lines
{
```

The VolAdmin class defines the VolAdmin Java bean and extends the Frame class. In the Java programming language, extending a class means that the newly defined class, called a subclass, inherits all of the objects and methods contained in the originally defined class, called a superclass.

For example, the VolAdmin subclass inherits THE_WIDTH and THE_HEIGHT objects from its superclass, the Frame class. (A Java subclass also contains objects and methods that are not inherited from its superclass.) Although the VolAdmin bean contains only one explicit class statement, this Java bean implements (or uses) the following previously defined classes: ActionListener, NwBrowserChangeListener, and ButtonListener. In addition, the VolAdmin bean instantiates other Java beans (such as NWBrowser and NWEntry) as objects.

To view the unabridged code for VolAdmin and other Java beans, go to http://developer.novell.com/ndk/doc/samplecode/bns_sample, and click the segment of code you want to see. ●

classes the programmer uses in any given program, the fewer bugs that program is likely to have.

Where can programmers find these previously created Java classes (which they can then reuse in their own programs)? Several sources are available. For example, most JVMs, including the NetWare 5 JVM, ship with the classes.zip file, which includes Java classes created by Sun Microsystems. On a NetWare 5 server, the classes.zip file is located in the SYS:\JAVA\LIB directory.

Java programs themselves usually include Java classes. These classes are typically contained in files that have a .zip or .jar extension and are usually located in the directory in which the Java applica-

tion resides. For example, suppose the Xyz Java program was stored in the SYS:\JAVA\Xyz directory. The classes necessary to run this program may be contained in the Xyz.zip file, which resides in the Xyz directory or in one of its subdirectories.

BEANS, MACHINES, AND OTHER THINGS

Java classes are key constituents of the Java Application Programming Interface (API), which, in turn, is a key constituent of a particular Java implementation, such as the NetWare 5 Java implementation. The Java API uses class files to enable the Java implementation to access operating system level functions, such as threading and read-write functions. For example, Java classes which are located in the SYS:\JAVA\LIB\classes.zip file enable Java programs to call NetWare 5 operating system functions, such as multithreading or writing lines of text to a screen.

The NetWare 5 Java implementation also includes the following constituents:

- Novell JVM for NetWare
- Symantec Just-in-Time (JIT) compiler
- Java beans
- Java Database Connectivity (JDBC) API
- Java Naming and Directory Interface (JNDI) API
- Java servlet API

Novell JVM for NetWare

The Novell JVM for NetWare reads the compiled code contained in Java class files, including the class files in the classes.zip file. This compiled code is called *bytecode* or *JVM instruction set code*. The Novell JVM for NetWare interprets the instructions contained in the bytecode and then executes those instructions.

Symantec JIT Compiler

To improve the performance of Java programs, the Symantec JIT compiler translates the bytecode into the particular machine code that is meaningful to your computer's CPU. Because the Symantec JIT compiler performs this translation before a Java program runs, the program can run at top speed once it begins to execute.

To enable the NetWare 5 JVM to use the Symantec JIT compiler, you must first load the JVM on the NetWare 5

Chat With Java on NetWare 5

VolanoChat from Volano, L.L.C. is a Java application that enables you to set up chat rooms through which users can communicate in real time. For example, you can create a chat room that serves as an online help desk through which customers have real-time access to your company's customer service personnel.

Before you run VolanoChat, you must first install Netscape FastTrack Server on the NetWare 5 server. (Netscape FastTrack Server is included on the NetWare 5 installation CD-ROM. For information about other system requirements, see the System Requirements section of the VolanoChat Administrator Guide at <http://www.volano.com/guide21/requirements.html>.) You must then complete the following steps to install VolanoChat on the NetWare 5 server and then to run VolanoChat on that server.

INSTALL VOLANOCHAT ON THE NETWARE 5 SERVER

1. From your workstation, download the version of VolanoChat that Volano recommends for installation on Windows NT, 98, and 95. (To download a trial version of VolanoChat 2.1.1, go to <http://www.volano.com>, and click on Free Server Trial.)
2. Create a VolanoChat directory in the SYS:JAVA directory on the NetWare 5 server. For example, you may create the vchat2.1.2 directory.
3. Double-click the VolanoChat .exe file you downloaded on your workstation's hard drive.
4. Click OK to install the application.
5. When the installation program prompts you to select a directory in which to install VolanoChat, click Browse, and then select the directory that you created on the NetWare 5 server (SYS:JAVA\vchat2.1.2, for example).
6. Click Install.
7. After the installation is completed, locate the vcclient directory in the VolanoChat directory you created. Move this vcclient direc-

tory and the files contained therein to the public document directory on Netscape FastTrack Server (NOVONYX\SUITESPOT\DOCS\vcclient).

CREATE A NETWARE COMMAND FILE (NCF)

1. Create a text document that contains the following commands:

```
envset java_compiler=symcjit
envset CWD=SYS:JAVA\vchat2.1.2
envset classpath=$CLASSPATH;SYS:JAVA\vchat2.1.2
java COM.volano.Main
```

2. Save this document as a text (ASCII) file with the .NCF extension. For example, you could call the document Volano.NCF
3. Move this NCF to the NetWare 5 SYS:\SYSTEMS directory.

RUN VOLANOCHAT ON THE NETWARE 5 SERVER

To run VolanoChat on the NetWare 5 server, reboot the server, and type the name of the NCF at the server console. After VolanoChat is running on the NetWare 5 server, complete the following steps to initiate a VolanoChat chat session:

1. Make sure Netscape FastTrack Server is running on the NetWare 5 server on which VolanoChat is running. To start Netscape FastTrack Server, enter the following command at the server console:

```
nsweb
```

2. Launch the browser on your workstation, and enter the following URL: <http://xxx.xxx.xxx.xxx/vcclient/index.html>. (Replace xxx.xxx.xxx.xxx with the IP address of the NetWare 5 server on which VolanoChat is running.)
3. Click the Chat button located in the middle of the VolanoChat web page. (See Figure 2 on p. 14.)

server by typing LOAD JAVA at the server console. You then type the following command at the server console:

```
envset java_compiler=symcjit
```

Java Beans

Java beans are program components that consist of one or more Java classes (which in turn encompass one or more Java objects) and Java code called *messages*. (See "Classes and Java Beans" on p. 8.) Java messages are instructions—such as get and set—that allow objects from one class to interact with objects from another class.

Java beans make Java programmers even more productive by allowing them to assemble these ready-made components (along with non-bean Java code) to create a variety of applications. The program code that comprises a Java bean is typically located in a compressed file

that has a .jar extension. (To learn more about the enterprise Java beans Novell offers in the Novell Developer Kit, visit <http://developer.novell.com/ndk/bns.htm>.)

JDBC API

The JDBC and JNDI APIs require classes that are stored in .zip and .jar files. On a NetWare 5 server, these .zip and .jar files are part of Novell's Java Class Libraries (NJCL), which are located in the following directories on the NetWare 5 server: SYS:JAVA\LIB and SYS:JAVA\CLASSES.

The classes for the NetWare 5 JDBC API have Structured Query Language (SQL) statements and commands as objects. These statements and commands enable programmers to write Java programs that can access relational databases, which are located on the NetWare 5 server.

JNDI API

The JNDI API enables Java programs to access information contained in various naming systems. Naming systems operate on the notion that objects existing within these systems are bound to contexts that also exist within those systems. For example, in Novell Directory Services (NDS) a particular User object is bound to the NDS tree and Organizational Unit (OU) in which that User object exists.

The JNDI API provides access to naming systems such as the following:

- Naming systems used in NDS, X.500-compliant directories, and Lightweight Directory Application Protocol (LDAP) compliant directories
- Operating system file systems
- Domain Naming System (DNS)

JNDI API classes define provider-specific objects and





```

JAVA_HOME=SYS:\JAVA
OSA_HOME=SYS:\JAVA
MGMT_HOME=SYS:\PUBLIC\MGMT
CLASSPATH=SYS:\java\lib\classes.zip;SYS:\java\classes;.
CLASSPATH=%CLASSPATH%;%OSA_HOME%\lib\swing.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\lib\jg13.1.0.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\lib\help.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\lib\servertop.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\lib\jndi.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\lib\njcl.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\lib\vbjorb.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\lib\vbjapp.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\lib\vbjtools.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\lib\ucs.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\
CLASSPATH=%CLASSPATH%;%OSA_HOME%\beans\NWDDir.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\beans\NWSess.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\beans\NWPrctQldm.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\beans\NWSockets.jar
CLASSPATH=%CLASSPATH%;%OSA_HOME%\classes\HotJava1.1.5
CLASSPATH=%CLASSPATH%;%OSA_HOME%\classes\HotJava1.1.5\lib
CLASSPATH=%CLASSPATH%;%OSA_HOME%\classes\HotJava1.1.5\lib\classes.zip
CLASSPATH=%CLASSPATH%;%OSA_HOME%\classes\HotJava1.1.5\lib\HotJavaBean.jar

```

Figure 1. You can add CLASSPATH variables to the java.cfg file.

the methods by which these objects can be accessed. These provider-specific classes include the NetWare file system and NDS. Novell's provider-specific classes for the JNDI API are located in the njcl.jar file, which resides in the SYS:\JAVA\LIB directory on the NetWare 5 server.

The JNDI API also includes classes that define generic (or abstract) naming system objects and the methods by which these objects can be accessed. These classes define the framework for provider-specific classes and are located in the jndi.jar file that resides in the SYS:\JAVA\LIB directory on the NetWare 5 server. (For more information about how JNDI provides Java applications with access to naming-system information, see the white paper located at <http://www.developer.novell.com/whitepapers/jndi>.)

Java Servlet API

Programmers can use Java beans, the JDBC API, and the JNDI API in both servlets and non-servlet server-side Java programs. Servlets can access information from various other applications (via the JDBC and JNDI APIs, for example), including other Java applications.

For example, a programmer may write a servlet that allows a user to access personnel information, such as the beneficiary of a user's company-provided life insurance. To request this information, this user accesses and completes an HTML form via his or her web browser. A Java applet embedded in this form then delivers the user's information to the servlet.

The servlet, in turn, uses this information to determine what kind of information the user is requesting—in this case, the user's beneficiary. The servlet then uses the JNDI API to access NDS to verify that this user has the necessary rights to receive the information requested.

If the user has the necessary rights, the servlet uses the JDBC API to access the human resources database in which beneficiary information is stored. The servlet then creates an HTML web page to display the retrieved information and delivers this web page to the user's browser.

The classes that allow you to run servlets on a JVM have the .cla (short for class) file extension. (On the NetWare 5 server, these .cla files are located in the SYS:\JAVA\JAVAX directory.) Because servlets run on the NetWare 5 server and can access that server's resources, such as NDS, the Java servlet API contains built-in security features.

For example, the Java servlet API specifies that Java classes that are imported from outside the server upon which the servlet resides (such as Java classes that might be sent within a user's request for servlet-provided information) must be stored separately from the Java class files that reside locally. Because imported Java classes are stored in their own separate locations (called *name spaces*), an imported Java class cannot impersonate a local Java class to access confidential information stored on the NetWare 5 server.

In fact, before any imported Java code (including Java classes) can be processed, the NetWare 5 JVM subjects the code to a

process called bytecode verification. Bytecode verification is a series of tests that are implemented through Java classes. These tests ensure that Java code received from outside sources adheres to built-in Java security rules. (For more information about built-in Java security features, see <http://www.java.sun.com/docs/white/langenv/Security.doc.html>.)

HAVE SOME JAVA WITH NETWORKWARE 5

If you are not interested in writing a Java application, you may think you do not need to understand Java classes and the objects they comprise. Even if you are not interested in writing a Java application, however, understanding Java classes is important if you want to run many of the commercially available Java applications on your company's NetWare 5 server.

How to Serve a Typical Java Application

Although programmers can create Java applications without regard to the platform upon which these applications can run, network administrators who install Java applications must take into account the platform on which the application will run. For example, the platform upon which you run a Java application determines the storage and access methods used for that application's files, including executable files and the class files the application and the JVM call.

For example, an installation program that was written for Windows may try to install the Java application in a directory such as C:\Windows\Java. Obviously, this directory path does not exist on a NetWare 5 server. If the Java application's installation program is not written specifically for NetWare, you must create a directory for the application on the NetWare 5 server. You must then either install or copy the application files into this directory.

After the Java application files reside in a directory that is meaningful to the NetWare 5 server, you must let the NetWare 5 JVM know where to find the class files it needs to run the application. As a general rule, you can provide the NetWare 5 JVM with the information it needs in one of three ways:

- You can use Java envset commands each time you want to run the Java application.
- You can add the application's path to the java.cfg file, which resides in the

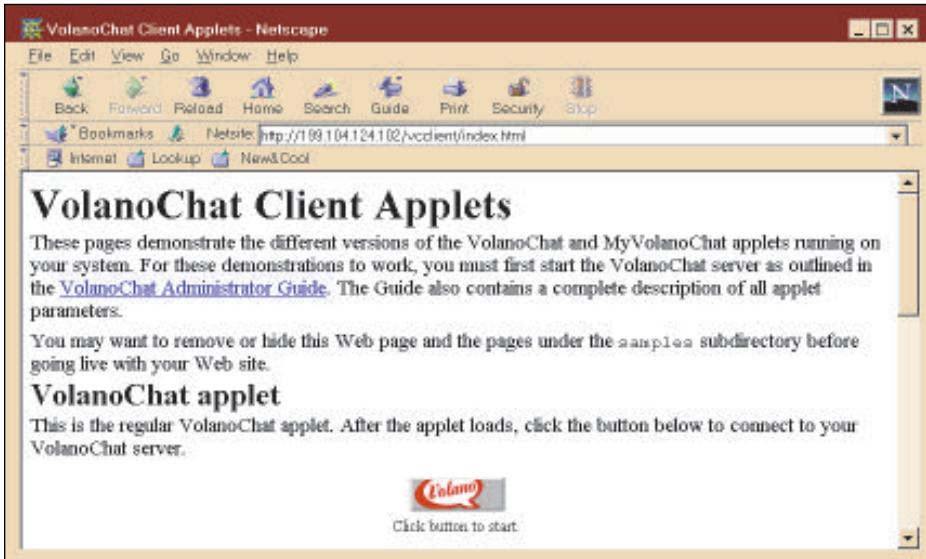


Figure 2. With VolanoChat, you can create chat rooms through which groups of users can communicate in real time.

SYS:\ETC directory on the NetWare 5 server.

- You can create a NetWare Command File (NCF) and copy it in the SYS:\SYSTEM directory.

A Good Solution

Like all Java commands that you type at the server console, envset commands are deleted each time you unload the NetWare 5 JVM. As a result, you must reenter envset commands each time the JVM is loaded. (For a complete list of standard Java commands, type java at the server console. For an extended list of Java commands that are specific to NetWare, type java -nwhelp at the server console.)

You can use the envset command to set two environment variables that allow the NetWare 5 JVM to find and to run Java applications:

- The Current Working Directory (CWD) variable
- The CLASSPATH variables

The CWD variable should be set to the directory that contains the Java application you want to run. For example, if the Java application is located in the SYS:\JAVA\Xyz directory, you should type the following command at the server console:

```
envset CWD=SYS:\JAVA\Xyz
```

The CLASSPATH variable should be set to include every class file that the NetWare 5 JVM calls to run a given Java

application, including the series of class files necessary to run the JVM itself. Typically, the CLASSPATH statement contains all of the application's class files (.cla), program files (.pro), and Java bean files (.jar), which are often located in the application's \LIB directory.

For example, suppose that most of the Xyz application's class files are contained in the classes.zip file and the Xyz application contains one Java bean in the XyzBean.jar file. To set the CLASSPATH variable, you would type the following command at the server console:

```
envset CLASSPATH=$CLASSPATH;
SYS:\JAVA\Xyz; SYS:\JAVA\Xyz\lib;
SYS:\JAVA\Xyz\lib\classes.zip;SYS:\
JAVA\Xyz\lib\XyzBean.jar
```

In this command, \$CLASSPATH is a shorthand way of including the current value of the CLASSPATH variable. This variable directs the NetWare 5 implementation to the class files it needs to run the JVM. (To view the series of CLASSPATH variables included in this shorthand statement, see Figure 1 on p. 12. These variables reside in the java.cfg file, which is located in the SYS:\ETC directory on the NetWare 5 server.)

To run the Java application, you then enter the Java command that launches the application. This command has the following structure:

```
java application <parameters> <application
startup command>
```

For example, you would enter this command to run the Xyz application:

```
java -ns COM.xyz.Main
```

In this command, the parameter -ns instructs the JVM to run the application in a new screen.

The Better Solution

If you don't want to type a lengthy CLASSPATH variable each time you run a Java application, you can add the application's CLASSPATH variable to the java.cfg file. (See Figure 1 on p. 12.) For the example above, you would add the following statements at the end of the CLASSPATH variable in the java.cfg file:

```
CLASSPATH=$CLASSPATH;$OSA_HOME\
classes\Xyz
CLASSPATH=$CLASSPATH;$OSA_HOME\
classes\Xyz\lib
CLASSPATH=$CLASSPATH;$OSA_HOME\
classes\Xyz\lib\classes.zip
CLASSPATH=$CLASSPATH;$OSA_HOME\
classes\Xyz\lib\XyzBean.jar
```

After you add an application's CLASSPATH variable to the java.cfg file, you must initialize the newly changed file by unloading the JAVA NetWare Loadable Module (NLM) and then reloading it. To unload the JAVA NLM, type UNLOAD JAVA at the server console. To load the JAVA.NLM, type LOAD JAVA. (For an example of adding an application's CLASSPATH variable to the SYS:\ETC\java.cfg file, see "How to Brew HotJava" on p. 16.)

If you add a Java application's CLASSPATH variable to the java.cfg file, you must enter a Java command at the server console to run the application. This command should include any special parameters the application needs to run. For example, in the case of the hypothetical Xyz application, you would type the following at the command prompt:

```
java -ns COM.xyz.Main
```

Unfortunately, using this method to run Java applications on a NetWare 5 server has two inherent drawbacks: First, adding even one application to the java.cfg file can add significantly to the length of that file, making java.cfg difficult to troubleshoot. Second, to run the application, you must remember its exact

How to Brew HotJava

HotJava Browser from Sun Microsystems is a server-side web browser. You can use HotJava Browser to create web-enabled applications such as applets.

To install Sun Microsystems' HotJava Browser on the NetWare 5 server and add this application's CLASSPATH variable to the java.cfg file, complete the following steps:

INSTALL HOTJAVA ON THE NETWARE 5 SERVER

1. From your workstation, download the version of HotJava Browser that Sun recommends for platforms other than Solaris for SPARC, Windows 95, or Windows NT. (You can find this free download at <http://www.java.sun.com/products/hotjava/1.1.5>.)
2. Create a HotJava Browser directory in the SYS:JAVA directory on the NetWare 5 server. For example, you could create the HotJava1.1.5 directory.
3. Add the following lines to the classes.zip file in the SYS:ETC directory on the NetWare 5 server:

```
CLASSPATH=$CLASSPATH;OSA_HOME\HotJava1.1.5.
CLASSPATH=$CLASSPATH;OSA_HOME\HotJava1.1.5\lib
CLASSPATH=$CLASSPATH;OSA_HOME\HotJava1.1.5\lib\
jndi.jar
CLASSPATH=$CLASSPATH;OSA_HOME\HotJava1.1.5\lib\
HotJavaBean.jar
```

RUN HOTJAVA ON THE NETWARE 5 SERVER

1. To run HotJava Browser, type the following commands at the server console:

```
unload java
load java
```

2. Then type the following command at the server console:

```
java -ms4m -mx32m -Dhotjava.home=SYS:JAVA\HotJava1.1.5
-Djava.home=SYS:JAVA\sunw.hotjava.Main
```

name and any special parameters that must be specified. (The names of Java executable files and class files are case sensitive.) Even if you have a picture-perfect memory and relish the idea of memorizing application startup commands, you probably don't want to make the java.cfg file any larger than it has to be.

The Best Solution

Fortunately, you don't have to add a Java application's CLASSPATH variable to the java.cfg file in order to run this application on the NetWare 5 server. There is an easier way. You can create an NCF that contains all of the information necessary to run the application.

To create an NCF, open any ASCII text editor (such as Notepad), and type the entire series of commands needed to run the application. (You must start each new command on a new line.) For example, if you created an NCF for the hypothetical Xyz application, this file would contain the following commands:

```
envset CWD=SYS:JAVA\Xyz
envset
CLASSPATH=$CLASSPATH;SYS:JAVA\
Xyz;SYS:JAVA\Xyz\lib;SYS:JAVA\Xyz\
lib\classes.zip;SYS:JAVA\Xyz\lib\
XyzBean.jar
java -ns COM.xyz.Main
```

After you have created the NCF, save it with a name that is easy to remember and

with an .NCF file extension (Xyz.NCF, for example). Then copy this file to the SYS:\SYSTEM directory on the NetWare 5 server. To run the application, you simply type its NCF filename (Xyz) at the server console. (For an example of using this method to run an application on a NetWare 5 server, see "Chat With Java on NetWare 5" on p. 10.)

IF ALL ELSE FAILS, READ THE INSTRUCTIONS

What happens if you try these three methods and still find it difficult to run a Java application on your company's NetWare 5 server? How should you troubleshoot the problem? First, you should search the application's files for a text file that contains documentation. If such a file exists, it may include important configuration information. For example, HotJava Browser from Sun Microsystems includes HotJava1.1.5.CMD, a text file that lists a series of parameters that must be set in the application's startup command. If these parameters are not set, HotJava Browser will not run. (HotJava Browser is a web browser application. For more information about this application, see "How to Brew HotJava.")

Second, you should search the application's files for .jar files that are not included in the application's current CLASSPATH variable. Although .jar files are often located in the application's \LIB directory, these files may be stored in other directories as well. All of the application's

.jar files must be included in the application's CLASSPATH variable, regardless of which method you are using to run the application on the NetWare 5 server.

Finally, you can usually get information about running a particular vendor's product on NetWare 5 by calling that vendor's support staff directly—especially if the product is Novell Yes, Tested and Approved. (For a list of Novell Yes, Tested and Approved Java applications, go to http://developer.novell.com/infosys/lv3_1962.htm, and then click Java Showcase.)

LOOKING AHEAD

In addition to explaining Novell's commitment to Java, Holbrook predicted that the world of network computing will look increasingly to server-side Java applications for answers to its computing problems. This prediction is tenable, to say the least. For example, many networking devices such as Internet-enabled cellular phones are becoming smaller and have less computing power than desktop workstations. Using server-side Java applications, these devices can deliver information that they would otherwise lack the computing power to deliver.

As the need for server-side applications grows, the number of software developers who write applications to meet this need will arguably grow in tandem. These developers will choose Java both for its platform independence and for the increased productivity that is a by-product of the Java language.



A Java Sampler

Company	Product	Product Description
BulletProof Corp. Voice: 1-800-505-0105 Fax: 1-305-933-1216 http://www.bulletproof.com	JDesignerPro 3.0	This Java application builder is integrated with Novell Directory Services (NDS) and Oracle and enables you to develop applications via a drag-and-drop capability. (A 90-day trial download for NetWare 5 is available at http://www.bulletproof.com . For NetWare-specific information, visit http://developer.novell.com/yes/50008.htm .)
Cereus Design Corp. Voice: 1-650-947-0880 Fax: 1-650-429-2029 http://www.cereus7.com	HotTEA 5.0 (Green TEA) HotTEA 5.0 (B.R.I.S.K. TEA)	This software enables you to quickly write applets and small applications using the Basic programming language. This software is a scripting kit that enables you to write complex Java applications using the Basic programming language. (An evaluation download is available at http://www.cereus7.com . For NetWare-specific information, visit http://developer.novell.com/yes/47695.htm .)
The Prestige Software International division of Computer Associates Inc. Voice: 1-201-592-0009 Fax: 1-201-585-6513 http://www.prestigesoft.com	Masterpiece/Net 1.1	This application provides web-enabled financial management tools for multinational, multiorganizational companies. This application includes numerous features including General Ledger, Accounts Payable, Accounts Receivable, Fixed Assets, and Inventory Control modules. (For NetWare-specific information, visit http://developer.novell.com/yes/49688.htm .)
D2K Inc. Voice: 1-888-770-4636 Fax: 1-408-451-2015 http://www.d2k.com	Tapestry 2.0	This application automates the data extraction, transformation, and loading processes needed to build and maintain data warehouses and data marts. Data sources include various mainframe and client-server databases, as well as PeopleSoft Enterprise Resource Planning (ERP) and SAP database applications. (For NetWare-specific information, visit http://developer.novell.com/yes/49590.htm .)
Global Communications Ltd. Voice: (0)171-306-7300 (UK) Fax: (0)171-306-7371 http://www.teamphone.com	Teamphone Web 8.0	This application provides users with a single number and a single corporate directory that is based on Lightweight Directory Access Protocol (LDAP). Business contacts can use this number and directory to reach users wherever they happen to be. (For NetWare-specific information, visit http://developer.novell.com/yes/49542.htm .)
Open Concept International Voice: 1-780-434-2363 Fax: 1-780-438-2941 http://www.ocii.com	openQuote 1.1	This application consists of client-server software that allows users to quote product and service prices to potential customers. You can implement this product over the Internet and/or over your company's intranet. (You can download a trial version from Open Concept's web site. For NetWare-specific information, visit http://developer.novell.com/yes/49569.htm .)
TRADEX Technologies Inc. Voice: 1-888-487-2339 Fax: 1-813-222-5658 http://www.tradex.com	TRADE'ex Market Maker 3.11	This application consists of e-commerce software through which commercial and noncommercial entities can create a web-based market place. (For information about new product releases, visit TRADEX's web site at http://www.tradex.com . For NetWare-specific information, visit http://developer.novell.com/yes/49901.htm .) ●

If you're looking for an application to answer a present need, it isn't too early to explore the variety of server-side applications already available. Novell's Java Showcase features more than 250 Java applications. In addition to checking out the Java applications that are Novell Yes,

Tested and Approved, you may also want to check out the applications that Sun Microsystems has certified as being 100 percent pure Java. (For a list of the Novell Yes, Tested and Approved Java applications, go to http://developer.novell.com/infosys/lv3_1962.htm, and click Java Show-

case. For a list of the 100-percent Java applications, go to <http://java.sun.com/100percent/latestlist.html>.) Applications that are 100 percent pure Java will run on any JVM, including the NetWare 5 JVM.

Cheryl Walton works for Niche Associates in Sandy, Utah. ●