Getting Started with Java

CodeBase® 6.2

DataBase Management for C, C++,

Java, Visual Basic and Delphi Programmers.

Sequiter® Software Inc.

Contents

Introduction

Congratulations on your purchase of CodeBase, one of the fastest, most flexible database development tools available for C, C++, Visual Basic and Delphi programmers and now available for Java. CodeBase provides xBase file compatibility, which gives you the freedom to use CodeBase with many other commercially available tools.

The purpose of this *Getting Start with Java* booklet is to provide a fast and easy way to start running CodeBase Java applications. This booklet explains in detail how to run the CodeBase server and make sure that it is running correctly and how to run a Java application that uses the CodeBase Java Application Program Interface (API). For information on how to create Java applications using the CodeBase API refer to the *CodeBase Java API Reference Guide.* The reference guide provides in depth coverage of the entire CodeBase Java API and examples that show how to incorporate the API in Java applications.

Award Winning

At the heart of CodeBase is our speedy, award-winning database “engine”, a library of well-tested, highly optimized routines designed specifically for accessing xBase files. Your applications attain instant xBase capabilities by simply integrating one of our libraries with your program code. Unlike other xBase solutions, you don’t need to learn another language to use CodeBase. The CodeBase API simply becomes an extension to your programming language of choice, using the syntax of that language.

Comprehensive

The types of applications you can develop with the CodeBase libraries are comprehensive—you can access FoxPro, dBASE IV, and Clipper data, index and memo files. You can create client/server applications. Your client applications can be written for DOS, Windows 32-bit or UNIX .

Performance

CodeBase also contains built-in functionality to enhance program performance at high and low levels.

Whether you are writing applications for DOS, Windows 32-bit or UNIX, using FoxPro, dBASE or Clipper, we are sure you will find that CodeBase meets your database development needs.

What’s New

Perhaps the most obvious change to CodeBase is that fact that the product now ships on CD-ROM instead of floppy diskettes. We didn’t do this just to follow the crowd however—the reason we need so much extra disk space for CodeBase is that we’ve rolled all previous language-specific versions of the product (CodeBase, CodeBase++, CodeBasic and CodePascal) including the CodeBase Java API into one major offering. Now with CodeBase you can now program in C, C++, Basic, Pascal or Java, all with the same product!

We feel this is an important benefit, as more and more developers are using more than one development tool (such as Visual C++ and Visual Basic) to create their applications. For example, If you need to write a C application for one project, and a Visual Basic application for another, the time you spend learning the CodeBase C API will pay dividends when it comes to learning the almost identical CodeBase Basic API. This type of synergy leads to faster application development, and enhances your skills as a developer by giving you the ability to quickly create database applications using a variety of development tools.

In addition to all this extra value for you, CodeBase has also added the following major new features.

- Addition of Java language Support. In addition to the CodeBase C, C++, Visual Basic and Delphi API's CodeBase provides an API that can be used in Java applications. Now the power of CodeBase can be tapped across many platforms and on the Internet. By using Java and CodeBase you can now build Web pages with full database capabilities.
- Addition of Client/Server Support. In addition to the stand-alone support found in previous editions of CodeBase products, CodeBase now includes full client/server capabilities. Now you create both types of applications with one product. Client/Server applications can bring a new level of security to your applications. Most of the time all you have to do to update a stand-alone application to client/server is to link with a different CodeBase library! Talk about easy.
- Support for Windows 95 and NT 3.5. CodeBase fully supports Microsoft’s latest 32-bit operating systems, Windows 95 and Windows NT 3.5.

- Speed and Stability Improvements. CodeBase has also undergone major internal enhancements to improve the speed, stability and flexibility of the library. Some of these changes are reflected in new CodeBase functions, other changes will be seen in better performance.

Manual Conventions

This document uses several conventions to distinguish between the various language constructs. These conventions are listed below:

CodeBase structures, constants, defines and functions are always shown highlighted:

Code4.connect()

Directory and file names are displayed in upper case:

CODEBASE\JAVA, EXAMPLE.JAVA

Configuration File Field names are displayed in italics:

ProcessId

Getting Started

What you Need

In order to use CodeBase for Java, you will need the Java Development Kit, as well as a basic understanding of Java. A reference guide on Java will also be helpful. Some database knowledge is also useful, but not essential.

The CodeBase Java API is used to create client applications. The CodeBase server must be installed under Windows NT or Windows 95. The installation program is a Microsoft Windows application.

Installation

The CodeBase installation program, SETUP.EXE, found on the root of the CodeBase CD, installs the CodeBase libraries, CodeBase server and at your discretion CodeUtil as well.

The installation program will offer you many choices regarding such items as what type of applications you wish to develop (stand-alone or client/server), which programming language you will use with CodeBase, what type of index-file format you require, your application type and target operating system, and many more such options.

Use the following instructions for installing CodeBase under Windows:

1. Using File Manager (Windows NT) or the Windows Explorer (Windows 95), locate the program item SETUP.EXE on the root directory of the CodeBase CD-ROM.
2. Double-click the SETUP.EXE program icon to launch the installation program.

The installation program will guide you through the various options available with CodeBase . The installation program may be used to install or reinstall all of CodeBase, or any parts thereof. If you have any problems during the installation, consult the on-line document INSTALL.TXT, located on the root drive of the CodeBase CD.

The following is a brief summary of the installation procedure for Java support:

1. The first step is to choose between stand-alone or client/server. Currently the CodeBase Java API only supports client/server, so you must choose this option.
2. The second step in the installation is choosing the directory in which CodeBase will be installed. CodeBase will be installed in C:\CODEBASE by default.
3. The third step is to choose the Application Target Type. The CodeBase Java API only supports 32-bit, so a 32-bit application target must be chosen.
4. The next step involves choosing the index-file compatibility. You have a choice of FoxPro, dBASE or Clipper.
5. By default, all the CodeBase API's are installed. If you just want Java support, then the Database server and Java language support options must be chosen. The other choices are optional.

By default, CodeBase is installed in a \CODEBASE directory on your hard drive. The libraries and support files related to the programming language you will use are installed in an appropriately named sub-directory as follows:

Programming Language	**Directory Created**
JAVA	CODEBASE\JAVA
C	CODEBASE\C
C++	CODEBASE\CPP
BASIC	CODEBASE\BASIC
PASCAL	CODEBASE\PASCAL

The following table lists the sub-directories that will be created during the installation of CodeBase with Java support. Some directories are only created based on certain installation options selected. All sub-directories are created under the main CODEBASE directory.

Sub-directory	Contents
\JAVA\EXAMPLES	CodeBase example programs for your programming language.
\JAVA\CLASSES	This directory contains the CodeBase '.java' source files and a file called CODEBASE.ZIP. The .zip file contains a directory called CODEBASE that contains the '.class' files for the CodeBase API.
\JAVA\JDK	This directory contains batch files and .txt files specific to the Java Development Kit. When other compilers are supported, new directories will be added to the \JAVA directory to support specific compilers.
\SERVER\32	The 32-bit database server and its accompanying support files are installed in this directory. This directory also contains the data, index and memo files used by the example programs found in the \JAVA\EXAMPLES directory.
\UTIL	CodeUtil transaction log file utility program and support files.
\BOOKS\JAVA	This directory contains the online help for the CodeBase Java API.

Electronic Documents

CodeBase supplies two forms of electronic documentation for Windows NT and Windows 95. There is a series of .TXT files that contain last-minute information, errata, etc. These files are found in various CodeBase directories, depending on their purpose. In addition, CodeBase offers online help that can be installed if desired.

.TXT Files

Of the many .TXT files contained in the CodeBase release, the one you should refer to first is README.TXT, located in the root of the CodeBase CD-ROM. This file contains any last moment changes to the product and documentation, and will direct you to other on-line .TXT document files as well.

Another import file is FAQ.TXT. This file contains Frequently Asked Questions of our tech support department. If you have a question that isn't answered in the Reference Guide, refer to this file. FAQ.TXT is installed in the programming language sub-directory. (e.g. CODEBASE\JAVA\FAQ.TXT).

Online Help

The online help is located in the CODEBASE\BOOKS\JAVA directory. Details on how to use the online help is in the file BOOKS.TXT, which can be found in the CODEBASE\BOOKS directory.

CodeBase Client/Server

This chapter discusses the server configuration and how to run the CodeBase server for the CodeBase Java API.

Client/Server Requirements

The following section provides information about the requirements for using the client/server configuration of CodeBase.

Although it is possible to develop CodeBase client/server applications on one machine, client/server applications generally run on two separate computers across a network.

In order to run a client/server application your machine needs to have communications protocol software loaded and the computer must be installed on a network of some type.

The CodeBase server is a 32-bit application that must be installed on a computer running Windows NT or Windows 95. The TCP/IP communications protocol must be installed on the computer.

In addition to an installed protocol, your client and server machines must also be connected to a network that supports the TCP/IP protocol. CodeBase should work on any network that supports that supports TCP/IP. You also need to know the IP (Internet Protocol) address or host name of the machine that the server resides on.

Steps for running the server

The following steps will help ensure that the server is running correctly so that client applications will be able to successfully connect to the server and execute without any problems.

1. Make sure that the TCP/IP communications protocol is installed on the server and client computers.
2. Configure the server.
3. Run the server.
4. Test the connection to the server by executing the D4TEST32.EXE or W4TEST32.EXE. These programs attempt to connect to the server.

Checking for TCP/IP

The following discussion on checking whether TCP/IP is installed assumes that the operating system involved is Windows NT or Windows 95. The Java client application might not be on a computer running Windows, since it is platform independent. Thus the steps involved in checking for TCP/IP may be different on the client machine than for the server machine.

Under Windows 95, follow these steps to check whether the TCP/IP protocol is installed.

1. Click the Start menu and choose the Settings|Control Panel option.
2. Double click the Network icon.
3. Click on the Configuration tab.
4. Look under the list of "network components installed" for TCP/IP.
5. If the TCP/IP protocol is not listed then install the protocol according to the documentation provided with Windows 95.

Under Windows NT 3.5x, follow these steps to check whether the TCP/IP protocol is installed.

1. Double click the Control Panel icon in the Program Manager.
2. Double click the Network icon.
3. Look for the TCP/IP Protocol under the "Installed Network Software" list.
4. If the TCP/IP protocol is not listed then install the protocol according to the documentation provided with Windows NT.

Configuring the Server

Client applications use the database server to handle all their data file requests. Upon execution, the database server attempts to find its configuration file (default name is S4SERVER.DBF). S4SERVER.DBF is installed in the same directory as the database server (CODEBASE\SERVER\32).

The configuration file contains fields that are used to determine how the server performs such tasks as connecting to clients, opening files, allocating memory, etc. These fields are discussed in detail in the "Configuration File" chapter of the *CodeBase 6.0 Getting Started* book.

Important Fields

The critical fields in the configuration data file that help the CodeBase database server connect to the clients are *Protocol* and *ProcessId*.

Protocol — The *Protocol* field specifies which communications protocol the server will use. The *Protocol* field must be "S4SOCK.DLL" for TCP/IP communications support.

ProcessId — The *ProcessId* field specifies the process identification (port number) of the server application. This field is only used with TCP/IP, and the field must be unique for all client/server applications running on a particular machine. The default value for this field is the arbitrary value '23165'. You typically don't need to change this value when calling the connection function unless a conflict occurs with another application on the server machine.

Another important field in the configuration file is *DefPath.* This field specifies the default directory for data, index and memo files used by the server. If you do not specify an explicit path when opening or creating data files, then the server looks in the directory specified by *DefPath*. *DefPath* is blank by default, and this tells the server to look in its current directory, which is the directory where the server executable is located.

You can modify the configuration file by changing its field values with a standard xBase tool or one of your own CodeBase applications. Alternatively, you can use the provided utility S4EDIT.EXE to update the file. This utility is described in the "Appendix A: S4EDIT Program" chapter of the *CodeBase 6.0 Getting Started* book.

Running the Database Server

After the server has been configured it can be executed. The usage for this program is:

```
S4SERVER.EXE [CONFIG_FILE]
```

CONFIG_FILE parameter specifies the name of the configuration file (the default extension is .DBF). If you do not use this optional parameter, the database server will open the configuration file S4SERVER.DBF.

The database server is a 32-bit Microsoft Windows program located in the CODEBASE\SERVER\32 directory. The server can be launched without command-line arguments by locating the program using the File Manager (Windows NT) or the Windows Explorer (Windows 95) and then double-clicking the program icon to launch the program.

If you want to specify a configuration file name when you run the server, you can use the Program Manager's Run command (Windows NT), or the Run command found in the Windows 95 Start menu. Alternatively, you can also start the server from the command-line of a DOS window.

```
C:\CODEBASE\SERVER\32> S4SERVER MYCONFIG
```

The above example will execute the 32-bit database server and specifies that it should use the configuration file MYCONFIG.DBF.

A second method for specifying command-line arguments is to set up a permanent program icon (Windows NT), or a Shortcut (Windows 95), that specifies the command-line arguments required.

Once you have the server running, you can then turn your attention to running client applications.

Testing the Network Connection

This section describes how to test the server. Before attempting to create your own client applications, you should verify that the current network hardware and software setup has been performed correctly. This step is simplified with some pre-built example test programs, all of which are located in the EXAMPLES sub-directory of the JAVA directory.

The test programs attempt to make a connection with a CodeBase server, by calling the CodeBase connection function **Code4.connect()**. When you run the test program, you specify connection information using a command line parameter in DOS, or interactively in Windows. The test program then attempts to establish the connection and if it cannot, it outputs an appropriate error message.

The following discussion assumes that the CodeBase database server is currently executing on a separate machine connected to the network. If the server and the client are running on the same computer, use the default host name "localhost".

DOS & Console Testing

Since, 32-bit operating systems such as Windows 95 have extended the services available in a DOS window to include the Windows API, it is possible to have a 32-bit character-based (console) application that can use the Windows Sockets API. Therefore we have a test program, D4TEST32.EXE located in the EXAMPLES directory, designed to test this type of application.

D4TEST32.EXE accepts up to two command-line arguments, the first being the name of the TCP/IP host machine on which the server is running, the second is a port number, which is equal to the *ProcessId* field of the server configuration file. If you don't specify the second argument, the default value of '23165' is used.

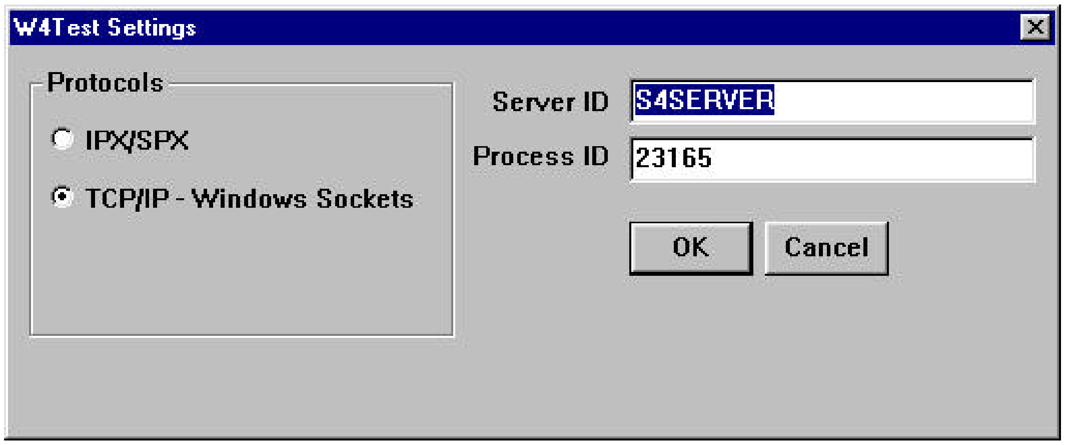
```
C:\CODEBASE\C\EXAMPLES> D4TEST32 JONS_COMPUTER 12345
```

In this example, D4TEST32 will attempt to find a server running on a computer that has the host name JONS_COMPUTER, using the port number 12345.

Windows Testing

There is a 32-bit Windows program W4TEST32.EXE, which is located in the EXAMPLES directory, that will attempt to connect to the server. You can execute this program from the Windows Program Manager by selecting the Run option from the File menu (Windows NT), or from the Start Menu (Windows 95).

When W4TEST32.EXE is executed, two windows will appear. Figure 1 is a screen shot of the topmost window. Make sure that the TCP/IP protocol has been chosen, the Server ID is set to the host name of the server and that the Process ID is set to the port number, which is equal to the *ProcessId* field of the server configuration file. Click the OK button when the settings are correct and the topmost window will disappear. Now click Start on the menu bar of the main window to test the connection to the server.


W4Test Settings

Protocols

IPX/SPX

[✓ TCP/IP - Windows Sockets]

Server ID: S4SERVER

Process ID: 23165

Figure 1

Testing Results

Both test programs use the database server to create the data file C4TEST.DBF and append two records. Each one will display its results on the screen. If these programs fail, they will indicate the problem using error messages. Keep in mind that the test data file will be created on machine where the server is running.

If the applicable tests were successful, you will know that your database server and communication protocols are correctly configured.

Connection Problems

If you cannot connect to the server using one of the supplied test programs, check that all the following points are true:

1. The TCP/IP protocol is installed on the server and client computers.
2. The CodeBase database server is up and running.
3. You have selected the correct port number for the host machine that the server in running on. The port number is equal to the *ProcessId* field of server configuration file.
4. The host name specified in the example matches the IP address or host name of the machine that is running the server.

Programming in Java

This chapter discusses how to run an example Java application that uses the CodeBase API.

The following discussion assumes that the Java Development Kit is being used with CodeBase and that the client application is running on a Windows 95 or Window NT machine.

On-line Documents and Batch files

An additional on-line documentation file exists for Java programmers, COMPILER.TXT. This file can be located in the CodeBase sub-directory for any installed compiler. For example, if you selected support for the Java Development Kit during installation, you can find the COMPILER.TXT file for this compiler in CODEBASE\JAVA\JDK.

COMPILER.TXT contains tips and solutions to common problems encountered when using CodeBase with Java. It also specifies exactly which version(s) of the compiler is supported by CodeBase.

The compiler directory also contains useful batch files. For example CodeBase provides the batch file C4AP.BAT, which may be used to compile and run an application. A C4AP.BAT batch file is located in the compiler directory CODEBASE\JAVA\JDK. As other compilers are supported, additional compiler directories will be added to the JAVA directory and appropriate batch files will be located in those directories. Directions on how to use the batch file are given when you run the batch file without any parameters.

Running An Example

This section documents how to get up and running with CodeBase by showing you the steps involved in compiling a small example. Before you proceed with this section you should already have installed CodeBase and selected the Java programming language option and started the CodeBase database server.

Here are the basic steps required to compile and interpret a CodeBase Java application.

1. Make sure the CodeBase 32-bit server is running. (Refer to the "CodeBase Client/Server" chapter for details)
2. Set the CLASSPATH and PATH environment variables.
3. Switch to the CodeBase EXAMPLES directory.
4. Compile example.
5. Run the program with the Java interpreter to verify that everything worked.

Location of data files

The data, index and memo files that the example applications use are located in the CODEBASE\SERVER\32 directory. By default the server looks for data files in its current directory, which is the directory where the server executable is located. Refer to the section "Configuring the Server" in the "CodeBase Client/Server" chapter for information on how to change the default directory.

Run the CodeBase Server

The server is installed in the CODEBASE\SERVER\32 directory. Run the S4SERVER.EXE to start the server. Under DOS, enter the following commands.

```
C:\> CD \CODEBASE\SERVER\32
C:\CODEBASE\SERVER\32> S4SERVER
```

Refer to the "CodeBase Client/Server" chapter for complete details on how to run the server successfully.

Set the PATH environment variable

In order to run the Java compiler and the interpreter from the JDK you must first specify where the executables for the compiler and interpreter are located. Specify the path of the compiler and interpreter in the PATH environment variable. The following example illustrates how to set the PATH under DOS. The example assumes that the compiler and the interpreter are located in the C:\JAVA\BIN directory.

```
C:> SET PATH=C:\JAVA\BIN
```

If you want to permanently specify the path under Windows, set the PATH environment variable in your AUTOEXEC.BAT

Set the CLASSPATH environment variable

The Java compiler and interpreter need to know the location of the Java classes and the CodeBase classes. The Java interpreter must also know the location of the compiled classes that it runs. Create an environment variable CLASSPATH that points to the location of the Java classes, CodeBase classes and any other compiled classes. The paths are separated by semi-colons under Windows.

The following example shows how to set the CLASSPATH under DOS and it assumes the following is true:

- Any compiled classes that are run by the interpreter are located in the current directory. The current directory is specified by a period.
- The CodeBase classes are located in the default installation directory.
- The Java classes are located in the C:\JAVA\LIB\CLASSES.ZIP file.

Note

Make sure that you list the path of the Java classes last when specifying the environment variable CLASSPATH. At the time when this document was written the Java compiler would not look for files in directories specified after the Java classes.

```
C:> SET CLASSPATH =.;C:\CODEBASE\JAVA\CLASSES\CODEBASE.ZIP;C:\JAVA\LIB\CLASSES.ZIP
```

Your classes may be located in different directories, so set your CLASSPATH accordingly.

If you want to permanently specify the paths under Windows, set the CLASSPATH environment variable in your AUTOEXEC.BAT.

Switch to the EXAMPLES directory

At this point you should switch to the CodeBase EXAMPLES directory for the compilation of one of the supplied examples. For your own projects, it is recommended that you create a separate working directory to help keep programming jobs separate from one another.

Under DOS, enter the following command:

```
C:\CODEBASE\SERVER\32> CD   \CODEBASE\JAVA\EXAMPLES

C:\CODEBASE\JAVA\EXAMPLES>
```

Compile an Example

To compile the example EXAMPLE.JAVA use the **javac** command. If you do not set the CLASSPATH environment variable, the compiler has a -classpath option which can be used to specify the paths of the Java classes and CodeBase classes.

```
C:\CODEBASE\JAVA\EXAMPLES> JAVAC EXAMPLE.JAVA
```

The Java compiler compiles the EXAMPLE.JAVA program and creates a EXAMPLE.CLASS file, which it places in the current directory.

Run the Example

Once the example program EXAMPLE.JAVA has been compiled, the Java interpreter **java** runs the class EXAMPLE.CLASS. In this case, the '.class' extension is not specified. If you do not set the CLASSPATH environment variable, the interpreter has a -classpath option which can be used to specify the paths of the Java classes and CodeBase classes.

Note that the name “Example” is case sensitive in the following command:

```
C:\MYDIR> JAVA Example
```

Note that the steps presented here are not the only way to compile an example, they are simply the easiest to explain. How you compile and run an application depends on your compiler, and whether you are compiling from the command-line, or from a programming environment.

Building the CodeBase Classes

This section provides information how to re-build the CodeBase classes should you wish to change the CodeBase '.java' files, which are located in the CODEBASE\JAVA\CLASSES directory.

All pre-built CodeBase class files are located in the CODEBASE\JAVA\CLASSES\CODEBASE.ZIP file. Within the CODEBASE.ZIP file is a directory called CODEBASE that contains the '.class' files.

A batch file C4BUILD.BAT, which is located in the CODEBASE\JAVA\JDK, has been provided to re-build the class files.

Note that the batch file will not zip up the rebuilt '.class' files. Instead, the batch file will rebuild the CodeBase classes and place them into a directory called CODEBASE under CODEBASE\JAVA\CLASSES.

The following batch file recreates the Java class files.

```
C:\CODEBASE\JAVA\JDK> C4BUILD
```

For information on how to store the CodeBase classes in a zip file, refer to ZIPINFO.TXT, which is located in CODEBASE\JAVA directory.

Running Java Applets

The previous chapters gave general instructions on how to run a Java application and the server. This chapter explains how to run and test a Java applet.

Using Java Applets

An applet is a form of Java program that can be placed on a Web page. When a Web page containing an applet is loaded into a Web browser that supports Java, the applet is automatically executed. All subclasses that appear in applets must be declared as public. Consequently, the name of the '.java' source file must be the same as the public class.

The steps involved in testing an applet are as follows:

1. First compile the applet '.java' file as discussed in the "Programming In Java" chapter.
2. Set the CLASSPATH and PATH environment variable if you are using the **appletviewer**.
3. Create a '.html' file that incorporates the applet.
4. Run the CodeBase server as discussed in the "CodeBase Client/Server" chapter.
5. Run the JDK **appletviewer** or a Web browser that supports Java with the '.html' file to test the applet.

Compiling the Applet

Since the applet is a '.java' file, the applet is compiled in the same way as any Java application. Use the **javac** command from the JDK to compile the applet to create a '.class' file as discussed in the "Programming In Java" chapter.

Provided on-line is an applet example called APPLETEG.JAVA, which is located in the EXAMPLES directory. It has already been compiled and the resulting APPLETEG.CLASS file is also located in the EXAMPLES directory.

Set the CLASSPATH and PATH

If you are using the **appletviewer**, you must set the PATH environment variable to point to the location of the **appletviewer** executable. Set the CLASSPATH environment variable to point to where the CodeBase classes are located, as discussed in the "Programming with Java" chapter, so that the **appletviewer** will know where to look.

Creating a '.html' file

In order to run an applet on a Web page, it must be incorporated in a Web page. Web pages are created using HTML , which stands for Hypertext Markup Language. An applet can be incorporated into a '.html' file by using the <APPLET> tag, which tells the Web browser that the Web page contains an applet to execute.

The <APPLET> tag contains ten attributes, the most important being CODE, WIDTH and HEIGHT, which must be specified, the others are optional. The CODE attribute specifies the compiled Java applet that is to be executed. The default size of the window, in pixels, required by the applet is specified by the WIDTH and HEIGHT attributes. The </APPLET> tag signals the end of the applet instruction in the '.html' file. For more information about the <APPLET> tags consult a Java language reference guide.

An example '.html' file called APPLETEG.HTML has been supplied and it is located in the EXAMPLES directory. The following is the listing of APPLETEG.HTML, which runs the applet APPLETEG.CLASS.

```
<title>Hello, Applet!</title>
<hr>
<applet code=appleteg.class width=300 height=150>
</applet>
<hr>
```

Running the CodeBase server

If an applet is being executed by a Web browser and the applet connects to a CodeBase database server, then the server must be running on the same computer from which the applet itself was loaded. This is due to the applet security restrictions inherent in many Web browsers.

If the applet is just being tested using the **appletviewer**, the location of the server is not restricted. The applet and the server may be on different computers linked together on a network.

Instructions on how to run the CodeBase server is discussed in the "CodeBase Client/Server" chapter of this guide.

Testing the Applet using the appletviewer

The applet can tested by using the **appletviewer** that comes with the JDK. To run the applet using the JDK, just use the command **appletviewer** with the name of the '.html' file.

```
C:\CODEBASE\JAVA\EXAMPLES> APPLETVIEWER APPLETEG.HTML
```

When the **appletviewer** runs the '.html' file, a window pops up that contains the graphical interface of the applet. Under Windows 95 the following window pops up when the APPLETEG.HTML applet is run with the **appletviewer**.

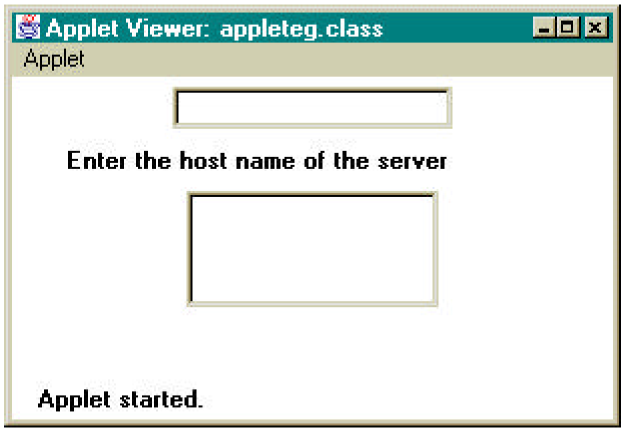

Applet Viewer: appleteg.class
Applet
Enter the host name of the server
Applet started.

Figure 2

The applet waits for the user to type in the host name of the CodeBase database server and then the applet connects to the server specified, opens a data file and prints out the first four records. This applet assumes that the port number of the server is equal to the default value 23165 and that the data file is located in the server's current directory.

Testing the Applet using a Web browser

The applet can be tested by using a Web browser that supports Java. When the APPLETEG.HTML file is opened with the Web browser, the applet is automatically started and the applet interface is created on the Web page.

The applet interface that appears on the Web page looks very much like the one created by **appletviewer** in Figure 2 and it works the same way. The applet waits for the user to type in the host name of the CodeBase database server and it connects to that server. The applet then opens a data file and prints out the first four records.

Note that the CodeBase classes and the applet class specified by the '.html' must be put in location accessible to the Web browser. The easiest way to achieve this is to copy the CodeBase classes and the applet class into the directory that contains the '.html' file. The Web browser will look in the current directory for the applet class and the CodeBase classes.

Note

The file WEBINFO.TXT, which is located in the CODEBASE\JAVA directory, contains important information on how to get client browsers to run a Web page containing an applet successfully. The WEBINFO.TXT discusses where you should put the applet class and the CodeBase classes in relation to your web page '.html' file. The text file also has information about where you should run the CodeBase database server so that the applet can connect to the server.

Where to Go From Here

This booklet describes how to run the CodeBase server and client applications, so the next step is learn how to incorporate the CodeBase Java API into a Java application. The *CodeBase Java API Reference Guide* discusses how to use the CodeBase API and it provides a complete listing of the classes and methods that make up the API. The *Reference Guide* also provides many examples that illustrate how to use the CodeBase API.