Using CLASSPATH & Other APIs

Prepared by Jeff Hunter, Sr. DBA *26-AUG-2002*

Overview

For many beginning Java developers, understanding and configuring the CLASSPATH has proved to be a significant challenge. Similar to the way the PATH variable is used by the operating system to find executable programs, the CLASSPATH setting is used by the several Java tools (`javac`, `java`, etc.) to find classes.

The class path tells SDK tools and applications where to find third-party and user-defined classes -- that is, classes that are not Java extensions or part of the Java platform. The class path needs to find any classes you've compiled with the `javac` compiler -- its default is the current directory to conveniently enable those classes to be found.

Java 2 SDK, the JVM and other SDK tools find classes by searching the following locations in the following order:

1. Java platform (bootstrap) classes
2. Any extension classes
3. The class path setting

Class libraries for most applications will want to take advantage of the extensions mechanism. You only need to set the class path when you want to load a class that's:

1. Not in the current directory or in any of its subdirectories
2. Not in a location specified by the extensions mechanism

Classes can be stored either in directories (folders) or in archive files (`.jar` and `.zip` files). The Java platform classes (the RunTime Stuff) are stored in `rt.jar`.

Setting the CLASSPATH

The class path can be set using either the `-classpath` option when calling an SDK tool (the preferred method) or by setting the `CLASSPATH` environment variable. The `-classpath` option is preferred because you can set it individually for each application without affecting other applications and without other applications modifying its value.

```
% sdkTool -classpath classpath1;classpath2...
```

-or-

```
% set CLASSPATH=classpath1;classpath2...
```

where:

sdkTool
A command-line tool, such as `java`, `javac`, or `javadoc`

classpath1:classpath2
Class paths to the `.jar`, `.zip` or `.class` files. Each classpath should end with a filename or directory depending on what you are setting the class path to:

- For a `.jar` or `.zip` file that contains class files, the class path ends with the name of the `.zip` or `.jar` file.
- For `.class` files in an unnamed package, the class path ends with the directory that contains the `.class` files.
- For `.class` files in a named package, the class path ends with the directory that contains the "root" package *(the first package in the full package name)*.

Multiple class path entries are separated by semi-colons in Microsoft Windows and colons in UNIX. With the set command, it's important to omit spaces from around the equals sign (=).

Classpath entries that are neither a directory nor an archive (`.zip` or `.jar` file) are ignored.

The Default the CLASSPATH Setting

The default class path is the current directory. Setting the CLASSPATH variable or using the `-classpath` command-line option overrides that default, so if you want to include the current directory in the search path, you **must** include "." in the new CLASSPATH setting.

Using the `-classpath` Option

The SDK tools `java`, `jdb`, `javac`, and `javah` have a `-classpath` option, which **replaces** the path or paths specified by the `CLASSPATH` environment variable while the tool runs. This is the recommended option for changing class path settings, because each application can have the class path it needs without interfering with any other application.

The runtime tool `java` has a `-cp` option, as well. This option is an abbreviation for `-classpath`.

Specification Order

The order in which you specify multiple class path entries is important. The Java interpreter will look for classes in the directories in the order they appear in the class path variable.

```
C:> java -classpath C:\java\MyClasses;C:\java\OtherClasses ...
```

In the example above, the Java interpreter will first look for a needed class in the directory `C:\java\MyClasses`. Only if it doesn't find a class with the proper name in that directory will the interpreter look in the `C:\java\OtherClasses` directory.

Using Extensions or Other Packaged APIs

As you continue to develop and grow you application library (APIs) so can the length of your CLASSPATH (or –classpath option). Starting with Java 2, you are no longer required to list each JAR file in your CLASSPATH variable. You can simply copy the JAR file into your `$JAVA_HOME/jre/lib/ext/` directory. Starting with Java 2, the runtime looks in this directory for any and all JAR and zip files, so no changes are needed to your CLASSPATH to access these archive files. This directory is commonly referred to as the Java Extensions Mechanism.