JavaPLT Developer's Guide

Eric Allen
eallen@cs.rice.edu
Rice University
6100 S. Main St.
Houston TX 77005

February 1, 2003

1 Introduction

This document explains to members of the Rice University JavaPLT development team how to set up your development environment for `javaplt` projects. To do so, all you will need is a basic understanding of simple `cvs` commands (*e.g.*, `checkout`, `update`, `commit`, etc.)[1], an account on the Rice CS network, and a computer running a Unix-based shell[2] with a cvs client program and CS network access (local or remote). You can install a `javaplt` environment on a Windows machine by first downloading Cygwin.

Note: Setting up a JavaPLT development environment is not necessary for working with the DrJava code base. DrJava is an open source project and is therefore not dependent on any proprietary tools for development. Nevertheless, many of the tools in the standard `javaplt` environment are general-purpose Java tools and can be convenient for working with DrJava as well other Java projects. For information on downloading the DrJava source code, see `http://drjava.sf.net`.

2 The JavaPLT CVS Repository

The various components of the JavaPLT environment are all kept under cvs control in the `javaplt` account on the CS network. You can access this repository

[1] A short tutorial on CVS is available online at `http://www.cvshome.org/docs/manual`.
[2] All shell commands in this document are written in bash syntax.

from any machine with access to Rice CS. To do so, set your CVSROOT to `<user-name>@cs.rice.edu:/home/javaplt/.cvsroot` and set your `CVS_RSH` variable to `ssh`. Then perform the following steps:

1. In your home directory, create a new subdirectory called `javaplt`.

2. In your new `javaplt` directory, checkout the following `javaplt` modules[3] :

   ```
   $ cvs checkout lib
   $ cvs checkout bin
   $ cvs checkout packages
   ```

   The `lib` module contains two files: `.bashrc` and `.cshrc`. Depending on what shell you are using, you should add a command to your own shell startup file to execute the corresponding file in this directory. Doing so will modify your environment to include all `javaplt` shell scripts (contained in the `bin` module) on your path, set your `CLASSPATH` to include `javaplt` classes, and set a default version of Java (using the `pickjava` command, in the `bin` module, discussed below).

   The `bin` module contains several convenience scripts for developing Java code. Some of these scripts are extremely powerful; before using one, be sure you know what it does.

   The `packages` module contains several useful Java tools for manipulating Java source and inspecting bytecode.

   Before continuing, start a new shell so that your path is updated with the new scripts.

3. At your discretion, you may want to install the following modules as well:

   ```
   $ cvs checkout java/<os-arch>
   $ cvs checkout public_html
   ```

   Where `<os-arch>` refers to the OS and architecture of your machine. Provided that the `bin` scripts are now on your path, you can find out exactly what this text should be by executing the `os-arch` script.

[3] If you are setting up a `javaplt` environment on the local cs network as opposed to a remote machine, then instead of checking out all `javaplt` modules discussed in this document, you might want to set symbolic links to some of the checked out copies in `/home/javaplt`, especially `/home/javaplt/java/<os-arch>` (discussed below). Doing so will make setup faster and easier, and it will save space, but it will also prevent you from modifying the modules and checking in new code.

The java/<os-arch> module contains several installation scripts for various versions of the JDK on your platform. If you choose to download this module, you should execute the downloaded installation scripts, installing the JDKs in java/<os-arch> directory. IMPORTANT: Whether you choose to download these scripts or not, you should place the various JDKs for your platform in java/<os-arch>, since some of the scripts in bin will look for JDKs in this directory. If you don't want to move the location of a JDK, just use a symbolic link.

Once you have installed JDKs in your java/<os-arch> directory, you can switch the version of Java you're using between them by executing

```
$ . pickjava <version>
```

Where <version> is the subdirectory under javaplt/java/<os-arch> in which the JDK you want to switch to is installed. For convenience, it is suggested that you rename the subdirectories in which JDKs are installed to simple names (e.g., 1.3, 1.4, etc.).

The public_html module contains the html source for

```
http://www.cs.rice.edu/~javaplt
```

At that website, you will find information relevant to JavaPLT developers, such as coding standards and the LaTeX source for this document. You will want to check out the public_html module if you intend to alter this website. NOTE: The live version of the website is a checked out copy available (by symbolic link) at /home/javaplt/public_html. To make a change to the live copy, you must have write access to the javaplt account. First alter your checked out copy. Then, when you are satisfied with your changes, commit your code, log in as javaplt, and update the copy in /home/javaplt.

3 Conclusion

Once you've checked out the modules above, you're ready to work on javaplt code. Be sure to try out the convenience scripts in bin. To check out the Java code for a project you want to work on, see your project-specific documentation.