Jagacy Software
www.jagacy.com

Jagacy VT

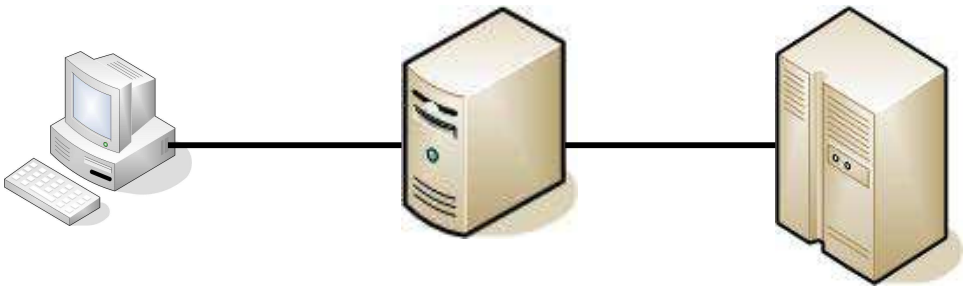
Jagacy VT is a feature rich VT100/220 terminal emulator and screen-scraping library written entirely in Java. It supports SSL, ANSI and XTERM protocols. The emulator is designed to help create screen-scraping applications. Jagacy VT screen-scraping is faster, easier to use, and more intuitive than HLLAPI. It excels in creating applications reliably and quickly.

Jagacy VT requires Java 1.5.0 or higher.

1. Introduction

Screen-scraping allows a user to retrieve information from a mainframe without using a VT100 terminal. It acts as if it is an automated terminal, sending keys and "scraping" information off the mainframe pages. Jagacy VT can be used in a stand-alone application, in an applet, or in an N-tier client/server environment, where the screen-scraping server is the data access tier:

N-Tier Architecture



Presentation

Functionality/
Data Access

Data

2. Features

Jagacy VT is configured using properties. Property values that begin and/or end with spaces can be quoted. Properties are discussed further in the next section.

Jagacy VT supports getting text at specified coordinates. It also supports writing keystrokes and waiting for a specific or general change to occur on the screen. Please refer to the Javadocs for more information.

Jagacy VT also comes with a VT100/220 emulator (Swing VT). This emulator indicates row/column coordinates for any character. A Swing VT window can also be displayed while the screen-scraping program is running. Swing VT is discussed in detail in a later section.

3. Properties

Pages change: text is added, deleted, and moved. Timeouts change when mainframes are moved or network equipment is replaced. Jagacy VT provides methods for reading row/column and timeout information from properties files (please see the Javadocs for more detail). If one of these changes, the property file can be changed without recompiling code.

Jagacy VT reads the property files from the current working directory (unless `jagacy.properties.dir` is set, see below). If one of the files does not exist, it skips it. Jagacy VT reads the properties in the following order:

1) jagacy.properties,
2) <SessionName>.properties,
3) System properties

If a property exists in two places, it is overwritten by the second occurrence in the above order. If you would rather not use properties to specify fields, coordinates, and timeouts, there are methods that allow this too.

If the System property `jagacy.properties.dir` is set, Jagacy VT will read the properties files from the specified directory. If the property is set to `classpath`, the CLASSPATH will be searched for the properties file(s) (for the -jar command line option use `jagacy.class.path`). This property can only be specified on the command line or with `System.setProperty()`. If this property is set, at least one of the properties files must exist in the specified directory.

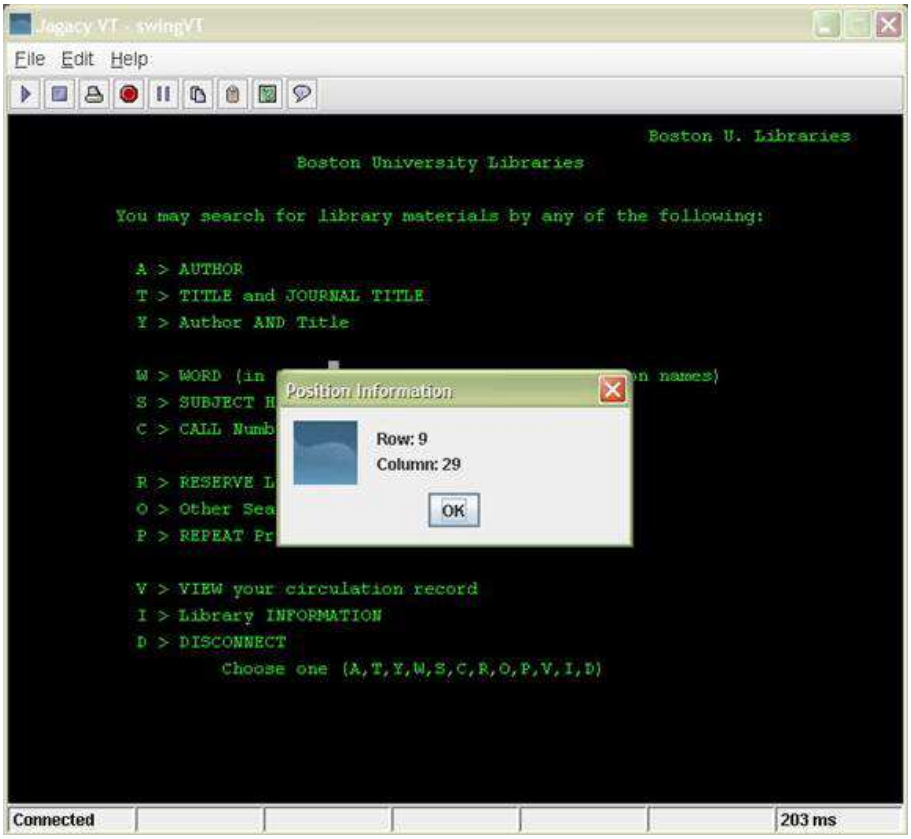
In addition to method properties, Jagacy VT supports the following properties:

Property	Default Value	Allowed Values
jagacy.host	none	Any host name or IP address.
jagacy.port	23	Any valid port number.
jagacy.terminal	VT100	VT100 DEC-VT100 VT220 DEC-VT220 ANSI XTERM
jagacy.class.path	empty string	%CLASSPATH% (for Windows) "$CLASSPATH" (for Linux/Mac OS X)
jagacy.ssl	false	true false
jagacy.ssl.keyFile	empty string	Valid key file directory and name
jagacy.ssl.keyPassword	empty string	Valid key password.
<SessionName>.logLevel (activated if log4j=false)	error	trace info debug warn error fatal
<SessionName>.logFile (activated if log4j=false)	out	out – System.out err – System.err Any valid file name
<SessionName>.log4j	false	true or false
<SessionName>.window	false	true or false
<SessionName>.answerback	empty string	Valid answerback string.
<SessionName>.autowrap	true	true or false
<SessionName>.newlineMode	true	true or false
<SessionName>.mainTextReceiveTimeout	1000	Amount of time (in milliseconds) to wait for text to echo.
<SessionName>.trailingTextReceiveTimeout	10	Amount of time (in milliseconds) to wait for trailing characters after text is echoed.
<SessionName>.tabAfterLabel	true	true if writeAfterLabel should use TAB; false otherwise.

where <SessionName> is the name of the Session specified when a `SessionVt` object is constructed. Properties can be specified in either property file.

4. Swing VT

Swing VT is a VT100/220 emulator tailored to developing and debugging screen-scraping applications:



It can be run from the command line as follows:

```
swingVT
```

for Windows, and:

```
swingVT.sh
```

for Linux and Mac OS X.

Swing VT allows the user to specify the screen update speed (and pausing the screen), to catch intermediate screens that are not normally seen during VT100/220 operation, but must be considered during screen scraping. Use the mouse to right click on any character to find out its row and column.

This information (along with the included examples in the src directory) should serve as a guide for writing screen-scraping applications. For further assistance, please contact support@jagacy.com.

5. Getting Started Quickly

The following description will demonstrate how easy it is to create a screen scraping application quickly. It uses pseudo-code to represent the steps necessary to implement code rapidly.

The logon Method

This method should sign in and navigate to the "base" page. The base page is the page where all queries, updates, inserts, and deletes originate from.

The logoff Method

This method should be able to navigate back to the sign off page from anywhere within the session. This is usually accomplished by using shortcut commands and keys that the mainframe programmers have placed in the application.

The processing Method

This method should navigate from the base page to the target page, perform queries, inserts, deletes, and/or updates, and navigate back to the base page.

Each method described above uses the same pseudo-code to navigate through pages:

a) Check for the unique string on a page. If the page is the target page, then finished.
b) Enter text and/or keys to navigate to the next appropriate page.
c) Call a wait method.
d) Goto a).

That's it! For a Java example of how this works, please see Example_a in the src subdirectory.

6. Tags and Trailing Timeouts

Unlike 3270 packets (which are page oriented), VT100 packets are stream oriented. This means that, although a unique string has been recognized on a

page, the entire page might not have been received. Tags and trailing timeouts take care of this.

Tags allow the developer to find a string in the raw VT100 data stream. By setting the property:

```
<SessionName>.logLevel=watch
```

(where is the `<SessionName>` is the name of the session), the developer can see the last escape sequence received for a page. This can be used as a tag in a `waitForChange` method.

Trailing timeouts “mop up” any remaining characters found after a tag is found, the mainTimeout expires, or a change occurs. These are a little harder to determine. The best way is a binary search: start with one second, see if the entire page is received, divide by two, and repeat the process. Once the developer has found the minimum trailing timeout, s/he should double it to allow for unforeseen circumstances.

If you use one of the `waitForCursor` methods you don’t have to worry about trailing timeouts.

7. Hints

When writing a screen-scraping application, please keep the following in mind:

- Run through a session using Swing VT. Note all text, keys, unique strings (with coordinates), and amount of time it takes to navigate through each screen.
- Extend `SessionVt` and provide logon and logoff routines.
- At startup, you should navigate to a base page in the mainframe application. From this page, any necessary pages should be navigable. This will save time during data retrieval.
- Always check to make sure the application is on the correct page, by checking for a string unique to that page. This can be accomplished using Jagacy VT’s wait functions.
- Determine the amount of time it takes for a page to appear and double it. Use the response time at the bottom right of the Swing VT screen as a guide. This will allow for unforeseen delays in the future.
- Find out from the mainframe application developer any shortcut keys for the application.

- Always close the session and logoff the userid. This will prevent the userid from being locked. Consider using state variables to indicate what page the application is on, in order to reverse the navigation and logoff. Jagacy VT automatically handles SIGTERM and SIGINT (Ctrl C) signals (unless `jagacy.signals=false`), and will close the session, and logoff the application, if either occurs.

- Do not run a window in a production environment (set `<SessionName>.window=false`). This may unnecessarily slow down the application.