JTcl and Swank

What’s new with Tcl and Tk on the JVM

Bruce A. Johnson, Tom Poindexter, & Dan Bodoh
JTcl and Swank

Bruce’s Motivation

Cross-platform, scriptable, desktop applications for analyzing and visualizing large (multi-Gbyte) data sets

Complex applications for small market requires high productivity
Why Java

- Extensive library
- Threading
- Write Once, Run on Mac, Windows, Linux...
- Stupid mistakes hard(er) to make
- More productivity, for the same reasons Tcl more productive tool than C.
Why Java

- configure!!!!!

- Tcl 8.5.8 configure is 20162 lines long

- Not a problem for pure Tcl, but as soon as you start using C coded extensions...

- 32 bit/64 bit, Linux (which Linux), Mac, Windows (oops, not configure on Windows)
Why Java

% set ds [java::new org.apache.commons.math.stat.descriptive.DescriptiveStatistics]
java0x259
% $ds addValue 9
% $ds addValue 10
% $ds addValue 11
% $ds addValue 8
% set mean [$ds getMean]
9.5
% set variance [$ds getVariance]
1.66666666666667
Jacl

- Java Command Language
- Implementation of Tcl that runs on the Java Virtual Machine
- Project started by Ioi Lam and Brian Smith at Cornell (Tcl/Tk Conference 1997)
- Continued at Sun, then Scriptics, then largely maintained by Mo DeJong
Jacl

- Part of the Tcl/Java Project
- [http://tcljava.sourceforge.net](http://tcljava.sourceforge.net)
- Includes TclBlend which allows one to load the JVM into the C version of Tcl
- Most recent release is 1.4.1 in April 2008
- Essentially Tcl 8.0 w/o Tcl Byte Code Compiler
- Includes TJC - Tcl to Java Compiler & ITcl
Jacl → JTcl

- Started with the Google Summer of Code-2009
- Goal to bring Jacl to Tcl 8.4 (Radoslaw Szulgo)
- After GSC we decided to fork Jacl to JTcl to accelerate the incorporation of the changes and do further modernization
JTcl

- Drop TclBlend
- No interest to team, makes build environment more complex
JTcl Modernization

- Testing
  - Drive from JUnit
  - Drop platform specific tests
  - Fix erroneous test results
J TCL Code Modernization

• Rearrange packages
• Reformat code to “standard” Java style
• Switch from make to Maven
• IDE/mvn friendly layout
• Move from 4 jars (jacl,tcljava,ltcl, tjc) to 1 jar file (and it includes much of tcllib).
  • java -jar jtcl.jar
  • jtcl and jtcl.bat
JTcl Code Modernization

- Regular Expressions
  - java.lang.regex.Pattern
  - java.lang.regex.Matcher
- plus translations from Tcl to Java syntax
JTcl Pipelines

• Pipelines for “exec” and “open” commands now supported


• Redirection handled by JTcl, some limitations that may go away with Java 7
<table>
<thead>
<tr>
<th>Java Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeekableChannel</td>
<td>Abstract class that adds seek() and tell()</td>
</tr>
<tr>
<td>FileChannel</td>
<td>Extends <strong>SeekableChannel</strong> to implement file I/O</td>
</tr>
<tr>
<td>ResourceChannel</td>
<td>Implements reading of a Java resource using a “resource:” prefix on the file name</td>
</tr>
<tr>
<td>ReadInputStreamChannel</td>
<td>Bridges a Tcl channel to a Java <strong>InputStream</strong></td>
</tr>
<tr>
<td>AbstractSocketChannel</td>
<td>Abstract class that has common code for socket channels</td>
</tr>
<tr>
<td>ServerSocketChannel</td>
<td>Implements Tcl server sockets</td>
</tr>
<tr>
<td>SocketChannel</td>
<td>Implements Tcl sockets</td>
</tr>
<tr>
<td>TclByteArrayChannel</td>
<td>Used internally to bridge Tcl channels to Tcl byte arrays</td>
</tr>
</tbody>
</table>
JTcl fcop y & File Events

- fcop y
  - Copies bytes from one channel to another
  - Copy occurs in second Java thread
- file vents
  - Depends on new non-blocking IO in channel system
JTcl File Events

Hallmark of Tcl is the event system that allows writing servers with a minimum of code

Simple test of File Events and Channels
Dustmote (http://wiki.tcl.tk/4333)
Web server in ~40 lines of code

It works, and spawns threads proportional to the number of simultaneous client requests.
JTcl Summary

• ~Tcl 8.4
• + dict and apply from Tcl 8.5 (thanks to Neil Madden)
Swank

- Success of Tcl, has a lot to do with Tk
- Swank written to provide “Tk” companion to Jacl (now JTcl) and necessary for my applications.
- Swank generated as a combination of Java code that is hand written and generated with JTcl scripts.
## Swing - Tk Correspondence - Standard Widgets

<table>
<thead>
<tr>
<th>Swing</th>
<th>Tk</th>
<th>Swing</th>
<th>Tk</th>
</tr>
</thead>
<tbody>
<tr>
<td>JButton</td>
<td>button</td>
<td>JRadioButtonMenuItem</td>
<td>radiobutton (on menus)</td>
</tr>
<tr>
<td>JCheckBox</td>
<td>checkbutton</td>
<td>JScrollBar</td>
<td>scrollbar</td>
</tr>
<tr>
<td>JFrame</td>
<td>toplevel</td>
<td>JSlider</td>
<td>scale</td>
</tr>
<tr>
<td>JLabel</td>
<td>label</td>
<td>JSpinner</td>
<td>spinbox</td>
</tr>
<tr>
<td>JList</td>
<td>listbox</td>
<td>JTextArea</td>
<td>message</td>
</tr>
<tr>
<td>JMenu</td>
<td>menu</td>
<td>JTextField</td>
<td>entry</td>
</tr>
<tr>
<td>JMenuBar</td>
<td>menubar</td>
<td>JTextPane</td>
<td>text</td>
</tr>
<tr>
<td>JPanel</td>
<td>frame</td>
<td>JFrame (composite)</td>
<td>labelframe</td>
</tr>
<tr>
<td>JRadioButton</td>
<td>radiobutton</td>
<td>JPanel (customized)</td>
<td>canvas</td>
</tr>
<tr>
<td>JTextArea</td>
<td>message</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JTextPane</td>
<td>text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swing</td>
<td>Tk</td>
<td>Swing</td>
<td>Tk</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------</td>
<td>-----------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>JDesktopPane</td>
<td>jdesktoppane</td>
<td>JProgressBar</td>
<td>jprogressbar</td>
</tr>
<tr>
<td>JComboBox</td>
<td>jcombobox</td>
<td>JScrollPane</td>
<td>jscrollpane</td>
</tr>
<tr>
<td>JDialog</td>
<td>jdialog</td>
<td>JSplitPane</td>
<td>panedwindow</td>
</tr>
<tr>
<td>JEditorPane</td>
<td>html</td>
<td>JTabbedPane</td>
<td>jtabbedpane</td>
</tr>
<tr>
<td>JInternalFrame</td>
<td>JInternalframe</td>
<td>JTable</td>
<td>jtable</td>
</tr>
<tr>
<td>JOptionPane</td>
<td>joptionpane</td>
<td>JToolBar</td>
<td>jtoolbar</td>
</tr>
<tr>
<td>JPasswordField</td>
<td>jpasswordfield</td>
<td>JTree</td>
<td>jtree</td>
</tr>
<tr>
<td>JPopupMenu</td>
<td>jpopupmenu</td>
<td>JWindow</td>
<td>jwindow</td>
</tr>
</tbody>
</table>
Swank Canvas

- Additional Canvas Items
  - htext: Displays basic HTML
  - connector: Line between two other objects
  - annotation: Arrow with text at other end
  - charts: Implemented with JFreeChart

- Affine Transforms
  - Transform configuration for each item
  - Custom items can generate their own
  - Canvas wide allows zooming whole canvas
Swank Canvas

• Item Handles
  • Intrinsic aspect of each item
  • handle subcommand

• Scene graph
  • Each item has a -node option
  • New canvas node item
  • Traditional canvas is scene graph with one root
  • raise/lower operate on nodes
Swank Canvas3D

• Rendering with Java 3D
• Create items like as on traditional canvas, but with x,y,z coordinates
• Sphere, Cylinders, Cones, Text
• I use custom 3D molecule item
Swank

What's Next - TkFX

• Swank depends on Java Swing
  • Swing is unlikely to see further development by Oracle
• JavaFX is the new focus of Oracle for Java GUI
  • New hardware accelerated graphics engine (Prism)
  • New windowing toolkit (Glass)
  • Media Engine for streaming content
  • Web component based on WebKit
  • 3D graphics
TkFX

toplevel button entry grid raise lower wm bind webview htmleditor
JTcl & Swank

• JTcl
  • http://jtcl.kenai.com

• Swank
  • http://swank.kenai.com

• Veery
  • http://veery.kenai.com
  • (Vector data inspired by Vlerq/Metakit in Java 0.0)

• Hyde
  • http://aejaks.sourceforge.net

• Critcl for Java (generate and compile Java code from JTcl)