Benefits Gained By Using Excelsior JET To Compile Trita

Case Study by Matt Jones, Trita Software

Background

Trita is a source code formatter for a wide range of languages including: C#, Java, Javascript, JSP, HTML, and PHP.

Trita's primary business goal is to easily integrate into users' IDEs and ideally partner with IDE vendors so that Trita is available to IDE end users the instant the IDE is installed.

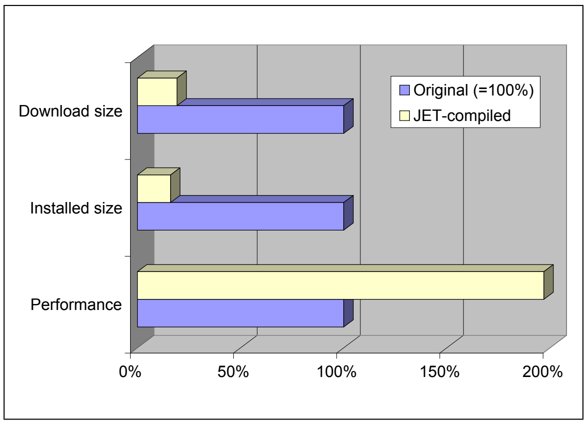
The original version of Trita came with a JRE, which brought the total distribution size to 7.8 MB. This was really too large for many IDE vendors to package it into their products, so there was enormous pressure to find a way to reduce Trita's size.

One solution was to rewrite Trita in C++, this would have taken at least six months. Happily, **Excelsior JET** provides a superior solution to shrink the distribution to a much more manageable size and without very much effort.

Here are the basic numbers extracted from my experience using JET:

Technical Benefits In A Nutshell

The original version included a 332 KB jar file and a 17.2 MB JRE which together zipped down to 6.3 MB.



The JET version eliminated the JRE and jar file by replacing them with a 2.54 MB exe and a 272 KB swt-win32-2130.dll. These zip down to 1.21 MB, resulting in:

- **Distribution size reduced 80.8%** from 6.30 MB to 1.21 MB.
- **Installed size reduced 83.9%** from 17.5 MB to 2.81 MB.

The JET version was also faster by 97%, or about double performance.

(Note these figures are for executable parts of Trita only, they do not include things which JET did not affect such as data files.)

Executive Benefits

1. Most importantly, JET makes Trita a much more saleable solution to IDE vendors by significantly reducing Trita's size.

2. Using JET reduced workload from spending 6 months porting and retesting 50k lines of Java to C++ down to about a week and a half using JET. This consisted of about one day getting familiar with JET and then setting up the JET compile scripts, followed by about four days rewriting Trita's AWT code into SWT, then a few days to tweak and test things to create the final JetPerfect version.

   (Note that this is very application dependent, if your application makes extensive use of the AWT, you will need to rewrite it all to SWT. This is not too much of a concern however, since the alternative is to rewrite it in C++ or Delphi which would take even longer. Also, I haven't found any bugs in SWT yet as opposed to needing three workarounds for bugs in the AWT.)

3. Downloads on the Trita web site were sometimes exceeding 1.6 gig per day so it was great to be able to cut web site bandwidth usage by 74%.

Other Niceties

I was happy to see how well JET's UI is put together. By UI I mean the whole experience including creating your setup scripts and then executing the scripts. JET comes with a slick tool for defining what goes into your exe, then gives the option to create the exe using the GUI, or creating a script that you can call from the command line (or Ant). This is a fantastic feature that makes cutting a new distribution a snap because you don't have to manually run the GUI.

Tips For Safe And Small Code Using JetPerfect

JetPerfect allows you to create a slimmed down and highly optimized polished executable, it is very powerful but requires some vigilance on your part:

1. Always test your exe using the JetPackII Trial Run feature to make sure it will run correctly on the client's machine.

2. If you're not using any AWT code, comment out all references to it. Don't rely solely on JetPerfect to remove the AWT code.

For the first production version of Trita I published I mistakenly did neither of the above. My app ran just fine on all the test machines I have as well as on most of my customer's machines. However it would fail to start on some (but not all) of the client systems. This was probably due to a DLL that was absent on some client systems.

When I tested my exe using JetPackII Trial Run I found that it did indeed not come up. I removed all references to the AWT in my code, then rebuilt it. The resulting exe ran correctly within JetPackII and was 4 MB smaller! The moral of the story is that JetPerfect is a very useful tool but it's best not to dump the entire task of size optimization on it, and that you should always test your app in JetPackII before releasing it.

Note that the usage file that JetPerfect generates contains only classes and methods that were accessed via Reflection at run time. If those classes in turn explicitly import other classes, the latter will not appear in the usage file. So even though your usage file does not list any AWT classes, your app may require AWT dlls.

Areas For Improvement

For credibility's sake I must mention the few areas where JET could be improved. JET is quite excellent so there are relatively few improvements that can be made to it

1. Cost. I can't help but to wonder why JET is positioned as a tool priced only for high-end users? JET is something every single Java developer can benefit enormously from, not just a select few niche users. In my view the supply/demand curve for JET should dictate a much much lower cost and much much higher sales volumes.

   For one thing, Tomcat could possibly be made to reasonably compete with Resin's JSP server if it were recompiled using JET. This would expand JET's customer base beyond just developers to include ISPs who want to provide a performant JSP solution and avoid Resin's $500/year subscription cost.

   JET should be made ubiquitous.

2. Only supports Windows. Support for Linux is on the way, but it would be good if JET supported all the platforms Java does. It would be even better if JET could be used as a cross compiler, so that you run one build script on one machine and create distribution files for all the platforms you need.

3. Generating the JetPerfect usage file can complicate the build process. JetPerfect is a very good thing, however creating the usage list for a SWT application requires a user be present during the build process to be there to click on various widgets in your app in order to make sure everything gets mentioned in the usage file. For instance, you have to manually click on scroller-controls because there is no way in SWT to programmatically simulate mouse scroll-wheel events. If you don't do this, and the user uses a scroll wheel to scroll a text box your app will crash because the code to handle that event is missing.

   Note that you are not required to use JetPerfect, and you only need to create a usage list if you require an extremely optimized executable.

Thoughts About SWT

In order to create a JRE-less Java application that has a GUI you will need to use SWT. SWT is an open source IBM/Eclipse project to replace AWT/Swing with something faster. If you don't mind including a JRE in your app's distribution, then you don't need to use SWT.

SWT is less buggy than Swing and has a far cleaner API than Swing, for one thing its layout management is better done than in AWT. SWT's `GridLayout` class is pretty simple to use and if you learn how it works you don't need to bother with the other layout manager classes.

However, SWT is a little less customizable than the AWT. For example there is no way to change the background colors of TabFolders and most of the classes are not meant to be subclassed, according to their javadocs. The terminology in SWT is a little different than in the AWT; this can make it confusing to find the class you need. For example, JTabbedPanes are called TabFolders and JFrames are called Shells. SWT also has less 3rd party tool support than Swing, so you will probably need to write all your SWT code by hand instead of using a Visual Studio style tool to construct your app's SWT GUI.

Overall, SWT is pretty good and any future Java GUIs I create will probably be done in SWT instead of Swing whether I need to use JET or not.

High Quality Customer Support

Every potential JET user should know that Excelsior provides excellent customer support, each time I have contacted their support engineers and sales representatives I have been exceptionally pleased with the quality and timeliness of their help.

Summary

JET is great, I highly recommend its use to everyone who wants to use Java but needs final exes that are smaller and faster than what Java alone can provide.

Excelsior, LLC
6 Lavrenteva Ave. Suite 441
Novosibirsk 630090 Russia
Tel: +7 (3832) 39 78 24
Fax: +1 (509) 271 5205
Email: java@excelsior-usa.com
Web: http://www.excelsior-usa.com

Matt Jones
100 Bayberry Drive
Springboro, OH 45066
Email: mljones@trita.com
Web: http://www.trita.com