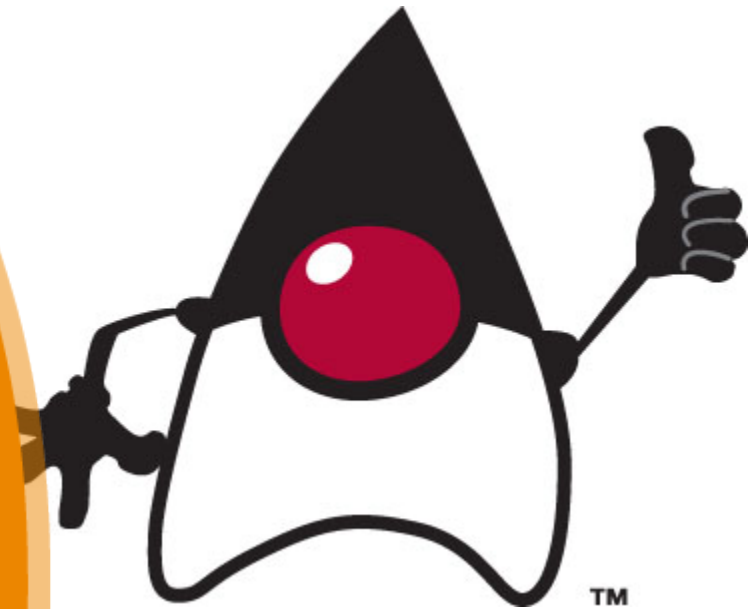


Java™ starting...

# Java Debugging

- L'uboš Koščo
- Solaris RPE Prague



# Agenda

- Debugging - the core of solving problems with your application
- Methodologies and useful processes, best practices
- Introduction to debugging tools
  - > bundled tools
  - > debuggers in Solaris, Linux and Windows
  - > java core dump analysis
  - > other methods/approaches to debugging
- Intro of available monitoring tools
- Sample debugging session of an average problem
- Backporting of tools, or what if I am locked into old JVM?
- Community, blogs, links, documentation, resources

# Debugging

- Monitoring of my application
- A problem occurs (crash, performance)
- Code overview needed for locating and narrowing the problem (opengrok, lxr, ... )
- Analyze & get to the root cause
- Reiterate with the fix

## since JDK 5.0

- Extensive monitoring and management support
  - > JMX (JSR-3), JMX Remote API (JSR-160),
- `java.lang.management` (JSR-163/174).
  - > “Out of the box” management capabilities
  - > `jconsole`
  - > JVM Tool Interface (JSR-163) to support a wide range
- development and production time tools.
  - > Also `java.lang.instrument`
- Fatal error handling mechanism
- `jstat` utility to collect and report performance stats

# JDK 6.0 – even more improvements

- Improvements to java.lang.OutOfMemoryError handling
- Heap dump when OOME thrown (\*)
- Stack trace and improved messages
- Fatal error log if VM aborts because out of native heap
- jhat to analyze heap dump
- Improvements to command-line utilities
- extend to live process inspection (all platforms)
- class-wise histogram of objects in heap
  - > jmap -histo <pid>
- capture heap dump
  - > jmap -dump:file=<file>,live <pid>
- change “manageable” options/flags dynamically

# JDK 6.0 ...

- jconsole improvements
- start management agent in local VM
- plug-in support
- UI improvements, ...
- HotSpotDiagnosticMXBean
- Built-in DTrace probes (Solaris 10)
- java.lang.management updated to expose
- java.util.concurrent lock information (also jstack -l)
- Improvements to the JVM Tool Interface
- Improved troubleshooting documentation

<http://java.sun.com/javase/6/webnotes/trouble/index.html>

# DTrace

- need to load agent in 5.0 (1.4.2) , transparent in 6.0
- **hotspot** Provider
  - > VM Lifecycle Probes
  - > Thread Lifecycle Probes
  - > Classloading Probes
  - > Garbage Collection Probes
  - > Method Compilation Probes
  - > Monitor Probes
  - > Application Tracking Probes
- **hotspot\_jni** Provider
- some methods DO introduce overhead

<http://java.sun.com/javase/6/docs/technotes/guides/vm/dtrace.html>

# Bundled tools

- Java Monitoring and Management Console (**jconsole**)
- Experimental JDK Tools and Utilities

NOTE - The tools described in this section are unsupported and experimental in nature and should be used with that in mind. They might not be available in future JDK versions.

- > Monitoring Tools (**jps, jstat, jstatd**)
- > Troubleshooting Tools (**jinfo, jhat, jmap, jsadebugd, jstack**)
- > Scripting Tools (**jrunscript**)
- jhat needs lots of memory (e.g. -J-mx512m )
- jhat uses OQL (object query language)
- Console enabled in ControlPanel

<http://java.sun.com/javase/6/docs/technotes/tools/>

# 3d party profiling SW

- Netbeans profiler
- JProfiler
- Cougaar Memory Profiler
- JBoss Profiler
- JBuilder profiler (Optimizelt)
- Jprof
- Eclipse Test & Performance Tools Platform

<http://java-source.net/open-source/profilers>

# Debuggers

- built in **jdb** , JPDA knowledge necessary (you can attach remotely! -g compiled code is useful!)
- **dbx** (part of SunStudio)
- Netbeans visually integrates jdb / JPDA
- JSwat
- Omniscient Debugger
- gcore, truss, ...

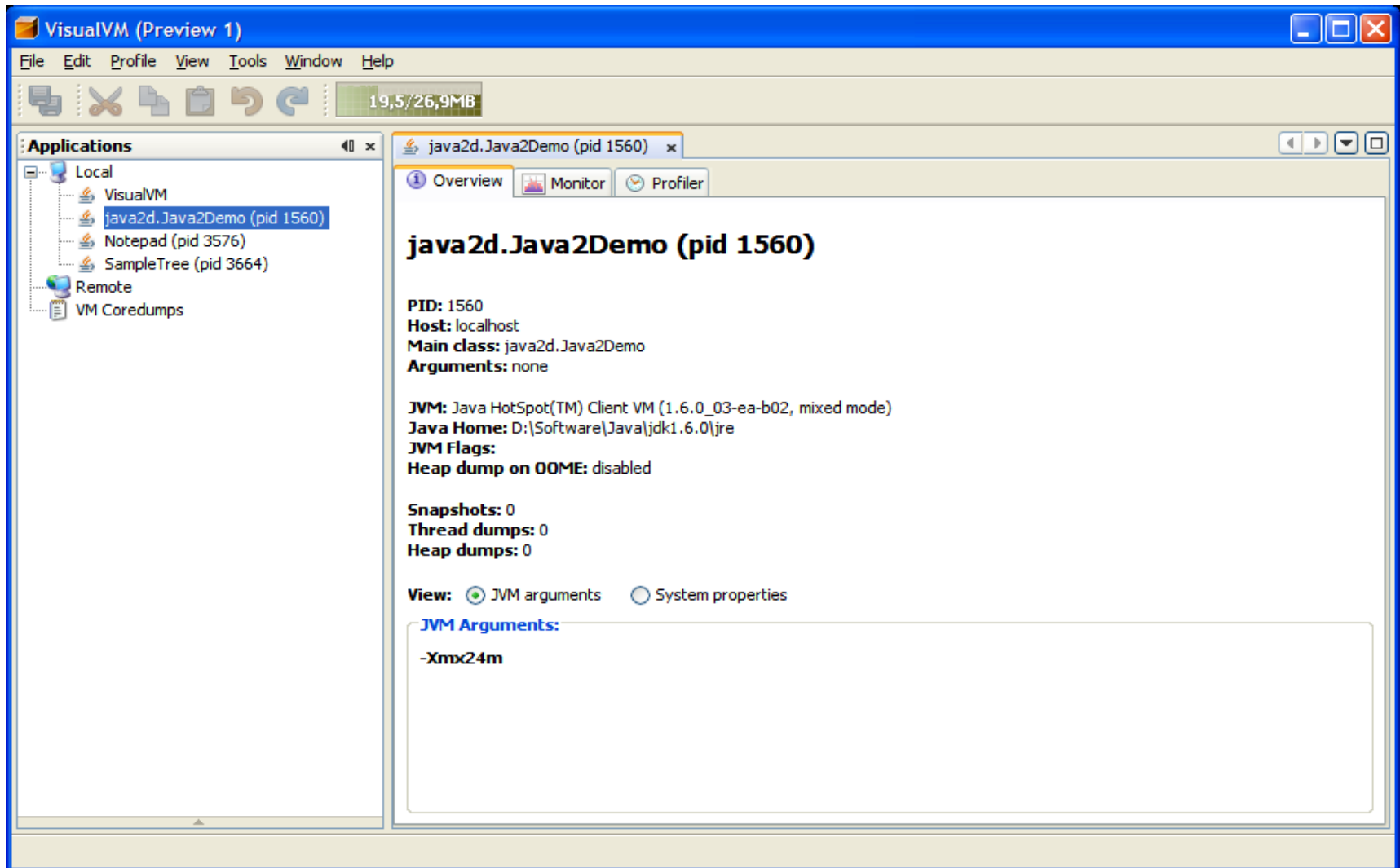
# Old JVM tools

- backporting of new tools doesn't happen
- some options in latest updates ( `-XX:+HeapDumpOnOutOfMemoryError` ), outputs can be parsed by latest tools
- **key** of how to get info is to look up documentation in latest builds and relevant bug documentation
- kill `-QUIT` gives stack trace

DEMO

# VisualVM – JavaOne 2007 proceedings

- <https://visualvm.dev.java.net/>



# Links

- look for JDK troubleshooting(TS) guides  
(<http://java.sun.com/javase/6/webnotes/trouble/TSG-VM/html/index.html> )
- Blogs!
  - > <http://weblogs.java.net/blog/mandychung>
  - > <http://blogs.sun.com/sundararajan>
  - > <http://blogs.sun.com/alanb>
  - > <http://jroller.com/nbprofiler/>
- Dtrace:  
[http://www.sun.com/bigadmin/features/articles/java\\_se6\\_observability.html](http://www.sun.com/bigadmin/features/articles/java_se6_observability.html)

# Even MORE links !

- [http://blogs.sun.com/scblog/entry/swing\\_based\\_jhat\\_oql\\_execution](http://blogs.sun.com/scblog/entry/swing_based_jhat_oql_execution)
- [http://blogs.sun.com/alanb/entry/heap\\_dumps\\_are\\_back\\_with](http://blogs.sun.com/alanb/entry/heap_dumps_are_back_with)
- <http://java.sun.com/developer/technicalArticles/J2SE/monitoring/>
- [http://weblogs.java.net/blog/kellyohair/archive/2005/09/heap\\_dump\\_snaps.html](http://weblogs.java.net/blog/kellyohair/archive/2005/09/heap_dump_snaps.html)
- <http://forum.java.sun.com/thread.jspa?threadID=735052&tstart=0>
- [http://developers.sun.com/sunstudio/articles/Java\\_debug/Java\\_debug\\_content.html](http://developers.sun.com/sunstudio/articles/Java_debug/Java_debug_content.html)
- <http://forum.java.sun.com/thread.jspa?threadID=735052&tstart=0>
- <http://forum.java.sun.com/thread.jspa?threadID=765050&tstart=0>
- <http://weblogs.java.net/blog/emcmanus/>
- <http://blogs.sun.com/kamg>

Thank YOU!

# Java Debugging

•Luboš Koščo

•Solaris RPE Prague

[Lubos.Kosco@Sun.COM](mailto:Lubos.Kosco@Sun.COM)

## Object Query Language (OQL)

OQL is SQL-like query language to query Java heap. OQL allows to filter objects. Pre-defined queries such as "show all instances of class X" are already supported. OQL is based on JavaScript expression language.

OQL query is of the form

```
select <JavaScript expression to select>  
[ from [instanceof] <class name> <identifier>  
[ where <JavaScript boolean expression to filter>
```

<class name> is fully qualified Java class name (example: java.net.URLEncoder; is name of java.io.File[] and so on. Note that fully qualified name is used at runtime. There may be more than one Java class with the same name. <identifier> to be id string of the class object. If **instanceof** keyword is used, only the instances of exact class specified are selected. Both **from** and (optional) **where** clauses, the expression used in JavaScript to filter objects so that fields may be accessed in natural syntax. Array elements can be accessed with array[index] syntax. Variable of the identifier name specified in **from** clause.

### Examples



# Troubles with troubleshooting

- `jps` – problem with `hsperfdata_<user>` in `%TMP%`
- `bash-3.2$ ./jinfo -flag HeapDumpOnOutOfMemoryError 3188`  
`-XX:-HeapDumpOnOutOfMemoryError`