JProbe® Suite

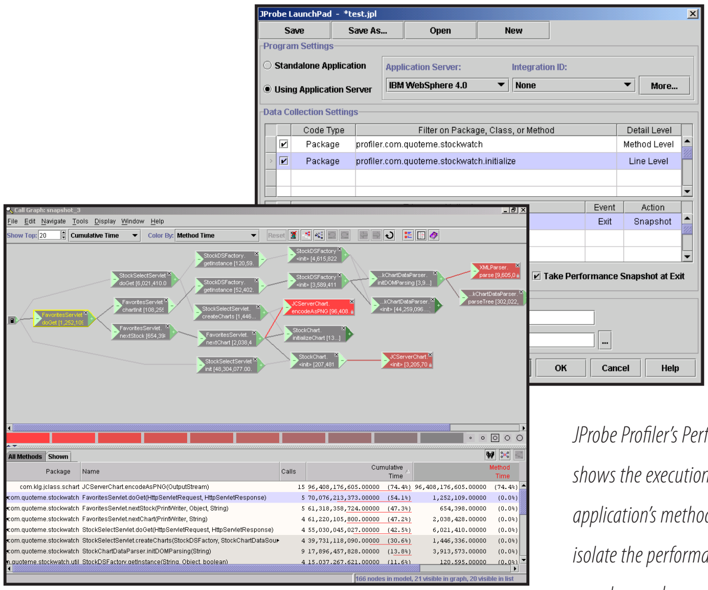
Comprehensive Java Performance Tuning

While J2EE and J2SE applications are relatively easy and fast to build, diagnosing and fixing performance and stability problems is still complex. Development and production support teams spend countless hours responding to and isolating the root cause of performance and memory issues. Meanwhile, deadlines are missed, budgets are spent and end-user experience is poor.

JProbe® Suite, three-time winner of the Software Development Magazine Jolt Productivity Award, enables Java developers and QA/test teams to diagnose code-level performance, memory and code coverage issues quickly, so they can find and fix problems faster. JProbe helps you accelerate development, contain costs and ensure a faster, more scalable Java application.

JProbe Suite enables you to:

- Find and fix performance bottlenecks quickly and easily
- Identify the lines of code that have the greatest effect on performance
- Eliminate bugs and sluggish performance earlier in the cycle to reduce ongoing hardware and development costs
- Release applications with confidence that they have been tested thoroughly
- Achieve higher levels of performance and end user satisfaction
- Automate the collection of performance information to occur at off-peak hours


JProbe LaunchPad - *test.jpl
Save
Save As...
Open
New
Program Settings
More...
Data Collection Settings
OK
Cancel
Help
Call Graph: snapshot_3
File
Edit
Navigate
Tools
Display
Window
Help
Reset
StockSelectServlet doGet [6,021,410.0
FavoritesServlet chartInit [108,25
FavoritesServlet doGet [1,252,10
FavoritesServlet. nextStock [654,39
StockDSFactory. getInstance [120,59.
StockDSFactory. getInstance [52,402.
StockSelectServlet. createCharts [1,446...
FavoritesServlet. nextChart [2,038,4
StockSelectServlet init [48,304,077.00.
StockDSFactory <init> [4,615,822
StockDSFactory <init> [3,589,411
JCServerChart. encodeAsPNG [96,408.
StockChart. initializeChart [13...]
StockChart. <init> [207,481
...kChartDataParser. initDOMParsing [3,9...]
...kChartDataParser. <init> [44,259,096...]
JCServerChart. <init> [3,205,70
XMLParser. parse [9,605,0
...kChartDataParser. parseTree [302,022,
All Methods
Shown

○ Standalone Application
● Using Application Server

Application Server: IBM WebSphere 4.0
Integration ID: None

	Code Type	Filter on Package, Class, or Method	Detail Level
☑	Package	profiler.com.quoteme.stockwatch	Method Level
☑	Package	profiler.com.quoteme.stockwatch.initialize	Line Level

Event	Action
Exit	Snapshot

☑ Take Performance Snapshot at Exit

Show Top: 20 Cumulative Time
Color By: Method Time

Package	Name	Calls	Cumulative Time		Method Time	
com.klg.jclass.schart	JCServerChart.encodeAsPNG(OutputStream)	15	96,408,176,605.00000	(74.4%)	96,408,176,605.00000	(74.4%)
com.quoteme.stockwatch	FavoritesServlet.doGet(HttpServletRequest, HttpServletResponse)	5	70,076,213,373.00000	(54.1%)	1,252,109.00000	(0.0%)
com.quoteme.stockwatch	FavoritesServlet.nextStock(PrintWriter, Object, String)	5	61,318,358,724.00000	(47.3%)	654,398.00000	(0.0%)
com.quoteme.stockwatch	FavoritesServlet.nextChart(PrintWriter, String)	4	61,220,105,800.00000	(47.2%)	2,038,428.00000	(0.0%)
com.quoteme.stockwatch	StockSelectServlet.doGet(HttpServletRequest, HttpServletResponse)	4	55,030,045,027.00000	(42.5%)	6,021,410.00000	(0.0%)
com.quoteme.stockwatch	StockSelectServlet.createCharts(StockDSFactory, StockChartDataSour	4	39,731,118,090.00000	(30.6%)	1,446,336.00000	(0.0%)
com.quoteme.stockwatch	StockChartDataParser.initDOMParsing(String)	9	17,896,457,828.00000	(13.8%)	3,913,573.00000	(0.0%)
n.quoteme.stockwatch.util	StockDSFactory.getInstance(String, Object, boolean)	4	15,037,267,621.00000	(11.6%)	120,595.00000	(0.0%)

166 nodes in model, 21 visible in graph, 20 visible in list

JProbe Profiler's Performance Call Graph shows the execution path of your application's methods. This helps you isolate the performance bottlenecks in your Java code.

> *"To go from a manual, hand-instrumenting process to a tool like JProbe resulted in huge productivity gains, not just for me personally, but for the group as a whole. JProbe should be made available to all developers."*
>
> —Ed Rybak,
> Developer,
> IAS Performance Group, Oracle

- Deploys across any combination of platforms, offering the broadest O/S and application server support
- Pinpoints specific lines that have the greatest impact on performance and memory
- Complete performance management solutions for J2EE and J2SE applications
- Provides unique investigative tools including memory leak calculator, automated data collection, line-level analysis, snapshot differencing

QUEST SOFTWARE®

System Requirements

Application Server Integration:

- BEA WebLogic Server
- IBM WebSphere Application Server
- IBM WebSphere Portal Server
- Sun Java System Application Server
- Apache Tomcat
- Apache Geronimo
- Oracle9i Application Server
- JBoss

Integrated Development Environment (IDE) Support:

- IBM WebSphere Studio Application Developer (WSAD)
- Eclipse
- Borland JBuilder
- IntelliJ IDEA
- Sun Java Studio
- Oracle JDeveloper
- NetBeans

Operating System Support:

- Windows 2000/XP/2003
- Solaris SPARC
- AIX 5L
- Linux
- HP-UX 11i
- Linux on zSeries
- Linux on 64-bit POWER

JDK Support:

- JDK 1.3.x, 1.4.x, and 1.5.0

Note: not all JDKs are supported on all platforms.

QUEST SOFTWARE®

www.quest.com
e-mail: info@quest.com
Please refer to our Web site for
international office information.

JProbe Profiler combines a visual Call Graph interface and sophisticated data collection technology to provide precise performance diagnostics. Using method and line-level analysis, you can locate method hotspots and drill down to measure performance, line-by-line. Profiler measures elapsed and CPU time to help you track the end user experience and locate computational bottlenecks. Advanced filtering and triggers enable you to target key areas of code for investigation. As you use Profiler, you will generate "snapshots" of actual performance. When testing different fixes, use Snapshot Differencing to see the impact of your code changes on performance. Profiler allows broad and custom reporting capabilities for printing and exporting. You can export to PDF, TXT, HTML or CSV format.

JProbe Memory Debugger quickly pinpoints memory leaks and object cycling in Java code with real-time views of memory and object use. You can track memory growth at runtime with easy-to-use, two-button analysis. The Memory Instance Calculator calculates the size of memory leaks, and the Leak Doctor pinpoints possible sources of memory leaks. You can also trace memory use and object references, perform garbage collection analysis and, as with Profiler, use Snapshot Differencing to reveal the impact of code changes on memory use.

JProbe Coverage locates unexecuted code and precisely measures statements that have been exercised, making it easier to assess the reliability and accuracy of test runs. The built-in Coverage Browser and Source views quickly isolate untested or dead code. Conditional Coverage Analysis measures the coverage of conditional code paths. You can filter Catch Blocks for greater accuracy in generated coverage reports. Coverage can run in batch mode to integrate with nightly test runs. Coverage data can be exported to XML, TXT, CSV or HTML formats. Snapshot Merging enables a consolidated view of multiple program runs.

About Quest

Quest Software, Inc. delivers innovative products that help organizations get more performance and productivity from their applications, databases and Windows infrastructure. Through a deep expertise in IT operations and a continued focus on what works best, Quest helps more than 18,000 customers worldwide meet higher expectations for enterprise IT. Quest Software can be found in offices around the globe and at **www.quest.com**.

IBM Business Partner

bea 8.1 VALIDATED BEA WEBLOGIC WORKSHOP

bea VALIDATED 9 BEA WEBLOGIC SERVER

software development 15th annual productivity award