Profiling Java applications using Eclipse* Test and Performance Tools Platform
Facilitators

Yunan He
Intel Corporation
yunan.he@intel.com

Chris Elford
Intel Corporation
chris.l.elford@intel.com

Eugene Chan
IBM Corporation
ewchan@ca.ibm.com
Agenda

- Unit 1 – Eclipse* TPTP Overview
  - Please work with facilitators during this presentation to get TPTP and samples installed during this presentation

- Unit 2 – Using the TPTP Java Profiler
  - Profiler Architecture
  - Profiling and Logging Perspective
  - Launch and Attach
  - Profiling options and views
  - Demo
Participants

Tell me a little about yourselves…

- From where
- Company or academic institution
- Users or Adopters
- Eclipse* and TPTP experience
- Java* experience
- Have you used a Java* profiler before?
- Tutorial expectations
Unit 1 – Eclipse* TPTP Overview
  Please work with facilitators during this presentation to get TPTP and samples installed during this presentation

Unit 2 – Using the TPTP Java Profiler
  Profiler Architecture
  Profiling and Logging Perspective
  Launch and Attach
  Profiling options and views
  Demo
Eclipse* TPTP Overview

- Eclipse* Tools Project, Hyades, Dec 2002
  - Promoted to Eclipse* project, 2004

- Open-source platform for Automated Software Quality (ASQ) tools including reference implementations for testing, tracing and monitoring software systems

- Addresses the entire test and performance life cycle, from early testing to production application monitoring, including test recording, editing and execution, monitoring, tracing and profiling capabilities

- Integration with tools used in the other processes of a software lifecycle under Eclipse* environment

- Reduce the cost and complexity of implementing effective automated software quality control processes

- Share data through an OMG-defined trace, log, statistical and test model implemented via the Eclipse* Modeling Framework (EMF)

- Active participants:
  - IBM*, Intel®, OC Systems*

- Former participants:
  - Compuware, SAP, Scapa Technologies
More about Eclipse* TPTP Overview

➢ Composed of 4 sub projects:
  ▪ Platform
  ▪ Trace and Profiling
  ▪ Test
  ▪ Monitoring

➢ Principles:
  ▪ Extension of the Eclipse* Value Proposition
  ▪ Vendor Ecosystem
  ▪ Vendor Neutrality
  ▪ Standards-Based Innovation
  ▪ Agile Development
  ▪ Inclusiveness & Diversity

➢ TPTP is highly extensible.

TPTP Profiler developed/maintained by these teams
Eclipse* TPTP Profiling Tool Overview

- Subject of the day

- Target use cases
  - For performance analysis and deeper understanding of Java* programs
    - Visualization of program execution and threading behavior
    - Pinpointing operations that take most resources
    - Exploring patterns of program behavior
  - For early-in-the-cycle tests of your application

- Easy to use with extensive GUI’s
  - Profiling and Logging Perspectives
  - A number of graphical and tabular views

- Low Overhead
  - Enables on-demand profiling by running applications with agent-on at near full speed and later attaching to gather data in various phases

- Advanced data processing
  - Assorted filtering functionalities to help localize problems and reduce data volume for long running applications
  - Sophisticated input stream analyzer
Eclipse* TPTP Agent Controller & Java Profiler

- A daemon process that resides on each system under test (SUT) and the TPTP workbench
  - Enables workbench to launch processes on SUT (or locally)
  - Interacts with other agents that coexist on the SUT (or locally)
  - Capable to manage local or remote applications from a local TPTP workbench
  - Option to use integrated agent controller when the workbench and the SUT are the same system
- The Java* profiler is a managed agent that can be used to profile local or remote Java* applications from a local TPTP workbench
Data Collection Workflow

TPTP Workbench
- Workbench Process
- Agent Controller Interface
- TPTP Presentation

Agent Process
- Agent Controller
- Data

Application Process
- Java Profilers
- Profiling Framework

System Under Test (SUT)
- Application
- Control
- Data

Control
- Data
Eclipse* TPTP Project

TPTP Architecture

Presentation System
- Eclipse* TPTP GUI
  - Runtime Monitor/log
  - Trace Analysis
  - Test Creation and Execution
  - Artifact Management
- Standard Widgets and Core Plug-ins
- Reference Perspectives and Workflow
- EMF Data Models
  - Test
  - Trace
  - Log
  - Statistical
- Real Time Export
- XMI
- Data Loader

Eclipse* Platform

Target System
- Test Engine
  - Testability Interface
- Application
  - Trace Collection
  - Log Collection
- Execution Environment
  - JVMPI Monitor
  - JVMTI Monitor
  - System Performance Monitor

Distributed Control Framework

Distributed Data Collection Framework

Artifact Control Interface

Data Collection Interface
Unit 1 – Eclipse* TPTP Overview

- Please work with facilitators during this presentation to get TPTP and samples installed during this presentation

Unit 2 – Using the TPTP Java* Profiler

- Profiler Architecture
- Profiling and Logging Perspective
- Launch and Attach
- Profiling options and views
- Demo
The Java Profiler

- Set of libraries to attach to JVM & record Java App behavior
- Identify performance details (e.g.,): classes or methods responsible for execution bottlenecks; analyze application heap to find memory leaks; visualize threading behavior.
- Applications under test can reside in Eclipse workspace, binaries on file system, or hosted in a J2EE application server.
- Can be launched from the Eclipse IDE or as a standalone program using Java command-line options
- Output in the form of XML fragments (XML4Profiling)
- Extensible framework: core runtime component (Martini); agent managed by the Agent Controller (JPIAgent); set set of data collection libraries built on top of the Martini runtime.
JVMTI-based Java* Profiler Architecture

Application Under Test (Java* Virtual Machine)

Java* Virtual Machine Tools Interface (JVMTI)

Martini Framework for Java*

Agent Control and Data Collection module (JPIAgent)

Martini-based Data Collectors

- Call Graph (CGProf)
- Heap (HeapProf)
- Thread (ThreadProf)

Distributed Control and Data Collection Framework

Presentation System (Eclipse* Workbench)
Profiling and Logging Perspective

- Profiling and Logging perspective provides resources for starting a profiling session as well as obtaining comprehensive information on the performance of the monitored application.

- The profiling tool provides information pertaining to:
  - Execution analysis
  - Object allocations
  - Thread interactions
Profiling and Logging Perspective

- Combinations of views and editors that are best suited to perform application profiling

- Profiling Monitor view
  - Administer profiling resources and manage activities

- Profiling views
  - Visualize and analyze profiling data

- Profiling actions
  - Control profiling resources
  - Actions are resource sensitive
  - Choice of action depending on type and status of the object in selection
    - Attach and Detach the agent from process
    - Start and Stop monitoring on an agent
    - Terminate a process
Monitor actions
Open view actions
View level actions
Monitor and Navigator view
Profiling views
Host: system under profile

Process: application under profile

Profiling Agent & Analysis types
Launch and Attach

- Profiling session is started by launching an application or by attaching to a running application
  - Launch: start an application with a profiling agent
  - Attach: attach to an application which is already started and invoked with a profiling agent
- How? Launch configuration is where you start the session.
Launch and Attach - Configuration

- A place to create, run and manage profiling sessions.

- Configure the details of a profiling session
  - Target host
  - Application to profile
  - Scope of the profile
  - Profile data destination

- Select launch configuration type according to the location of the target Java* application:
  - Within Workbench > “Java Application” launch configuration
  - Outside Workbench > “External Java Application” launch configuration
Launch and Attach – Configuration Tabs

- Collect and display detailed information about a configuration

Tabs for profiling:
- Monitor tab: defines details of data collection on a profiling session
- Destination tab: defines the destination of the profiling data
- Host tab: for external application configurations only, defines the location of the process to be launched or attached
- Agents tab: for attach configurations only, lists agents available for attaching
Launch and Attach – Monitor Tab

Data Collectors

Analysis Types
Launch and Attach – Profiling Options

![Profiling Options Interface]

- **Name:** InOut
- **Data Collectors:**
  - Agent Discoverer
  - Java Profiling - JRE 1.5 or newer (double click to modify filters)
  - Execution Time Analysis
  - Memory Analysis
  - Thread Analysis
  - Probe Insertion
  - Instrument Collector - pre JRE 1.6

- **Options:**
  - Edit Options
  - Test Availability

Press F2 to see a quick description of the selected data collector or analysis type.
Profiling Options – Filter Set

- Limit the scope of a profiling session
- Ensure that only relevant details are collected
- Especially useful when speed and efficiency is critical
  - Excluded classes and methods will not be instrumented and will execute at full speed
- Only the first applicable filter is applied. When you are specifying filters, ensure that you declare the most specific filter criteria first
Launch and Attach – Options
### Execution Time Analysis Options

- Specifies the type of information to collect during execution time analysis

- Use the “Show execution statistics” option to identify the most time-consuming methods
  - Low-overhead. Can be used without extensive filtering

- Use the “Show execution flow graphical details” option to identify the relationships between executing methods (call-graph).
  - High-overhead. Filtering is recommended.
Memory Analysis Options

- Specifies the type of information to collect during memory analysis

- Use the “Track object allocation sites” option to identify the source line where each object is allocated
  - May slightly increase analysis overhead
Thread Analysis Options

- Specifies the type of information to collect during thread analysis

- Use the “Contention analysis” option to analyze thread contention
  - May slightly increase analysis overhead
Launch and Attach – Destination Tab

- Specify the destination of the profile data
  - Workbench: Visualize and analyze in profiling views (default)
  - File: export to XML file or Bin file (default), available for import.

- Import profiling file
  - File > Import ... > Profiling file
Profiling an External Application – Host Tab

![Host Tab](image)

- **Default Hosts:**
  - localhost:10002
  - localhost:10006
  - lab.testhost:10002

- **Host name or IP address:** lab.testhost
- **Agent Controller port:** 10002

- **Buttons:**
  - **Edit...**
  - **Add...**
  - **Remove**
  - **Test Connection**
Attaching to a Running Application – Monitor Tab

Select the data collectors and analysis types for the launch. Press F2 to see a quick description of the selected data collector or analysis type.

- Agent Discoverer
- Java Profiling - JRE 1.5 or newer (double click to modify filters)
- Execution Time Analysis
- Memory Analysis
- Thread Analysis
- Probe Insertion
- Instrument Collector
Profiling Views

- A collection of views to visualize and analyze profiling data
  - Memory Statistic view (tabular)
  - Execution Statistic view (tabular)
  - Coverage Statistic view (tabular)
  - Method Invocation Detail view (tabular)
  - Execution Flow view and table (graphical + tabular)
  - Method Invocation view and table (graphical + tabular)
  - Execution Call Graph (graphical + tabular)
  - UML2 Trace Interactions view (graphical)
Recommendations for effective profiling

- Use filters to collect only necessary data
- Pause/resume to select activities to profile
- Attach/Detach to eliminate overhead when you don’t want to profile
Access to Attach/Detach and Pause/Resume
Java* Profiling - Attach

- Start application under profile in standalone mode
- Invoke the Java* Profiling Agent, a library that attaches to a Java* virtual machine (JVM)
  - Use the -agentlib JVM option to invoke the Java* Profiling Agent (see blow)
    - -agentlib:JPIBootLoader=JPIAgent:server=enabled;{profiler}
- Communication with the invoked agent is done from the client workbench
- Attach to the agent: Profile Configuration... > Attach to Agent
Java* Profiling - Demo

- Profiling a local Java* application
  - Execution Time Analysis
  - Memory Analysis
  - Thread Analysis
Demo: Execution Time Analysis

- Demo application: ProductSample
  - Load product data and display all information
  - Totally 24 products and data is stored in 24 XML files respectively.
- How to launch?
  - Profile as java application
  - Arguments tab: provide the products directory in “Program arguments”
  - Monitor tab: Choose execution time analysis with option “show execution statistics”
## Session summary

The table below shows the execution time analysis for the **createParser()** method of the **SAXParser** class. This method has the highest base time of 1.149800 seconds, with an average base time of 0.047908 seconds and cumulative time of 1.149800 seconds. It is called 24 times.

<table>
<thead>
<tr>
<th>Package</th>
<th>Base Time (seconds)</th>
<th>Average Base Time (seconds)</th>
<th>Cumulative Time (seconds)</th>
<th>Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.sample.product.util</td>
<td>1.330860</td>
<td>0.013443</td>
<td>1.330860</td>
<td>99</td>
</tr>
<tr>
<td>ProductCatalog</td>
<td>1.330860</td>
<td>0.013443</td>
<td>1.330860</td>
<td>99</td>
</tr>
<tr>
<td>createParser() Saxon</td>
<td>1.149800</td>
<td>0.047908</td>
<td>1.149800</td>
<td>24</td>
</tr>
<tr>
<td>parseContent(java.io.File)</td>
<td>0.179300</td>
<td>0.007479</td>
<td>0.350206</td>
<td>24</td>
</tr>
<tr>
<td>startElement(java.lang.String, java.lang.String)</td>
<td>0.000901</td>
<td>0.000019</td>
<td>0.000901</td>
<td>48</td>
</tr>
<tr>
<td>readData(java.lang.String)</td>
<td>0.000582</td>
<td>0.000582</td>
<td>1.330788</td>
<td>1</td>
</tr>
<tr>
<td>ProductCatalog()</td>
<td>0.000049</td>
<td>0.000049</td>
<td>0.000049</td>
<td>1</td>
</tr>
<tr>
<td>getContent() java.lang.String</td>
<td>0.000023</td>
<td>0.000023</td>
<td>0.000023</td>
<td>1</td>
</tr>
<tr>
<td>com.sample.product</td>
<td>0.036997</td>
<td>0.012332</td>
<td>1.367857</td>
<td>3</td>
</tr>
</tbody>
</table>
Demo: Memory Analysis

- Demo application: MemoryLeak
  - Three buttons: leak some memory, allocate some memory, run garbage collector
  - Leak some memory action: create new object and store in hash buffer. The newly allocated object will not be used again.
  - Allocate some memory action: create new object and it won’t be assigned to any variable.
  - Run garbage collector action: run GC.

![Memory Leak Simulator](image)
Demo: Memory Analysis (2)

- How to launch?
  - Profile as java application
  - Monitor tab: Choose memory analysis with option “track object allocation sites”

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Package</th>
<th>Live Instances</th>
<th>Active Size</th>
<th>Total Instances</th>
<th>Total Size</th>
<th>&lt;Avg. Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>int[]</td>
<td>(default package)</td>
<td>10</td>
<td>8680</td>
<td>11</td>
<td>8760</td>
<td>2.18</td>
</tr>
<tr>
<td>boolean[]</td>
<td>(default package)</td>
<td>20</td>
<td>480</td>
<td>20</td>
<td>480</td>
<td>2</td>
</tr>
<tr>
<td>MemoryLeak[]</td>
<td>(default package)</td>
<td>1</td>
<td>376</td>
<td>1</td>
<td>376</td>
<td>2</td>
</tr>
<tr>
<td>short[]</td>
<td>(default package)</td>
<td>5</td>
<td>1264</td>
<td>6</td>
<td>1280</td>
<td>1.5</td>
</tr>
<tr>
<td>long[]</td>
<td>(default package)</td>
<td>6</td>
<td>344</td>
<td>3</td>
<td>584</td>
<td>1.5</td>
</tr>
<tr>
<td>byte[]</td>
<td>(default package)</td>
<td>36</td>
<td>354960</td>
<td>331</td>
<td>909280</td>
<td>1.28</td>
</tr>
<tr>
<td>char[]</td>
<td>(default package)</td>
<td>20</td>
<td>5512</td>
<td>200</td>
<td>37336</td>
<td>1.05</td>
</tr>
<tr>
<td>float[]</td>
<td>(default package)</td>
<td>3</td>
<td>2744</td>
<td>3</td>
<td>2744</td>
<td>1</td>
</tr>
<tr>
<td>int[]</td>
<td>(default package)</td>
<td>1</td>
<td>880</td>
<td>1</td>
<td>880</td>
<td>1</td>
</tr>
<tr>
<td>MemoryLeak$NullItems</td>
<td>(default package)</td>
<td>10000</td>
<td>160000</td>
<td>10000</td>
<td>160000</td>
<td>0.61</td>
</tr>
</tbody>
</table>
Demo: Memory Analysis (3)

- How to identify memory leak
  - Click on “Run garbage collection” to force a garbage collection event
  - Monitor the age of object, if object age continues to grow then this object is a candidate.
  - Navigate to the allocation details view to analyze the allocation sites
Resources

- TPTP (Documentation, Download, CVS, Newsgroups, mailing list, etc.)
  - http://www.eclipse.org/tptp/
- TPTP Online Help

Eclipse TPTP Project

Test & Performance Tools Platform (TPTP) Project - TPTP provides powerful frameworks and services for an open platform upon which developers build unique test and performance tools. TPTP offers open source and commercial tools that integrate with Eclipse and other tools and address the active test and performance lifecycle. Help developer testing through productive monitoring.

Usage help:

- TPTP at EclipseCon 2008 Posted Feb 07, 2008 | Permalink
- View the TPTP sessions at EclipseCon 2008.
- TPTP Webinar on Eclipse Live! Posted Nov 14, 2007 | Permalink
- TPTP webinar over the Eclipse Live! Webinar.
- Then try our Eclipse Test and Performance Tools Platform (TPTP) for Eclipse Live! for TPTP plugin and testing tools. If you missed the webinar, you can see the recorded webinar.
- TPTP 4.4.0.3 has been released! Posted Sep 28, 2007 | Permalink
- The TPTP 4.4.0.3 release is now available. This release contains bug fixes not available in TPTP 4.4.0.2.
- TPTP 4.4.0.3 is now available! Posted May 09, 2007 | Permalink
- The TPTP 4.4.0.3 release is now available. This release contains bug fixes not available in TPTP 4.4.0.2.
- TPTP 4.4.0.3 has been released! Posted Mar 22, 2007 | Permalink
- The TPTP 4.4.0.3 release is now available. This release contains bug fixes not available in TPTP 4.4.0.2.

Capabilities

- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
- Show documentation about capabilities that are disabled in the application; select the Show/Hide button ( ). When you choose to show all topics in the table of contents, the headings for any disabled capabilities are shown in the table of contents and also appear in search results.
Q & A
Legal Notices:

- Other company, product, or service names may be trademarks or service marks of others.