

# Eclipse Test and Performance Tools Platform (TPTP) Unit 1 – Introductions

# Instructors

- Eugene Chan, IBM Rational Software  
Software developer, committer of TPTP (Hyades) project.  
ewchan@ca.ibm.com
- Guru Nagarajan, Intel Corporation  
Intel Software Products Division  
guru.nagarajan@intel.com

# Audience

Please introduce yourself

- Name
- Knowledge/Experience on Eclipse-TPTP Profiling Tool
- What do you hope to get from the workshop?

# Agenda

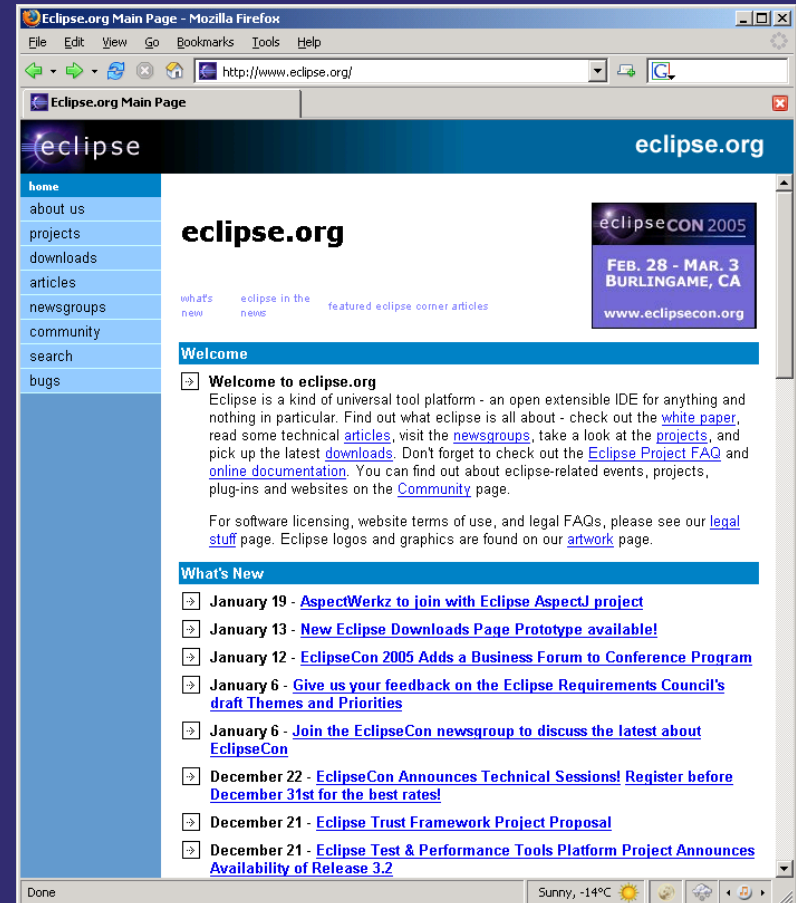
- Unit 1 - Introductions
- Unit 2 - Technology Overview
- Unit 3 - Profiling and Logging Perspective
- Unit 4 - Launch
- Unit 5 - Attach
- Unit 6 - Profiling options
- Unit 7 - Profiling Views
- Unit 8 - Preferences
- Unit 9 - Extension Points
- Unit 10 - Conclusion

# Eclipse Test and Performance Tools Platform (TPTP)

## Unit 2 - Technology Overview

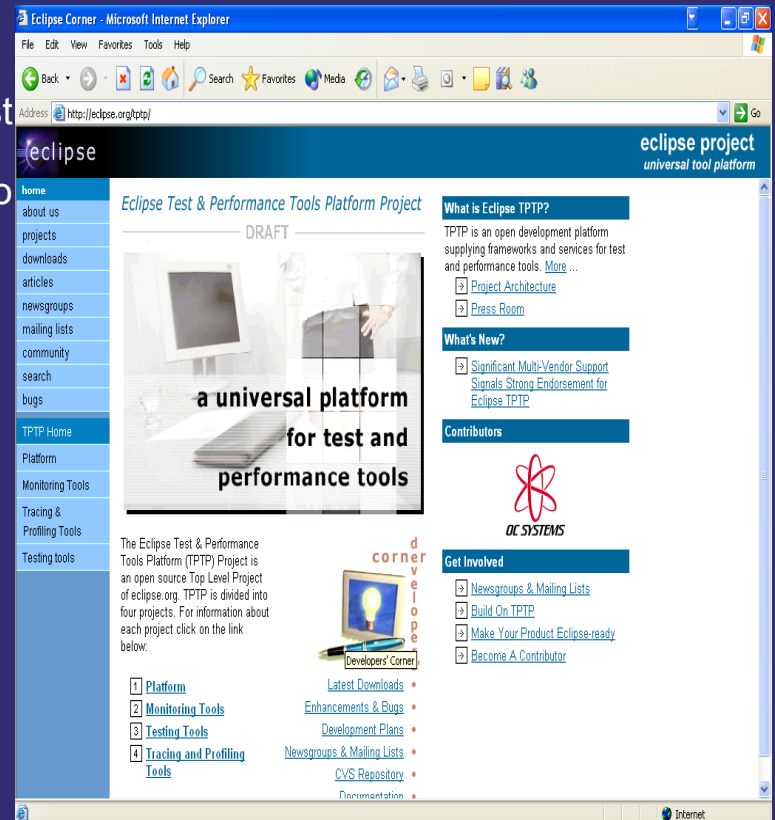
# Eclipse

- Formed in November 2001
- An open source software development project dedicated to provide a robust, full-featured, commercial-quality, industry platform for the development of highly integrated tools
- Composed of projects, each of which is overseen by a Project Management Committee (PMC) and governed by its Project Charter
- Each project is composed of its own subprojects and is licensed under the CPL (EPL starting 2005)



# Test and Performance Tools Platform (TPTP)

- Eclipse Tools Project, Dec 2002
- Formerly Hyades
- An integrated test, trace and monitoring environment, based on Eclipse
- Provides an open source platform for Software test and performance tools platform
- Aims to move software quality practices earlier into the application development cycle
- Provides a unified data model, a normative user experience and workflow, and a united set of APIs and reference tools that work consistently across the range of targets
- Aims to bring software test and performance tools into the Eclipse environment in a consistent way that maximizes integration with tools used in the other processes of the software lifecycle
- Reduce the cost and complexity of implementing effective automated software quality control processes
- Share data through an OMG-defined test profile model, Common Base Event Model and trace model implemented via the Eclipse Modeling Framework (EMF)

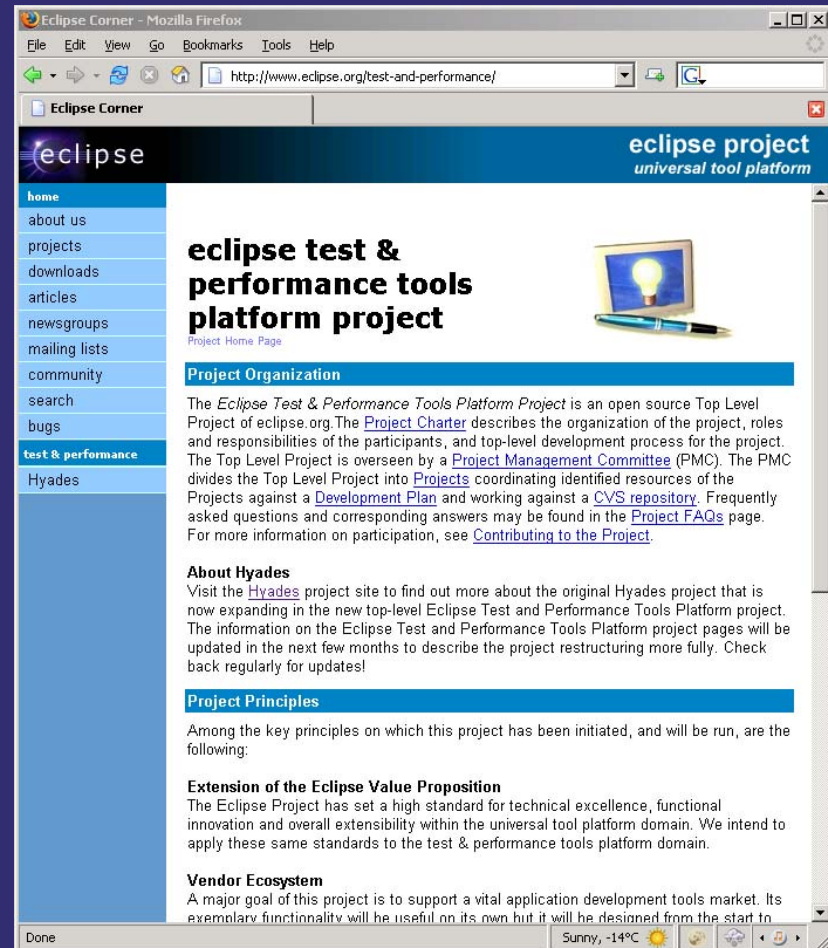


# The Eclipse Test and Performance Tools Platform Project

- Eclipse top-level project created by the Eclipse Foundation in August 2004
- Based on and extends the TPTP Project
- To build a generic, extensible, standards-based tool platform
- Provides an open development platform supplying frameworks and services for test and performance tools that are used throughout the software lifecycle
- Provides a platform for software developers to create specialized, differentiated, and interoperable offerings for world class test and performance tools
- Supports a spectrum of computing systems, from standalone through highly-distributed, and from embedded to enterprise

# Project Principles

- Extension of the Eclipse Value Proposition
- Vendor Ecosystem
- Vendor Neutrality
- Standards-Based Innovation
- Agile Development
- Inclusiveness & Diversity



# TPTP Project Structure

- **TPTP Platform Project**

covers the common infrastructure in the areas of user interface, EMF based data models, data collection and communications control, remote execution environments and extension points

- **TPTP Monitoring Tools Project**

collects, analyzes, aggregates and visualizes data that can be captured in the log and statistical models

- **TPTP Testing Tools Project**

provides specializations of the platform for testing and extensible tools for specific testing environments, initially 3 test environments: Junit, manual and URL testing

- **TPTP Tracing and Profiling Tools Project**

extends the platform with specific data collection for Java and distributed applications that populate the common trace mode, also viewers and analysis services

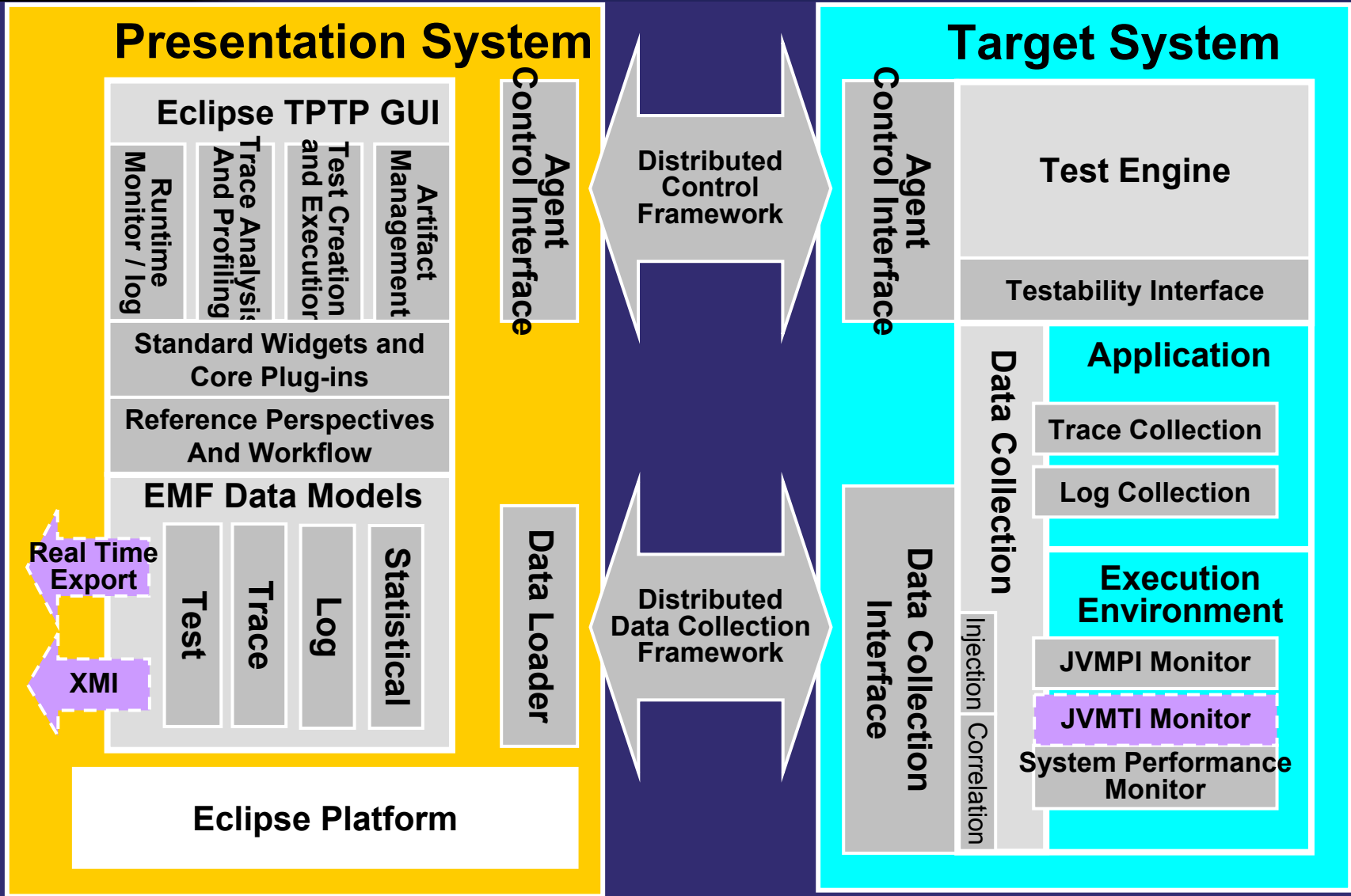
# Participants

- Intel
- IBM
- Scapa Technologies
- Compuware
- Computer Associates
- OC Systems
- SAP
- FOKUS

# Profiling Tool

- Broadly useful for performance analysis and for gaining a deeper understanding of a Java program
- Consists of the Profiling and Logging Perspective and a number of graphical and tabular views
- Enables you to profile and interact with your applications, to work with profiling resources, and to examine your applications for performance and memory usage problems
- Help you to visualize and understand your program execution, pinpoint the operations that take the most resource, as well as to explore patterns of program behavior
- Enables you to test your application's performance early in the programming development cycle for improvements

# Eclipse TPTP Architecture

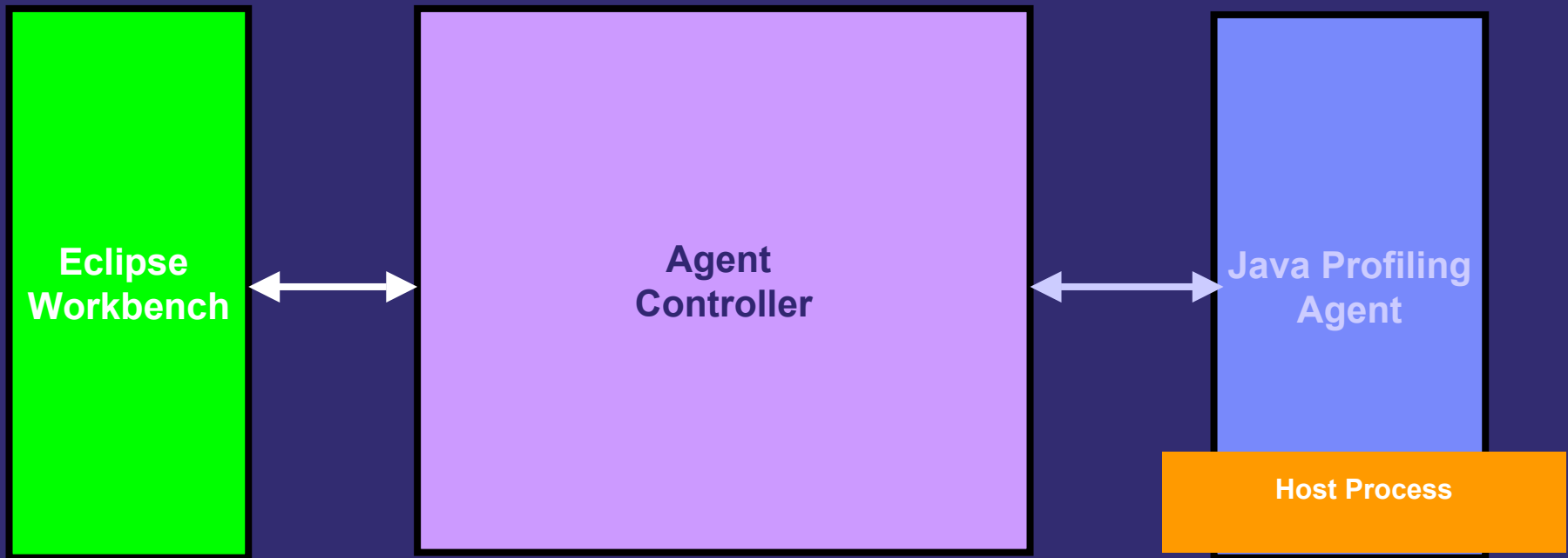


TBD

# Agent Controller

- A daemon process that resides on each deployment host and enables client applications to launch host processes and interact with agents that coexist within host processes
- Contains a server that has the capability to launch and manage local or remote applications from a local TPTP workbench
- The Java profiling engine included uses the JVMLI to profile local or remote Java applications from a local TPTP workbench
- Launch and profile local or remote Java applications and import local or remote logs
- Platform support : Windows, Linux, Solaris, HP-UX, AIX, Z/OS, OS/400

# Agent Controller



# Agent Controller Terminology

- **Host process**  
The process that contains the application under test
- **Agent**  
A reusable binary file that provides services to the host process, and more importantly, provides a portal by which application data can be forwarded to attached clients
- **Client**  
A local or remote application (eg Eclipse Workbench) that is the terminal destination of host process data that is externalized by an agent
- **The Agent Controller**  
A daemon process that resides on each deployment host and provides the mechanism by which client applications can either launch new host processes, or attach to agents that coexist within existing host processes. The client can reside on the same host as the Agent Controller, or it can be remote. The Agent Controller can only interact with processes on the same node, and it is required on the same machine the targeted JVM is

# The Java Profiler

- A library that attaches to a JVM to capture and record the Java application's behavior
- Is a type of agent managed by the Agent Controller
- Runs in the JVM (Java Virtual Machine) process and receives notifications of JVM events, based on the JVMPI (Java Virtual Machine Profiler Interface). This agent is best used to identify performance details such as classes or methods responsible for the poor execution performance, also be used to analyze application heap and find memory leaks
- Output from the profiling agent is in the form of XML fragments
- Can be launched from the TPTP workbench, Applications can be in workbench's workspace or binaries that are on the file system
- Can also be invoked using the `-Xrun JVM` option in command line

# Eclipse Test and Performance Tools Platform (TPTP)

## Unit 3 - Profiling and Logging Perspective

# Profiling and Logging Perspective

- The profiling tools available in the Profiling and Logging perspective provide comprehensive information about the performance of an application
- The profiling tool (in this scenario i.e. – Java Profiling) provides information pertaining to
  - JVM performance
  - Object allocations and references
  - Garbage collection
  - Object methods performance
  - Object->Object interactions
  - Thread interactions to name a few

# Profiling and Logging Perspective

- Profiling and Logging perspective provides combinations of views and editors that are best suited to performing application profiling
- Profiling Monitor view
  - Multiple views provide the profiling and logging perspective - Profiling Monitor view is the Primary view
  - Project
  - Monitor
  - Host
  - Process & Agent

# Profiling and Logging Perspective

The screenshot displays the Eclipse Profiling Monitor perspective. The main view shows a tree structure under the 'DefaultMonitor' node. The tree includes:

- nk75** (Host)
  - carModel.CarModel at nk75 [ PID:1740 ]** (Java process)
    - <monitoring> Profiling (23/08/04 3:37:32 PM) (Profiling agent)
      - Basic Memory Analysis
      - Execution Time Analysis
      - Method Code Coverage
  - com.ibm.etools.actools.web.onlinehelp.help30 at nk75 [ PID:3700 ]** (Java process)
    - <monitoring> Profiling (23/08/04 3:47:34 PM) (Profiling agent)
- nklab.torolab.ibm.com** (Host)
  - unknown at nklab.torolab.ibm.com [ PID:1244 ]** (Java process)
    - <monitoring> Profiling (23/08/04 3:46:15 PM) (Profiling agent)
      - Basic Memory Analysis
      - Execution Time Analysis
      - Method Code Coverage

Annotations with red arrows provide context:

- Monitor contains processes from hosts that are being monitored:** Points to the 'DefaultMonitor' node.
- Hosts being monitored in a profiling session:** Points to the 'nk75' and 'nklab.torolab.ibm.com' host nodes.
- Java processes associated with a host:** Points to the 'carModel.CarModel...' and 'com.ibm.etools.actools...' process nodes.
- Profiling agents:** Points to the '<monitoring> Profiling...' nodes under each process.

# Profiling and Logging Perspective

- Profiling and Logging perspective provides resources to administering and managing profiling
- Profiling resources are organized to provide granularity of usage
- Profiling resources
  - Project: Make a project of your profiling effort
  - Monitor: Aggregate different processes and agents
  - Host: The host you are profiling
  - Process: Very simply the executing program
  - Agent : Provides services to a process, a mechanism by which process data can be sent to (attached) clients
  - Profiling Type: Group profile data collection

# Profiling Monitor View

- A Profiling session creates numerous resources
- Administer and Analyze Profiling activity
- Object Control: Choice of action depending on the type of the object
  - Start and Stop the monitoring on an agent
  - Attach and Detach the agent from process
  - Terminate a process
- Context menu is resource-sensitive

# Profiling Monitor View

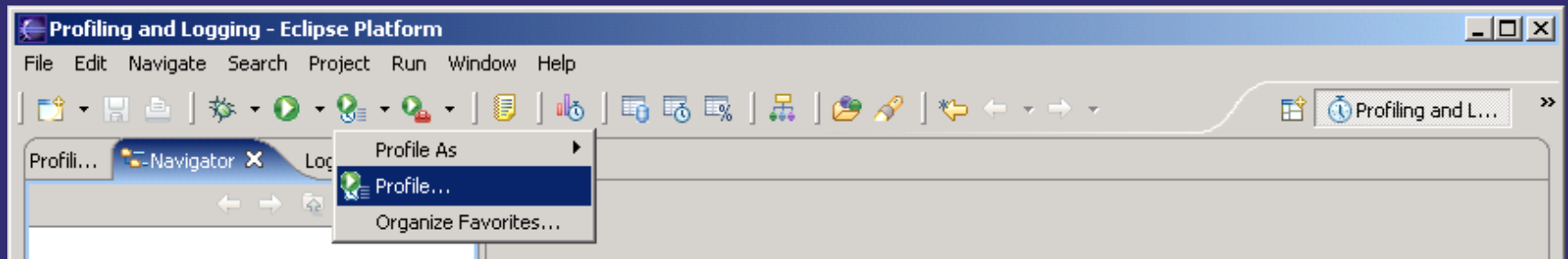
- Project:
  - Container of the profiling resources
- Monitor:
  - A logical container for the profiling information that is collected from a group of agents. The views at the monitor level show data from these agents. Monitors are useful for aggregating processes and agents from a distributed application
- Host:
  - Owns the processes that are profiled. A host runs processes. You can specify a host either by its name or by its IP address
- Process & Agent:
  - In the Agent Controller architecture model, an agent is a binary file that provides services to the host process by which application data can be transferred to attached clients

# Eclipse Test and Performance Tools Platform (TPTP)

## Unit 4 – Launch an application

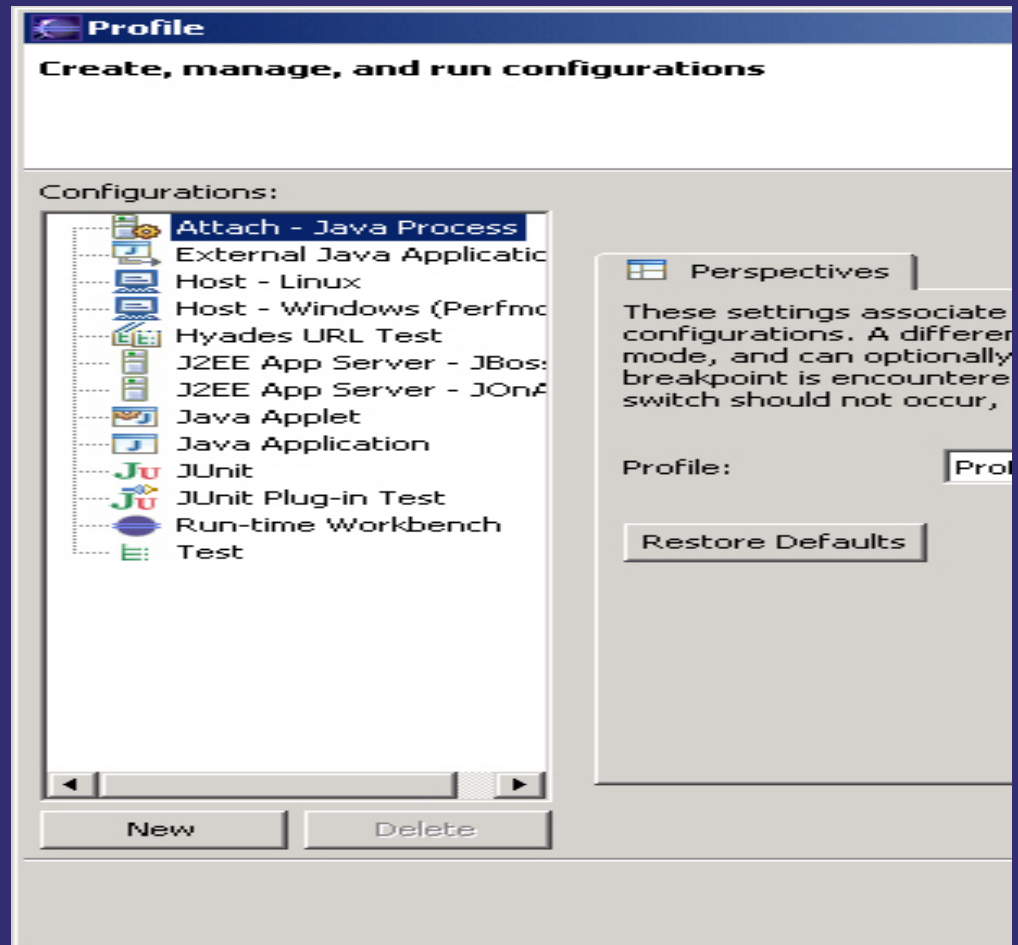
# Launch Configuration

- Profiling Tool provides the ability to attach to a running application or to launch an application for profiling with launch configurations
- Attaching an application means that a monitor is created to contain the results observed by an associated agent
- Launch a process means that the process is started a process with an agent associated to the process



# Configurations

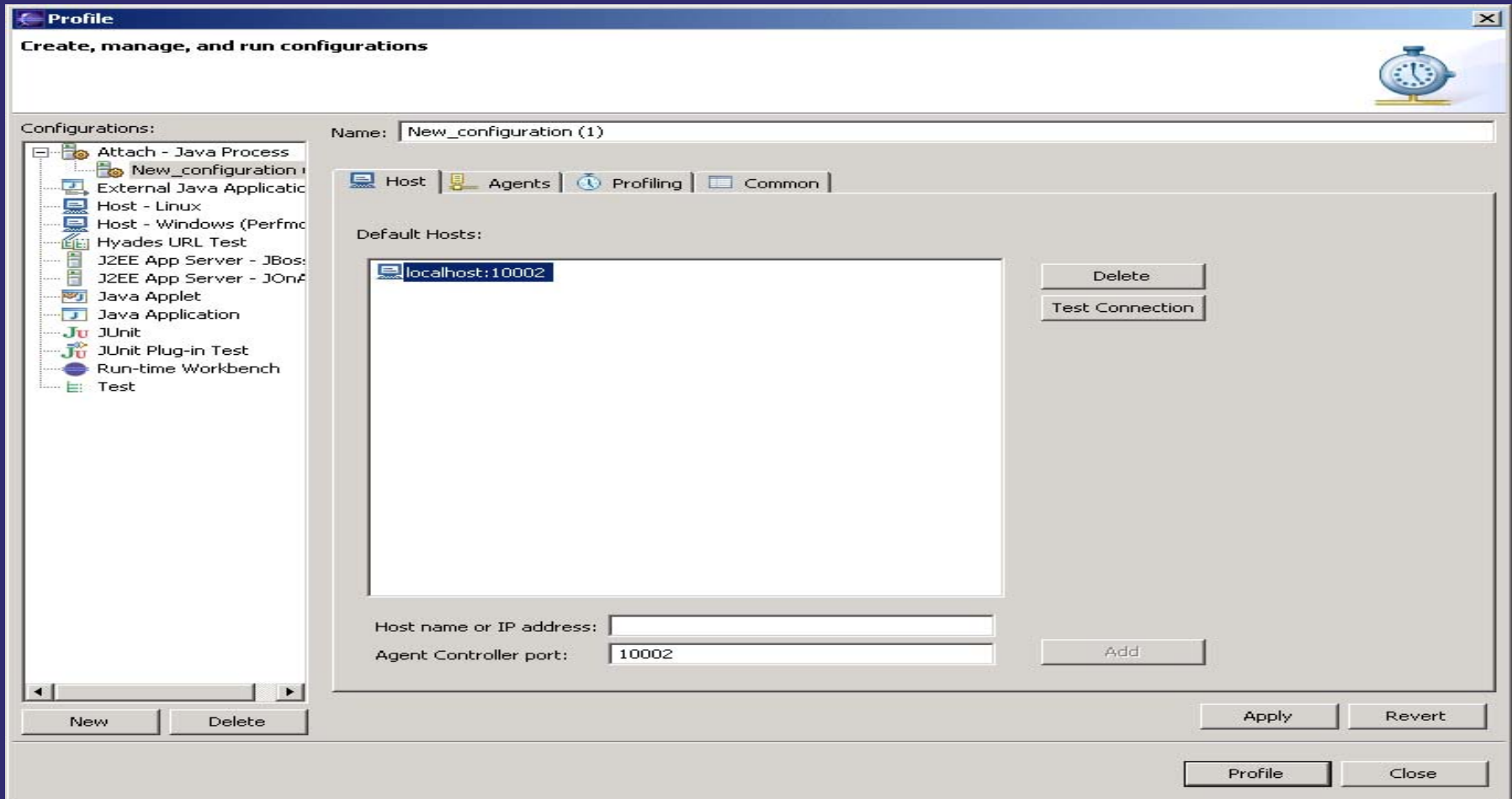
- Attach Java Process:
  - local or remote
- External Java Application:
  - local or remote
- Java Applet : workbench (local)
- Java Application :
  - workbench (local)
- Run-time Workbench : local
- Other launch configurations:
  - for Statistical data
    - Host
    - J2EE App Server
  - for Test
    - URL, JUnit, Manual



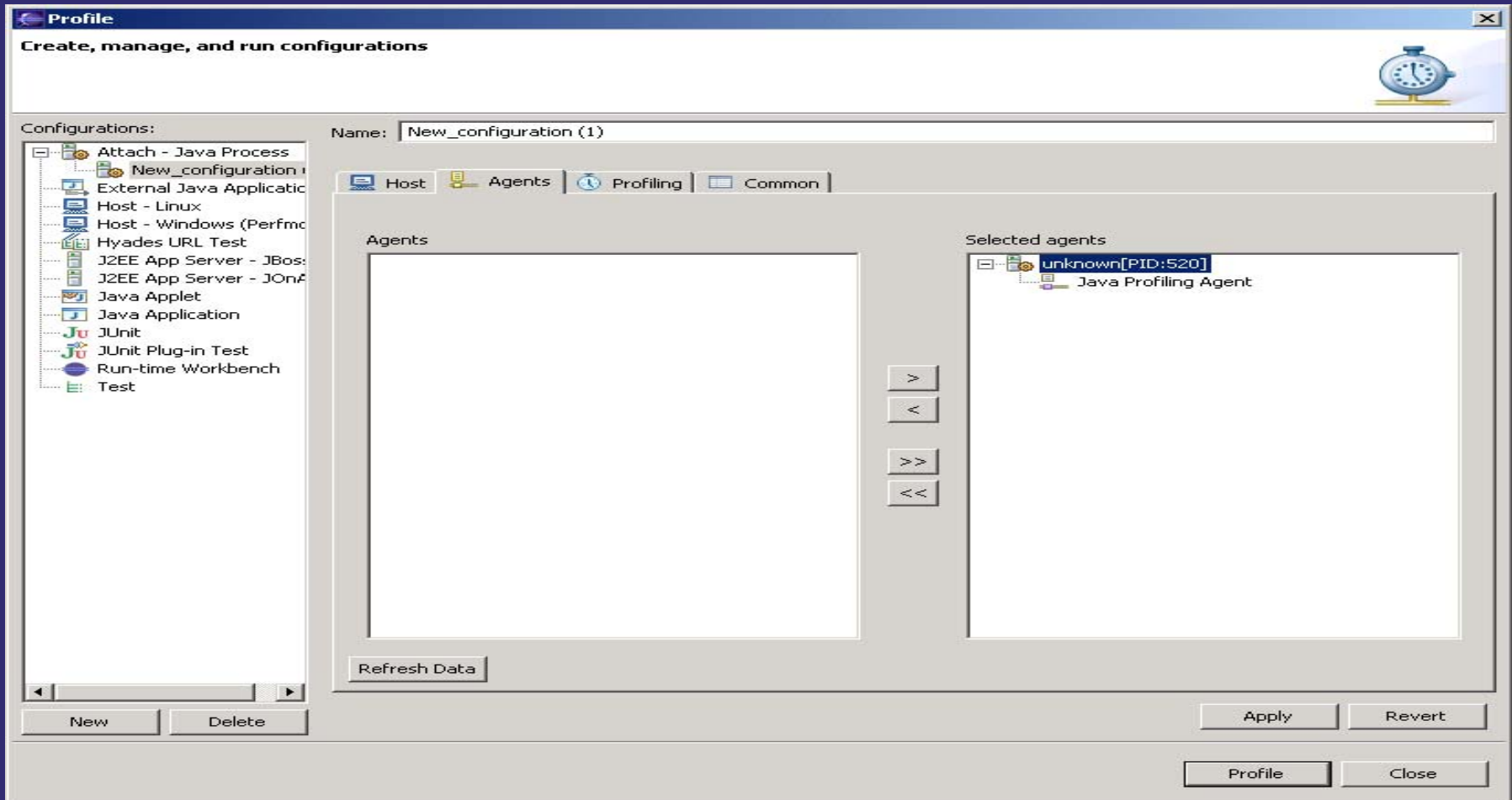
# Configuration tabs

- Each type of launch configuration defines a group of tabs that collect and display information about the configuration
- Tabs for profiling:
  - Host tab : defines the location of the process to be launched or attached
  - Agents tab : used for attach launch configurations, list the agents available for attach
  - Profile tab : defines detail of data collection on the profile process

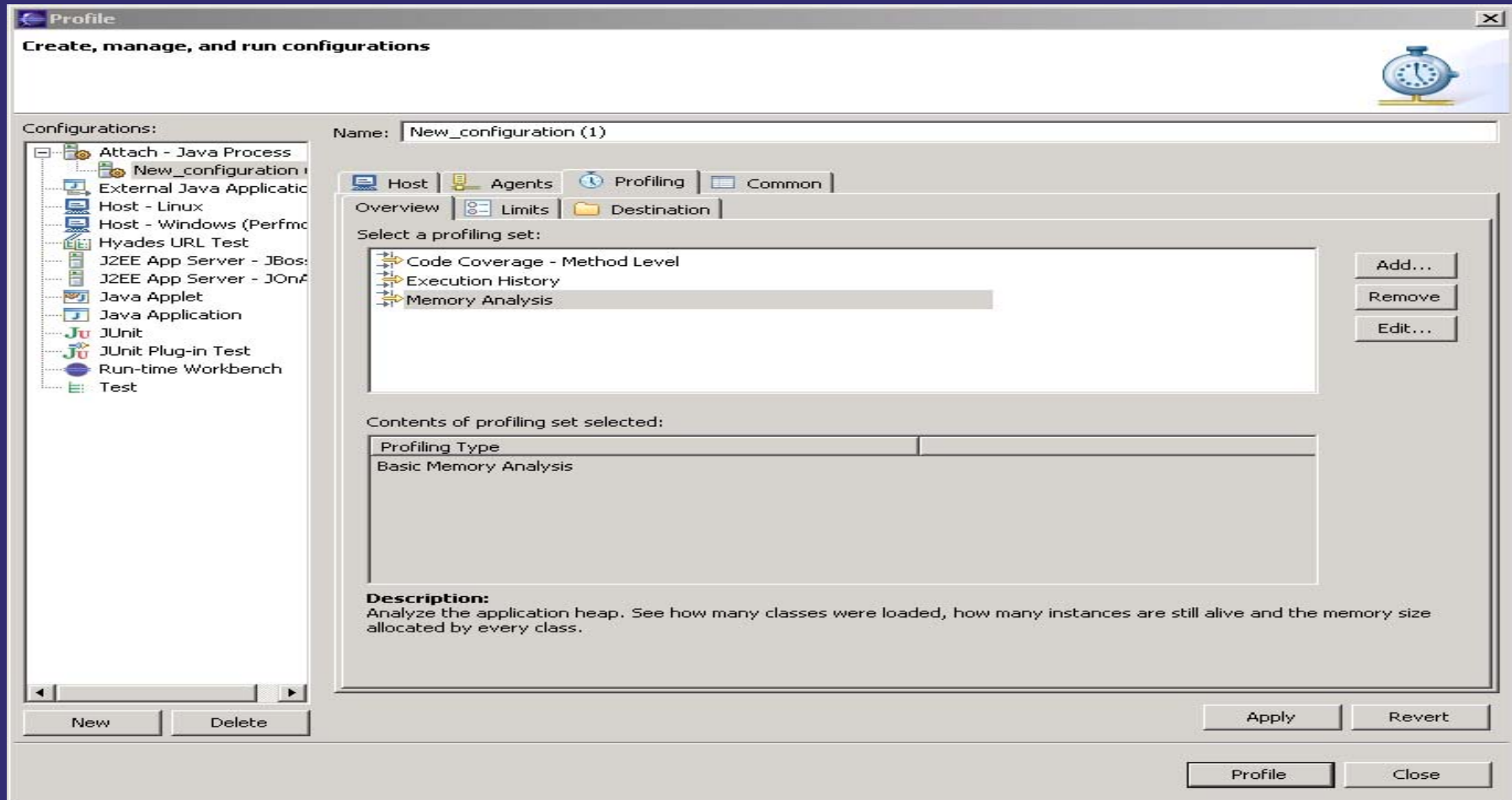
# Host tab



# Agents tab



# Profile tab



# Eclipse Test and Performance Tools Platform (TPTP)

## Unit 5 – Attaching to a Java application

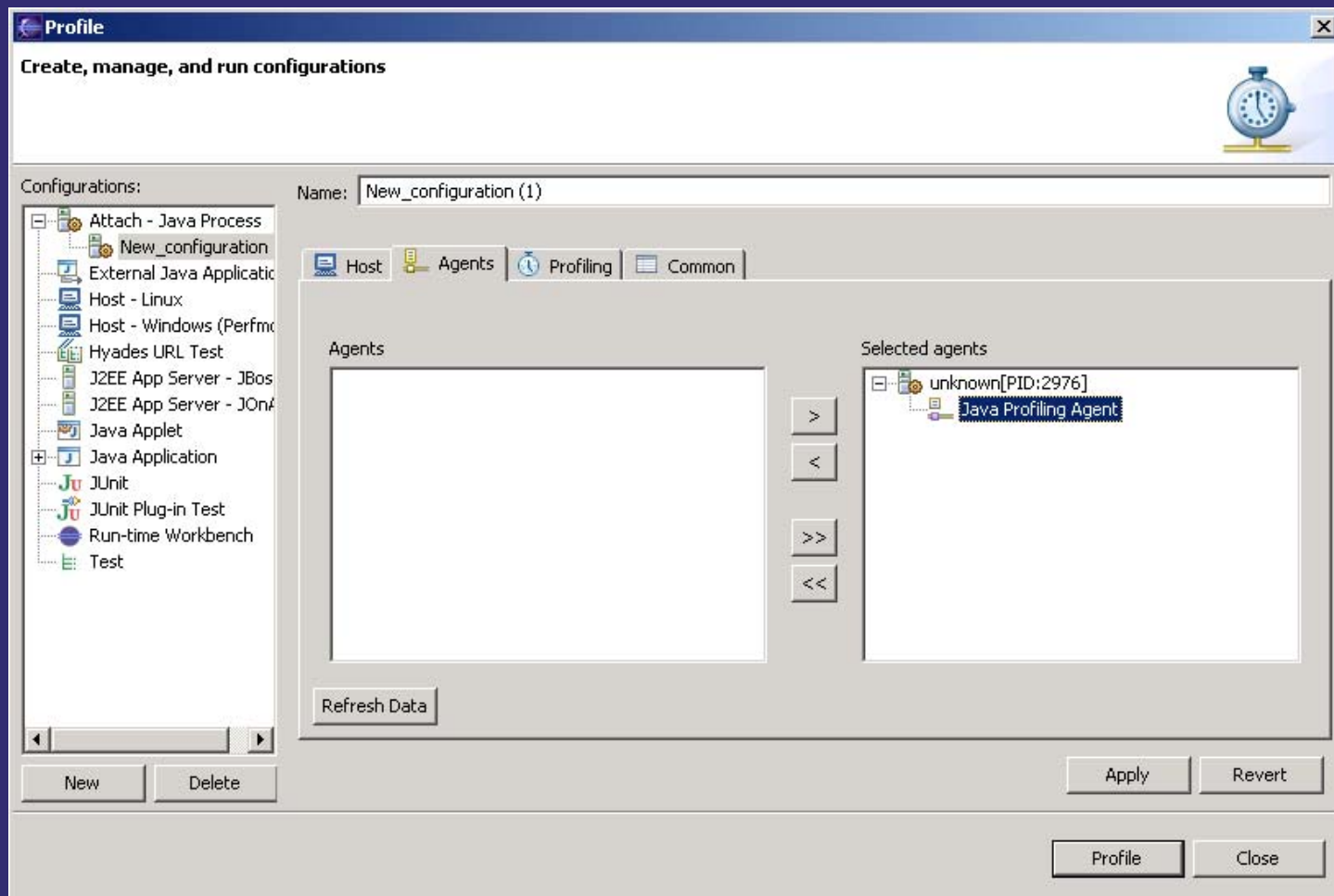
# Attaching to a Java application

- Attaching to a Java Application
  - Local Java Application
  - Remote Java Application
- Attach to a running application or to launch an application for profiling.
  - Attaching : Gather data observed by an associated agent.
  - Launching : the process is started with an agent associated to the process.

# Attaching to a Java application

- What is happening?
  - A logical representation of the Java process is created in the Profiling Monitor view
  - The process object i.e. logical representation of the Java process, is identified both by name and an ID number (PID) that appears in the view along with the associated agents

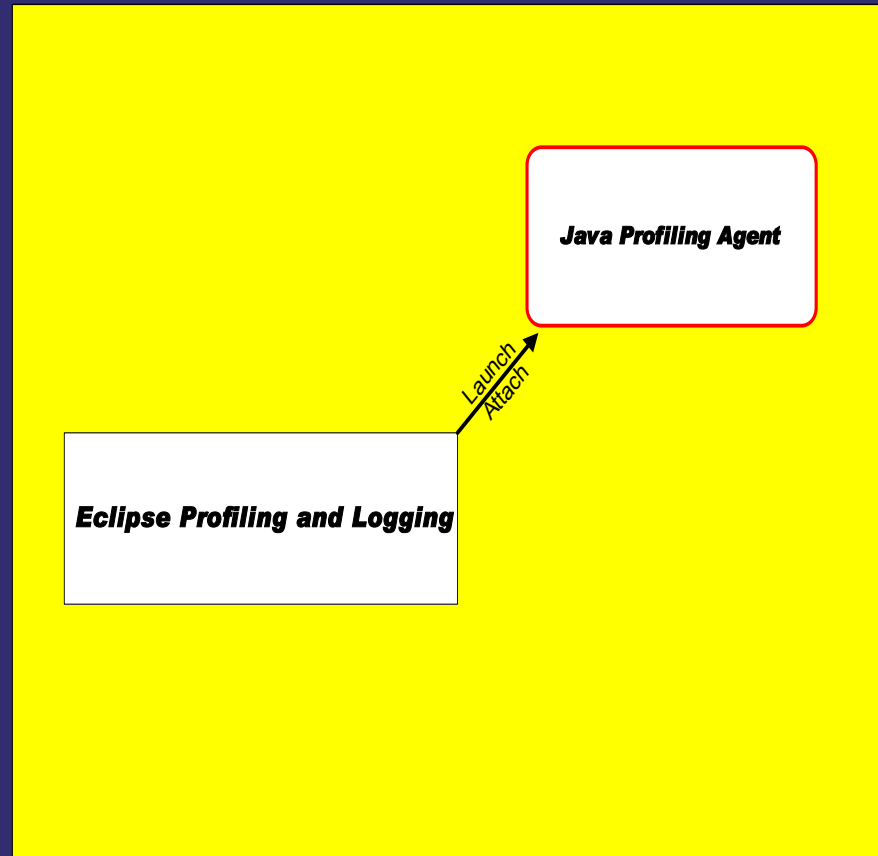
# Attaching to a Java application





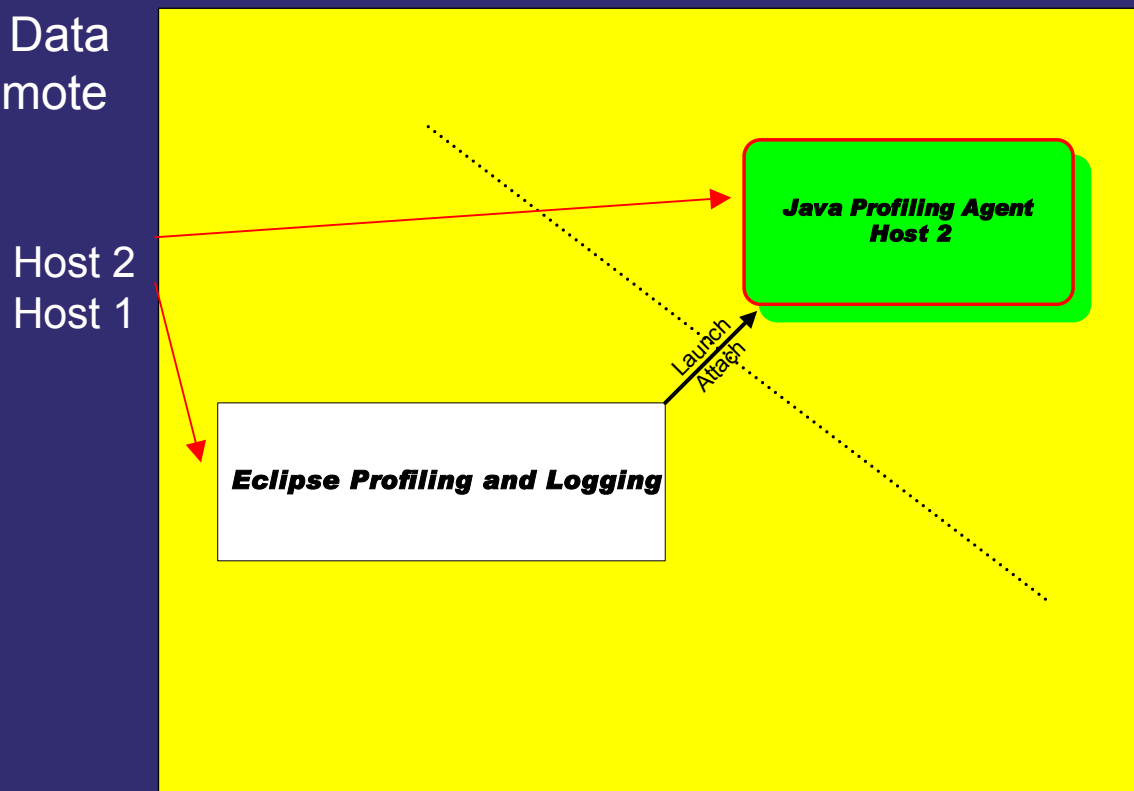
# Attaching to a Java application - Local

- Profiling and Logging perspective to attach to a running process locally



# Attaching to a Java application - Remote

- Required to Install Eclipse Data collection engine on the remote machine

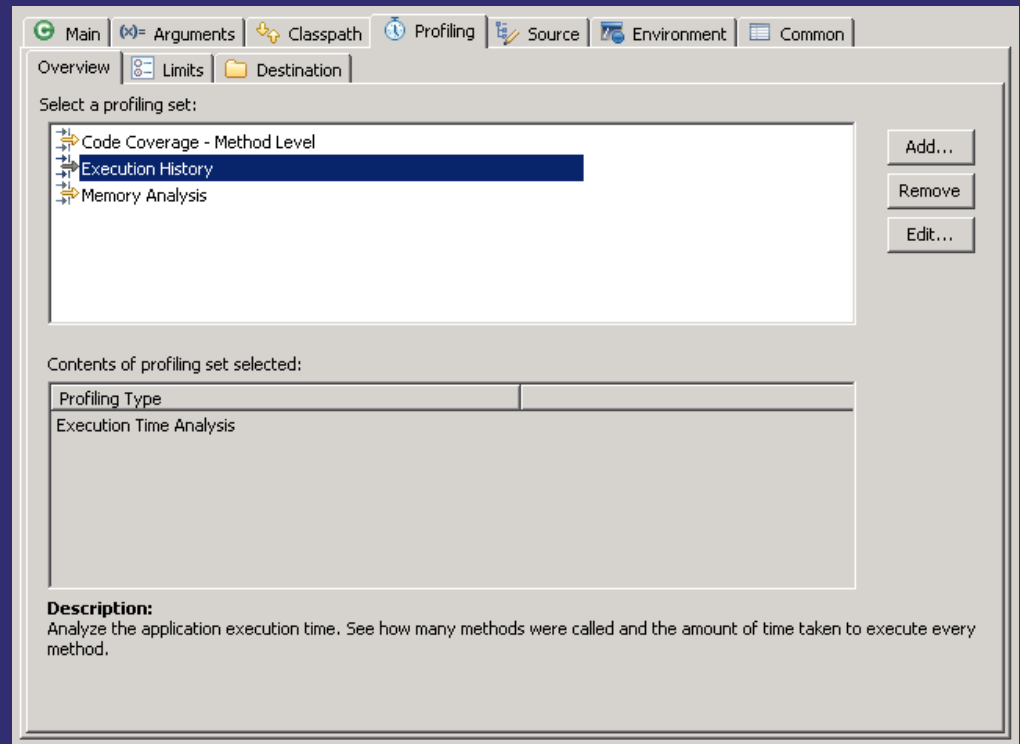


# Eclipse Test and Performance Tools Platform (TPTP)

## Unit 6 – Profiling Options

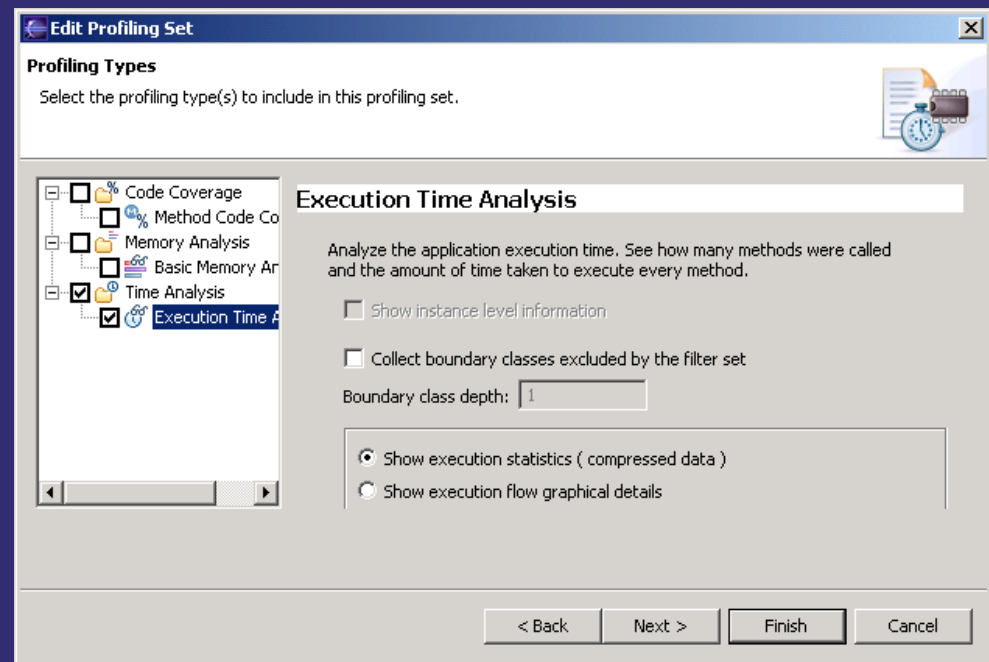
# Profiling Options

- The profiling options are used to control the type of profiling data collected
  
- Profiling Set
- Profiling Type
- Profiling Filter
- Profiling Limits
- Profile to File



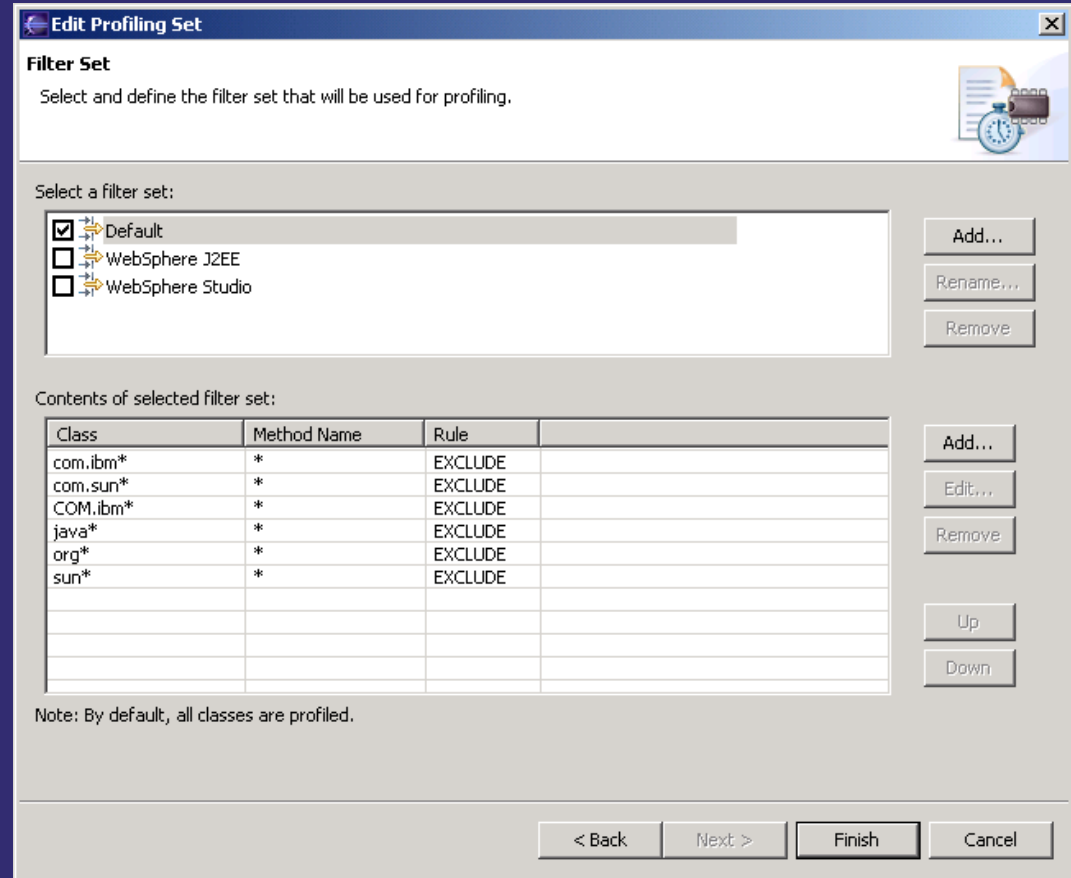
# Profiling Set and Profiling Type

- Specify an existing or build a set of profiling type to control the type of profiling data
- Pre-defined Sets:
  - Execution history
  - Memory analysis
  - Method Level Coverage



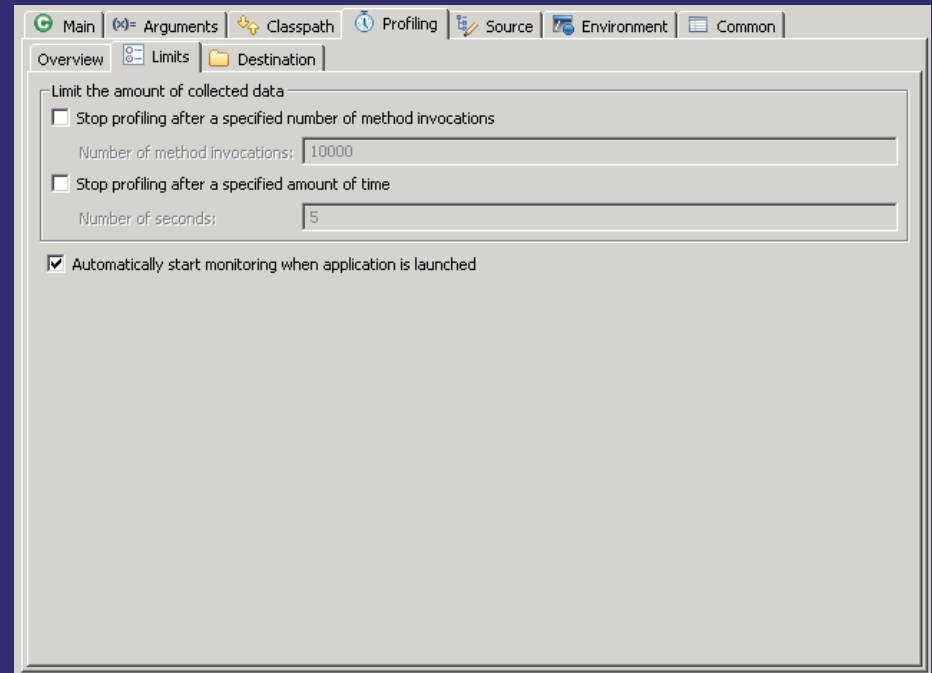
# Profiling Filter

- Limit the classes and method profiled
- Avoid unnecessary visual clutter and speed up the profiling task



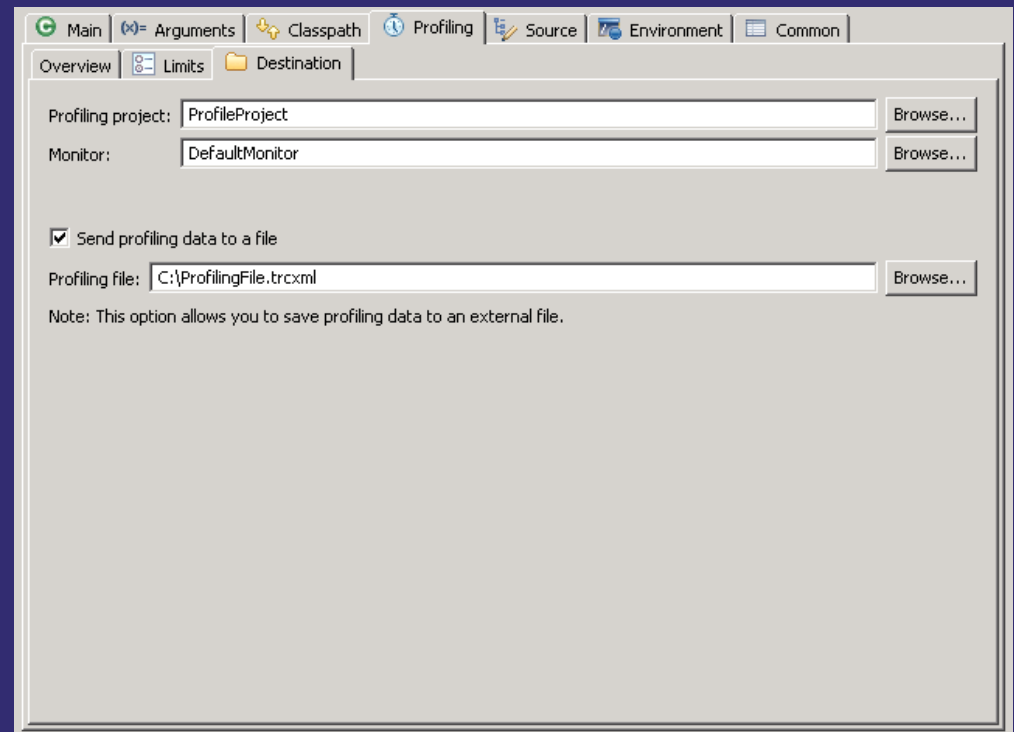
# Profiling Limits

- Profiling Limits page can be used to limit the amount of data collected from the application
- Specify the amount of data you want collected by specifying a limit on method invocations or on time.



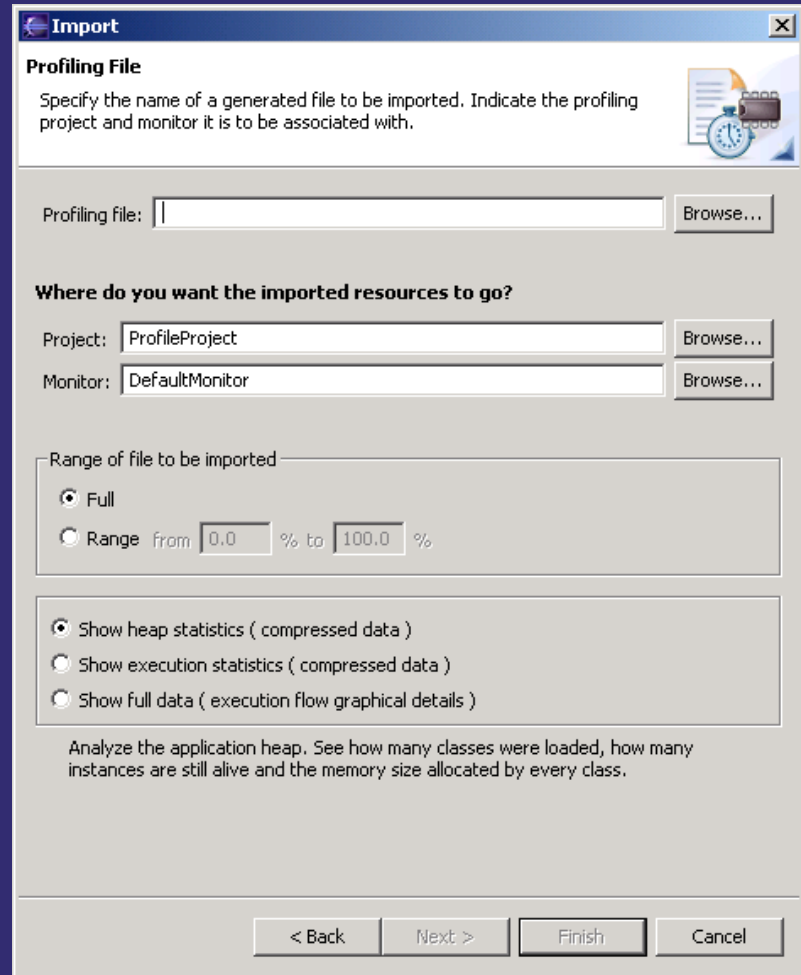
# Profiling File

- Route the profiling events to an xml file instead of the profiling views



# Import profiling file

- Profiling file can be imported into workbench for visual presentation and analysis
- File -> Import... -> Profile File
- Select range of file to be imported



# Eclipse Test and Performance Tools Platform (TPTP) Unit 7 – Profiling Views

# Profiling Views

- Number of views to visualize and organize profiling data
  - Memory Statistic view
  - Execution Statistic view
  - Coverage Statistic view
  - Object Reference view
  - Execution Flow view and table
  - Method Invocation view and table
  - UML2 Trace Interactions view

# Statistical Views

- Profiling Type oriented
  - Execution
  - Memory
  - Coverage
- Drill down capability
  - Package
  - Class
  - Method
  - Instance
- Additional Information available on toolbar
  - Choose columns
  - Show deltas
  - Show as percentage
  - Open Source
  - Export to HTML
- View shortcut on selection
  - Selection sensitive on context menu
  - Show Object Reference on Class
  - Show Method Invocation on Method

The screenshot shows the Eclipse IDE interface with the 'Profiling and Logging' window open. The window title is 'Profiling and Logging - CarModel.java - Eclipse Platform'. The main area displays 'Execution Statistics - CarModel at e...' with a table of performance metrics. The table has columns for Package, Base Time (sec...), Average Base T..., Cumulative Tim..., and Calls. The data is as follows:

>Package	Base Time (sec...)	Average Base T...	Cumulative Tim...	Calls
[short	0.000000	0.000000	0.000000	0
byte	0.000000	0.000000	0.000000	0
CarModel	16.251306	0.090789	16.260242	179
CarModel()	0.014827	0.000087	0.020630	171
main(java.lang.String[]) void	16.236289	16.236289	16.260242	1
simulateCarUsage(CarModel)...	0.000190	0.000027	0.003323	7
char	0.000000	0.000000	0.000000	0
Door	0.004641	0.000014	0.004954	342
close() void	0.000000	0.000000	0.000000	0
Door()	0.004641	0.000014	0.004954	342
open() void	0.000000	0.000000	0.000000	0
Engine	0.001793	0.000009	0.001793	192
Engine()	0.000165	0.000001	0.000165	171
rev() void	0.000542	0.000077	0.000542	7
start() void	0.000547	0.000078	0.000547	7
stop() void	0.000540	0.000077	0.000540	7
int	0.000000	0.000000	0.000000	0
long	0.000000	0.000000	0.000000	0
short	0.000000	0.000000	0.000000	0
Wheel	0.001234	0.000002	0.001234	691
align() void	0.000549	0.000078	0.000549	7
Wheel()	0.000684	0.000001	0.000684	684
Window	0.001268	0.000004	0.001268	349

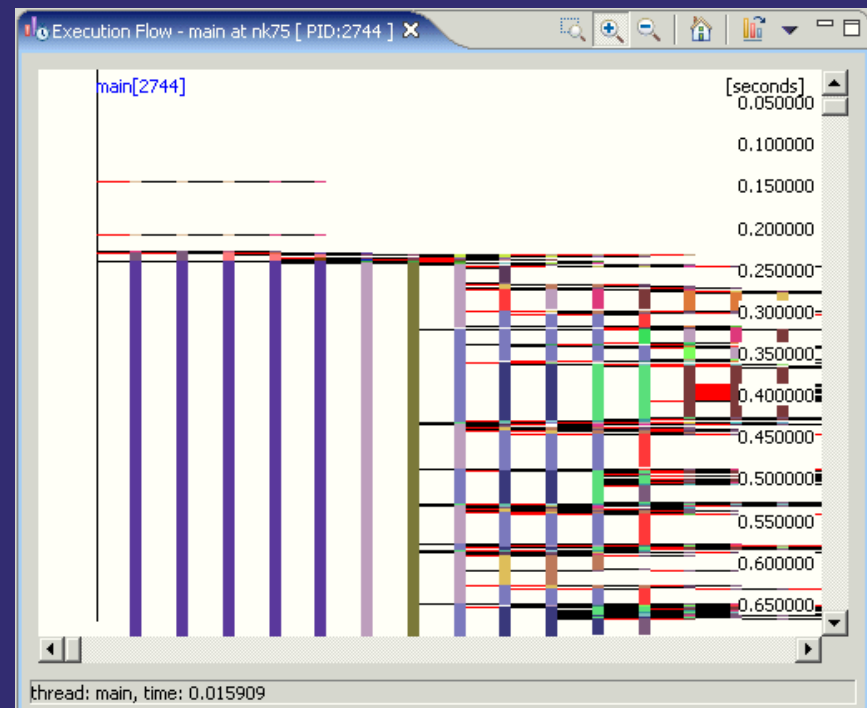
# Object Reference View

- Displays references by a set of objects.
- Examine data structures
- Help identify memory leaks
- To find unexpected references
- How? Collect Object References action
- Show reference TO or reference BY.

>Show Reference By	Package	Size	Number of References
[-] process2	org.eclipse.hyades.samples.processes	8	
[-] process2	org.eclipse.hyades.samples.processes	56	Is referenced by 1 object(s).
[-] [Object.2858	java.lang	56	Is referenced by 1 object(s).
[+] Vector.2857	java.util	24	Is referenced by 1 object(s).
[+] Properties	java.util	96	
PropertyChangeEvent	java.beans	0	
[+] PropertyChangeSupport	java.beans	24	

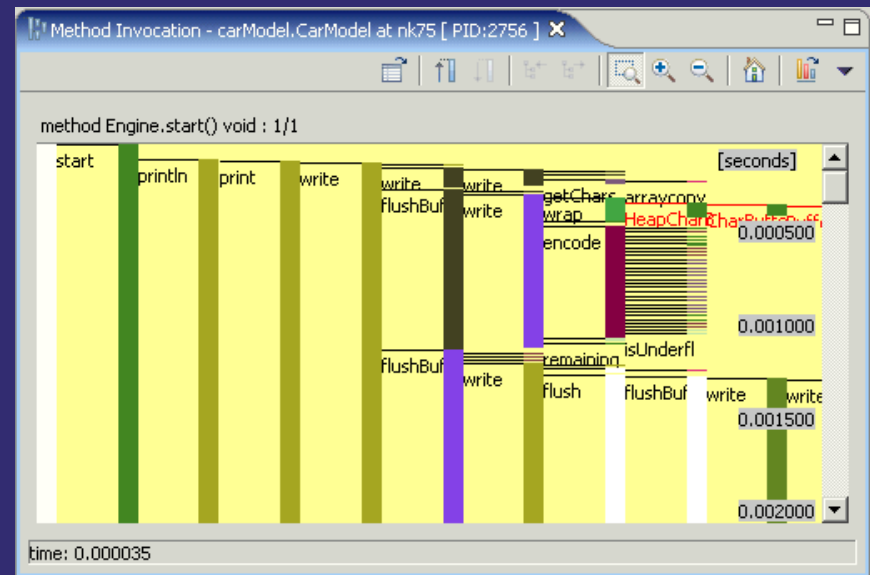
# Execution Flow View

- Give a clear global view on the overall application execution
- Analyze the application execution at the thread level.
- Threads are represented by vertical lines
- Execution stack goes from left to right
- Execution time goes from top to bottom
- Interactive: Zoom In/Out or Select Zoom



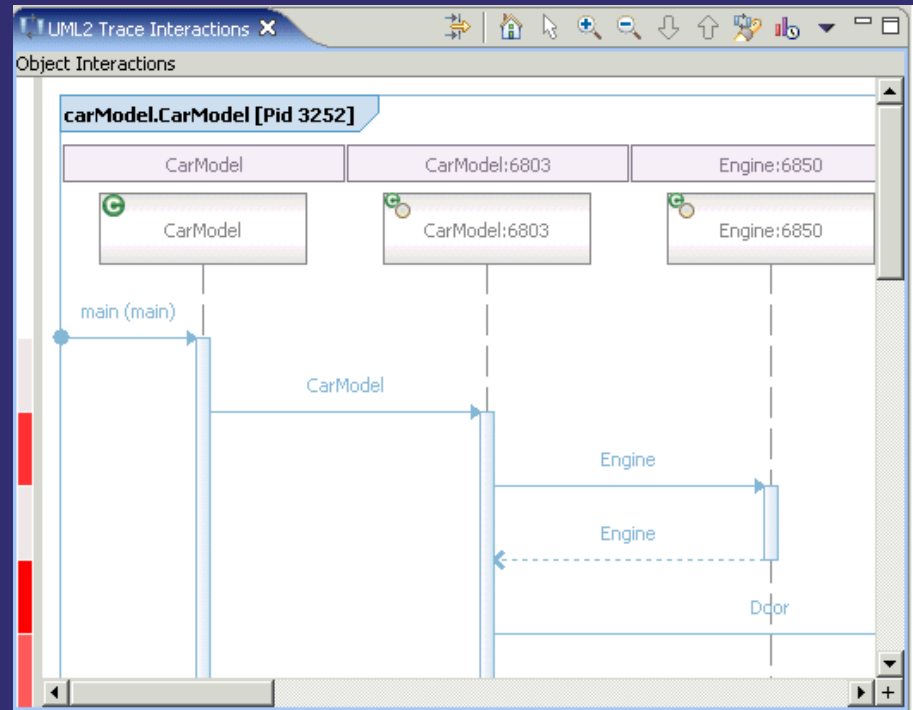
# Method Invocation View

- Graphical representation of the application execution that lets you see the method execution at the method invocation level
- Makes it easier to identify the execution patterns and differences between invocations of the same method
- Traverse between invocation
- Show Caller or Callee
- Open from any Method Selection



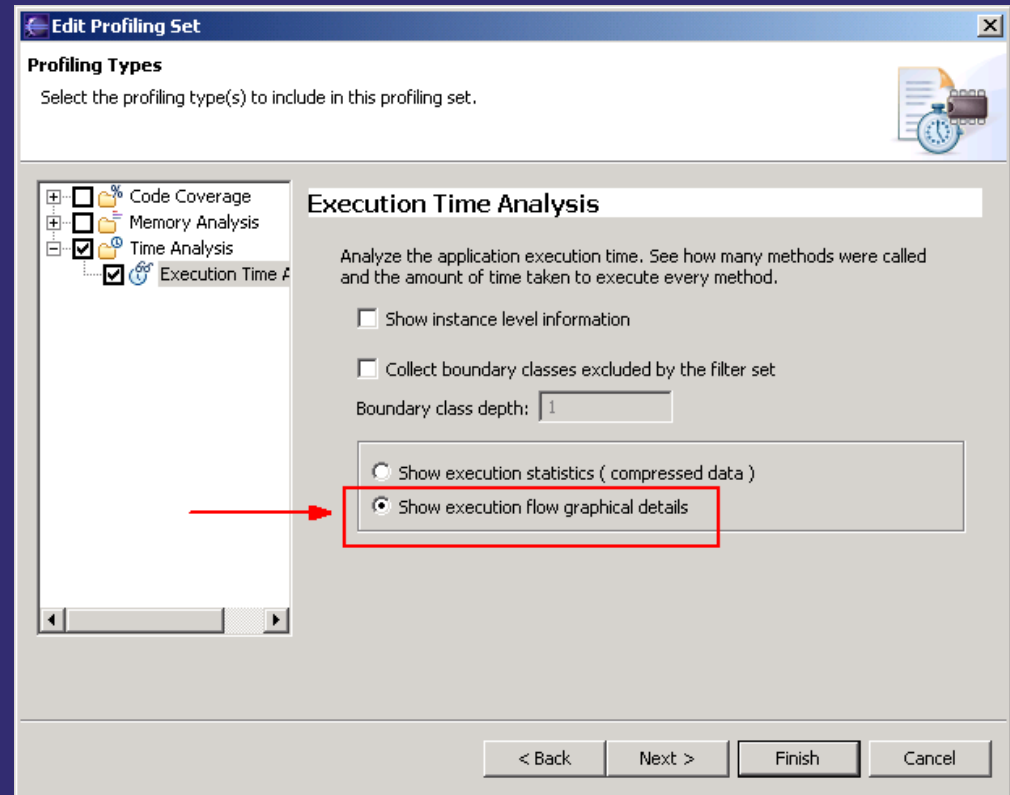
# Interactions Views (UML2 Trace Interactions View)

- The Interactions Views present execution flow of an application according to the notation defined by UML
- Host Interactions
- Process Interactions
- Thread Interactions
- Agent Interactions
- Class Interactions
- Object Interactions



# Enabling Graphical information

- Enable execution flow detail information be loaded and visualize in graphical views
- Execution Flow View
- Method Invocation View
- Interactions Views



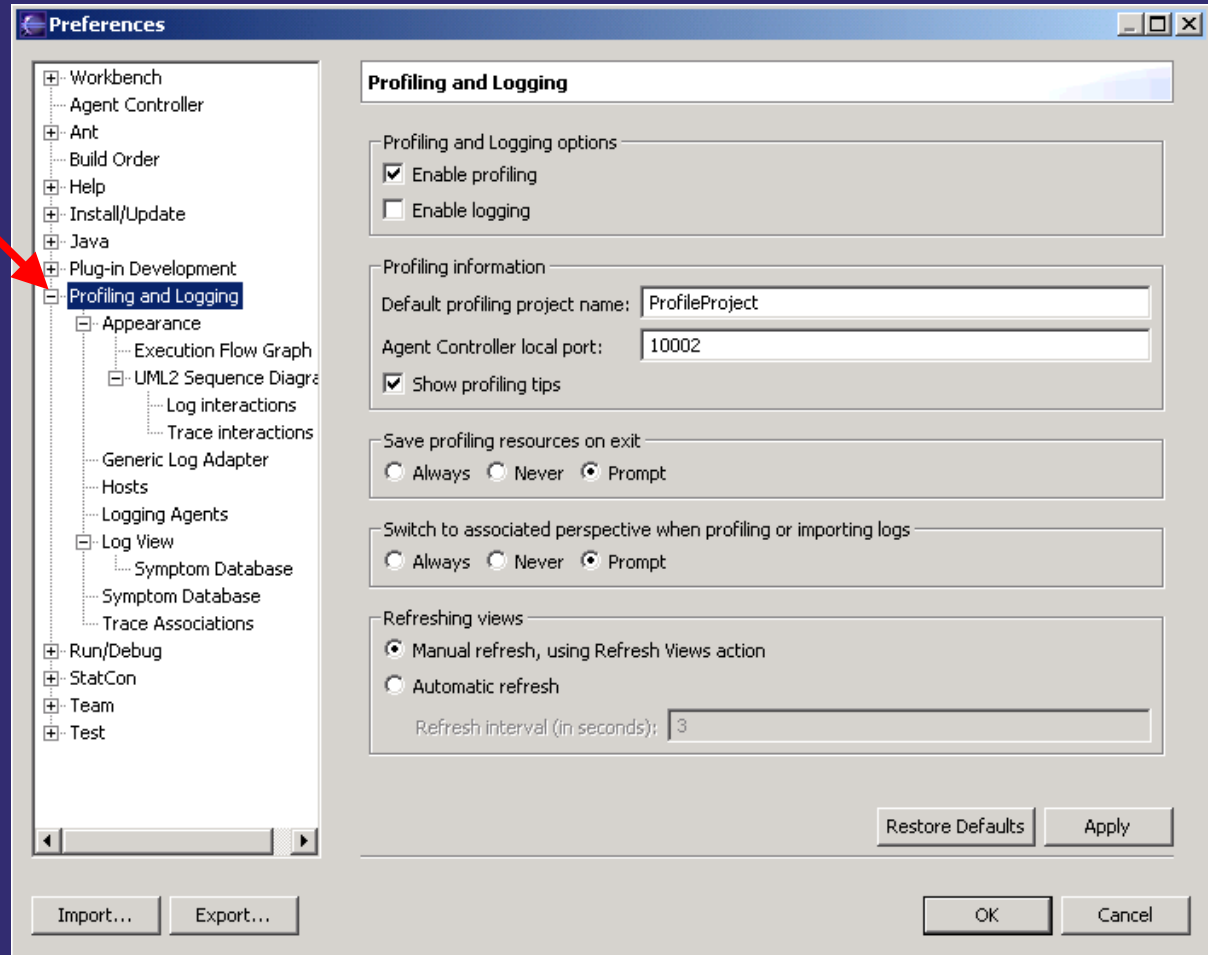
# Eclipse Test and Performance Tools Platform

## Unit 8 - Preferences

# Preferences

- Preferences: Customize or modify Profiling and Logging Preferences
- Enable or disable profiling or logging
- Specify the port number that the agent uses to connect to the Agent Controller
- If you are working with more than one machine, use Hosts to identify them
- To identify logging agents, use Logging Agents
- Associate views to Context menu of resources in the profiling monitor view

# Preferences



# Eclipse Test and Performance Tools Platform

## Unit 9 – Extension Points

# TPTP Extension Points

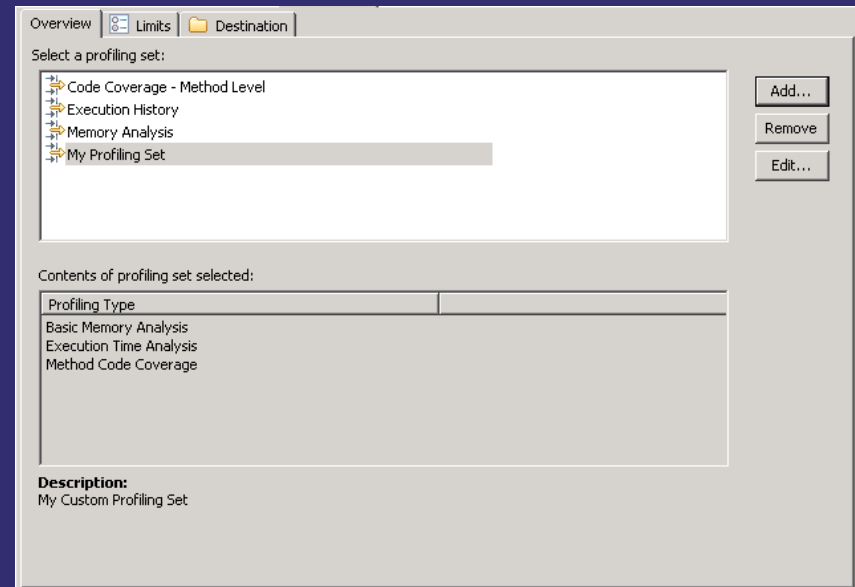
- Allow plug-ins to contribute and extend capability on profiling resources
- Companies can then build its solution on TPTP
- Trace Views
- Profiling Set and Type
- Examples
- More: Log Parser, Analysis Engine, Correlator

# Trace Views

- Plugging an analyzer view into the Profiling and Logging Perspective is relatively simple
- Consists of 3 main tasks
  1. Write a plug-in that extends the extension point `org.eclipse.hyades.ui.analyzerExtensions`, adds an action entry to the **Open With** menu
  2. Add an Eclipse view to the said plug-in
  3. Add code into the view to handle the TPTP profiling data

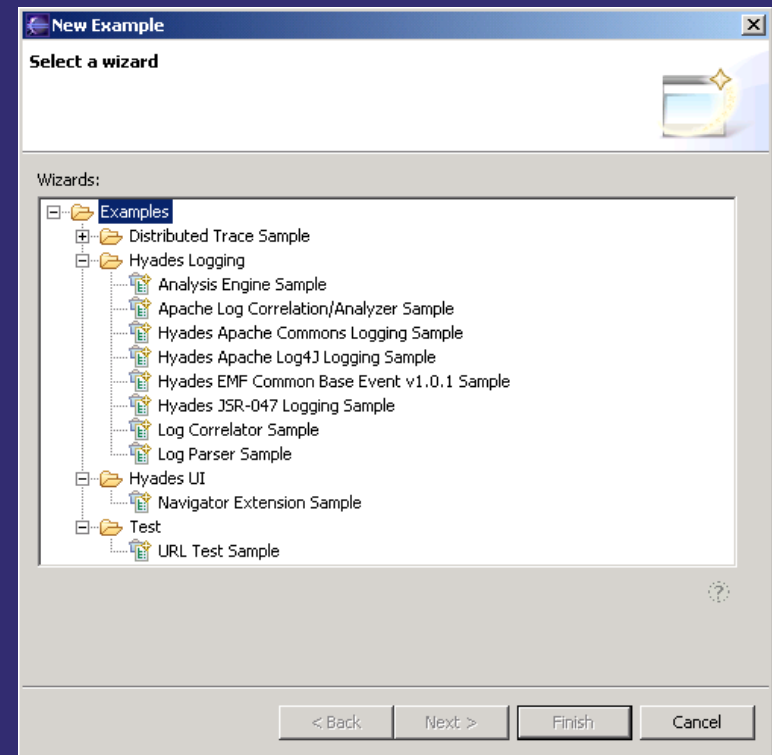
# Profiling Set and Profiling Type

- Profiling Set: A logical grouping of profiling Type
- Defined as extension point but can also be created or edited in user interface
- Extension point:  
org.eclipse.hyades.trace.ui.profilingSet
- Profiling Type: defines profiling options be sent to profiling agent.
- Extension point:  
org.eclipse.hyades.trace.ui.profilingType
- A profiling type can be linked to more than one profiling set



# TPTP Examples

- Designed to simplify the creation of project samples in Eclipse with New wizard
- Project(s) setup and creation, File(s) import, Open file(s) on wizard completion
- Extension point:  
`org.eclipse.hyades.ui.sampleWizards`



# Logging

- Extension points are provided for extending the various logging feature
- Log Parser :
  - Plug in parser for new log file type to be imported in TPTP
- Log Analysis Engine:
  - Plug in new algorithm for analyzing log file entries with defined rules, return list of solutions and action items on known entries.
- Log Correlator:
  - Plug in new algorithm for associating log records between different or same logs, and result be visualized in log interactions view

# Eclipse Test and Performance Tools Platform (TPTP) Unit 10 – Summary

# TPTP Test and Profiling Tools Project

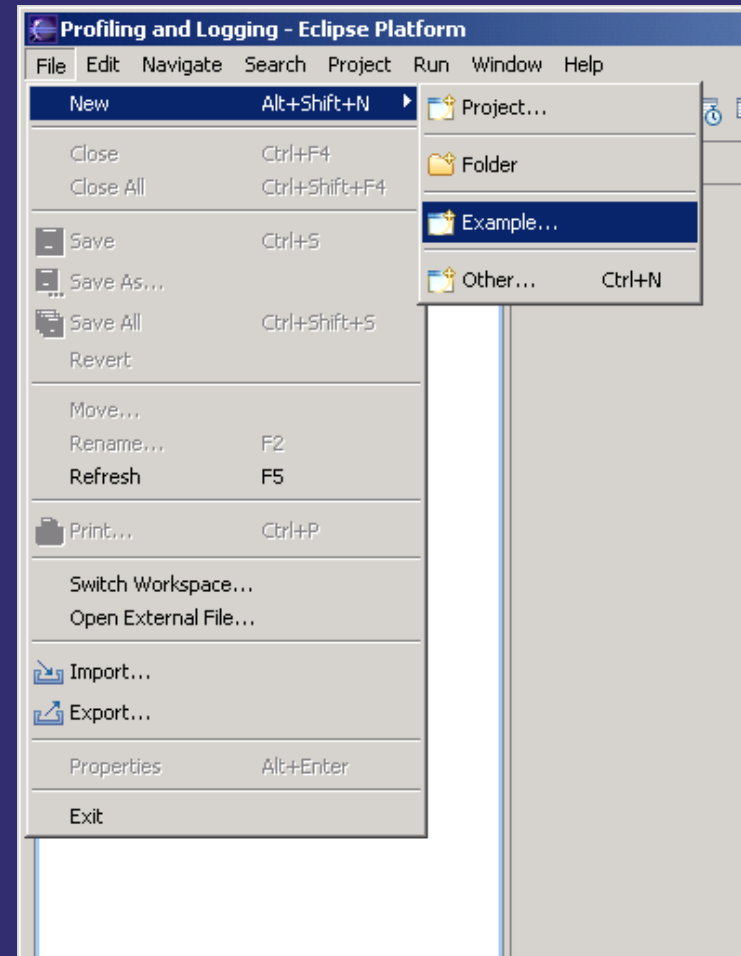
- Extending Eclipse to support deployment testing and functional trace support facilities
- An open source Eclipse Platform and extensible framework for software test and performance tools
- Aims to move software quality practices earlier in to the application development cycle
- Analyze application execution and identify performance problems, such as execution bottlenecks, object leaks, and system resource limitations

# TPTP – Java Profiling

- Trace, Test and Monitoring
- Local and remote hosts
- Agent Controller and Agents
- Launch or Attach
- Profile Filter and Options
- Profiling actions : Start/Stop monitoring, Attach/Detach to agents, GC, Collect Object Reference, Terminate Process
- Profiling Views

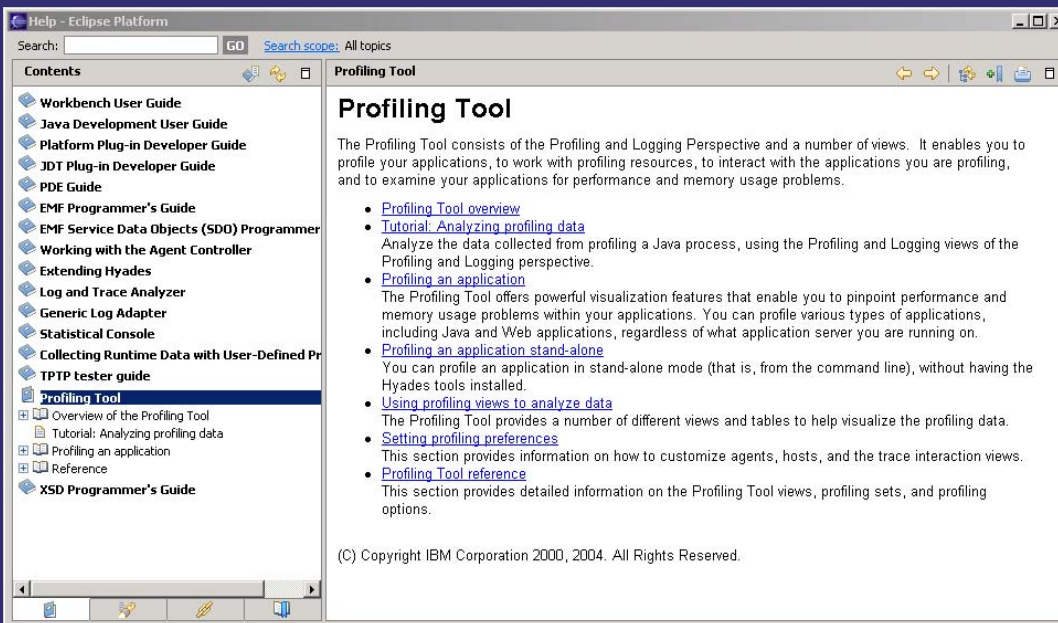
# Examples

- File -> New-> Examples...
- Examples on how to use TPTP, as well as how to extend TPTP



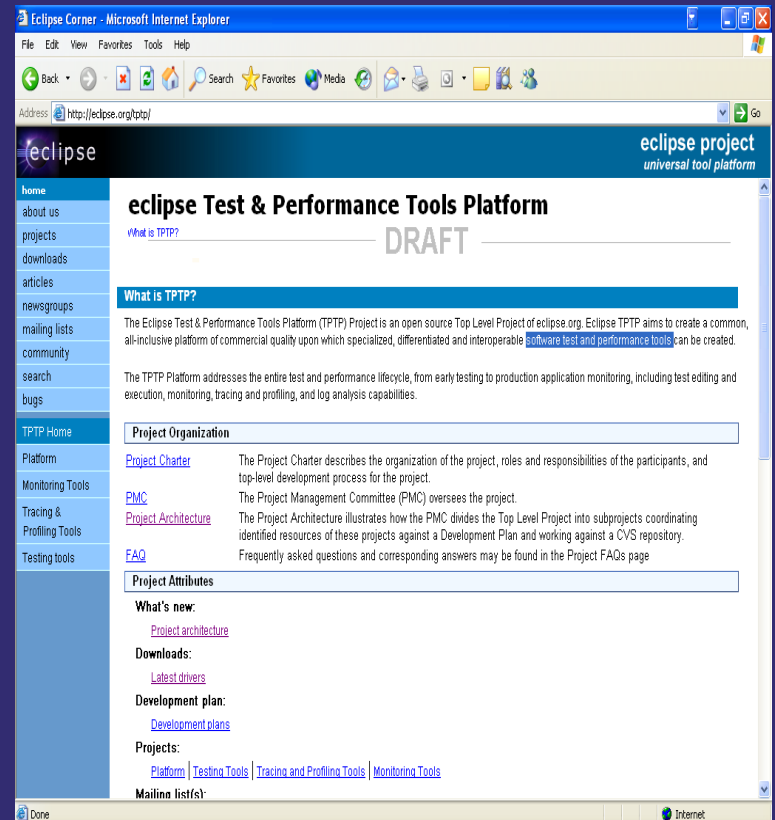
# Need more information?

- TPTP
  - http://www.eclipse.org/tptp/
- TPTP Online Help



# Download TPTP

- Requirements
  - Java runtime (JRE) or Java development kit (JDK) 1.4
  - Eclipse SDK
  - Eclipse Modeling Framework (EMF) SDK
  - XML Schema Infoset Model (XSD) SDK



# Community

- TPTP newsgroup - Questions and discussions about using the project and project-based tools  
<news://news.eclipse.org/eclipse.tptp>
- TPTP Mailing lists
  - PMC communications (including coordination, announcements, and Group discussions)  
[tptp-pmc@eclipse.org](mailto:tptp-pmc@eclipse.org)  
subscribe at <http://dev.eclipse.org/mailman/listinfo/tptp-pmc>
  - Platform Project developer discussions  
[tptp-platform-dev@eclipse.org](mailto:tptp-platform-dev@eclipse.org)  
subscribe at <http://dev.eclipse.org/mailman/listinfo/tptp-platform-dev>
  - Testing Tools Project developer discussions  
[tptp-testing-tools-dev@eclipse.org](mailto:tptp-testing-tools-dev@eclipse.org)  
subscribe at <http://dev.eclipse.org/mailman/listinfo/tptp-testing-tools-dev>
  - Tracing and Profiling Tools Project developer discussions  
[tptp-tracing-profiling-tools-dev@eclipse.org](mailto:tptp-tracing-profiling-tools-dev@eclipse.org)  
subscribe at <http://dev.eclipse.org/mailman/listinfo/tptp-tracing-profiling-tools-dev>

# Bugs

- <https://bugs.eclipse.org/bugs/>
- Product : TPTP Eclipse Test & Performance Tools Platform Project

