



JProbe[®] 8.3

Plugins for Eclipse Guide



© 2009 Quest Software, Inc. ALL RIGHTS RESERVED.

This guide contains proprietary information protected by copyright. The software described in this guide is furnished under a software license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of the applicable agreement. No part of this guide may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of Quest Software, Inc.

If you have any questions regarding your potential use of this material, contact:

Quest Software World Headquarters

LEGAL Dept

5 Polaris Way

Aliso Viejo, CA 92656

www.quest.com

email: legal@quest.com

Refer to our Web site for regional and international office information.

JProbe Patents

Patent pending.

Trademarks

Quest, Quest Software, the Quest Software logo, AccessManager, ActiveRoles, Aelita, Akonix, AppAssure, Benchmark Factory, Big Brother, BusinessInsight, ChangeAuditor, ChangeManager, DeployDirector, DirectoryAnalyzer, DirectoryTroubleshooter, DS Analyzer, DS Expert, ERDisk, Foglight, GPOAdmin, Imceda, IntelliProfile, InTrust, Invirtus, iToken, IWatch, JClass, Jint, JProbe, LeccoTech, LiteSpeed, LiveReorg, LogAdmin, MessageStats, Monosphere, NBSpool, NetBase, NetControl, Npulse, NetPro, PassGo, PerformaSure, Quest Central, Quest vToolkit, Quest vWorkspace, ReportAdmin, RestoreAdmin, SelfServiceAdmin, SharePlex, Sitraka, SmartAlarm, Spotlight, SQL LiteSpeed, SQL Navigator, SQL Watch, SQLab, Stat, StealthCollect, Storage Horizon, Tag and Follow, Toad, T.O.A.D., Toad World, vAutomator, vControl, vConverter, vFoglight, vOptimizer Pro, vPackager, vRanger, vRanger Pro, vSpotlight, vStream, vToad, Vintela, Virtual DBA, VizionCore, Vizioncore vAutomation Suite, Vizioncore vBackup, Vizioncore vEssentials, Vizioncore vMigrator, Vizioncore vReplicator, Vizioncore vTraffic, Vizioncore vWorkflow, WebDefender, Webthority, Xaffire, and XRT are trademarks and registered trademarks of Quest Software, Inc in the United States of America and other countries. Other trademarks and registered trademarks used in this guide are property of their respective owners.

Disclaimer

The information in this document is provided in connection with Quest products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Quest products. EXCEPT AS SET FORTH IN QUEST'S TERMS AND CONDITIONS AS SPECIFIED IN THE LICENSE AGREEMENT FOR THIS PRODUCT, QUEST ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL QUEST BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF QUEST HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Quest makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Quest does not make any commitment to update the information contained in this document.

License Credits and Third Party Information

See *Third_Party_Contributions.htm* in your JProbe \doc installation directory.

Plugins for Eclipse Guide November 2009

Table of Contents

| | |
|--|-----------|
| Introduction to This Guide | 7 |
| About JProbe..... | 8 |
| About This Guide..... | 8 |
| JProbe Documentation Suite..... | 9 |
| Core Documentation Set | 9 |
| Feedback on the Documentation..... | 9 |
| Text Conventions | 10 |
| About Quest Software, Inc. | 10 |
| Contacting Quest Software..... | 11 |
| Contacting Quest Support | 11 |
| Quest Communities | 11 |
| Installing JProbe Plugins for Eclipse..... | 13 |
| System Requirements..... | 14 |
| Installing the JProbe Plugins in Eclipse..... | 14 |
| Installing the Graphical Editing Framework (GEF) | 15 |
| Downloading and Installing JProbe Plugins for Eclipse..... | 16 |
| Launching Eclipse with JProbe Plugins for Eclipse | 18 |
| Installing the JProbe Plugins Using Pulse Explorer | 20 |
| Getting Started with JProbe Plugins for Eclipse | 27 |
| Opening the JProbe Perspective..... | 28 |
| Creating a Project for JProbe Snapshots | 29 |
| Importing Heap Dumps | 31 |
| Importing Snapshots | 33 |
| Creating a Project for the JProbe Demo Code..... | 34 |
| Creating and Managing Categories..... | 36 |

| | |
|---|-----------|
| Adding Categories | 36 |
| Adding Folders | 39 |
| Editing Categories | 39 |
| Renaming Categories or Folders | 40 |
| Deleting Categories or Folders | 40 |
| Managing JProbe Preferences | 40 |
| General JProbe Preferences | 40 |
| CSV Export Preferences | 41 |
| Launch Configuration Preferences | 41 |
| Coverage Preferences | 42 |
| Memory Preferences | 43 |
| Performance Preferences | 44 |
| Managing Licenses | 44 |
| Integrating JProbe into Eclipse | 47 |
| Integrating JProbe to Run Java SE Applications | 48 |
| Configuring a Java Application in Eclipse | 48 |
| Configuring an Eclipse Application in Eclipse | 51 |
| Configuring JUnit in Eclipse | 53 |
| Configuring JUnit Plug-in Test in Eclipse | 53 |
| Configuring an OSGi Application in Eclipse | 53 |
| Launching a Java SE Application with JProbe | 54 |
| Integrating JProbe to Run Java EE Applications | 55 |
| Creating an Application Server | 56 |
| Testing the Application Server | 57 |
| Setting the Server Timeout Delay | 57 |
| Running the Application Server with JProbe | 59 |
| Running a JProbe Analysis in Eclipse | 61 |
| Running a Java Application with JProbe in Eclipse | 62 |
| Launching the Application | 62 |
| Attaching JProbe to the Application | 63 |
| Running a JProbe Session in the Background | 64 |
| Viewing Data in a Runtime Session | 65 |
| Memory Session Runtime View | 65 |

| | |
|--|-----------|
| Performance Session Runtime View | 69 |
| Coverage Session Runtime View | 71 |
| Viewing the Execution Log | 72 |
| Viewing the Session History | 73 |
| Viewing Snapshot Data | 73 |
| Index | 75 |

Introduction to This Guide

This chapter provides information about what is contained in the *JProbe Plugins for Eclipse Guide*. It also provides information about the JProbe documentation suite and Quest Software.

This chapter contains the following sections:

| | |
|--|----|
| About JProbe | 8 |
| About This Guide | 8 |
| JProbe Documentation Suite | 9 |
| Text Conventions | 10 |
| About Quest Software, Inc. | 10 |

About JProbe

JProbe is an enterprise-class Java profiler that provides intelligent diagnostics on memory usage, performance, and test coverage. It allows developers to quickly pinpoint and repair the root cause of application code performance and stability problems that obstruct component and integration integrity.

JProbe provides three types of analysis:

- **Memory analysis**—allows a developer to identify and resolve Java memory leaks and object cycling, to ensure optimal program efficiency and stability.
- **Performance analysis**—allows a developer to identify and resolve Java bottlenecks and deadlocks, to ensure optimal program performance and scalability.
- **Coverage analysis**—allows a developer to identify un-executed lines of code during unit testing, to ensure test coverage and program correctness.

JProbe also offers an Eclipse plugin that provides intelligent code performance analysis and problem resolution from within the Eclipse Java IDE.

About This Guide

The *JProbe Plugins for Eclipse Guide* presents the process of installing JProbe Plugins for Eclipse and setting up your JProbe environment within Eclipse.

This document is intended for Eclipse users who want to analyze the memory usage and performance of their Java code or find out how well their test cases cover their code.

The *JProbe Plugins for Eclipse Guide* is organized as follows:

Chapter 1, Installing JProbe Plugins for Eclipse—describes how to install JProbe Plugins for Eclipse and how to launch Eclipse with the plugins.

Chapter 2, Getting Started with JProbe Plugins for Eclipse—presents how to set up a JProbe project and import snapshot files.

Chapter 3, Integrating JProbe into Eclipse—describes how to integrate JProbe into an Eclipse environment.

Chapter 4, Running a JProbe Analysis in Eclipse—describes how to run your Java application with JProbe in Eclipse, which allows you to identify problems with your application and investigate those problems.

JProbe Documentation Suite

The JProbe documentation suite is provided in a combination of online help, PDF, HTML, and TXT.

- **Online Help:** You can open the online help by clicking the **Help** icon on the JProbe toolbar.
- **PDF:** The complete JProbe documentation set is available in PDF format on SupportLink. The PDF documentation can also be found in the Documentation folder on the JProbe DVD. The default location of the documentation after an installation is `<jprobe_home>/docs`. Adobe Reader is required.
- **HTML:** *Release Notes* are provided in HTML and TXT format. The default location of this document after an installation is `<jprobe_home>/docs`.

The *Ant Tasks User Manual* is also provided in HTML format. The default location of this document after an installation is `<jprobe_home>/automation/doc`. To open it, navigate to `index.html`.

Core Documentation Set

The core documentation set consists of the following files:

- *JProbe Installation Guide* (PDF)
- *JProbe User Guide* (PDF and online help)
- *JProbe Reference Guide* (PDF)
- *JProbe Plugins for Eclipse Guide* (PDF)
- *JProbe Tutorials* (PDF and online help)
- *JProbe Release Notes* (HTML and TXT)
- *Ant Tasks User Manual* (HTML)

Feedback on the Documentation

We are interested in receiving feedback from you about our documentation. For example, did you notice any errors in the documentation? Were any features undocumented? Do you have any suggestions on how we can improve the documentation? All comments are welcome. Please submit your feedback to the following email address:

am.docfeedback@quest.com

Please do not submit Technical Support related issues to this email address.

Text Conventions

The following table summarizes how text styles are used in this guide:

| Convention | Description |
|---|--|
| Code | Monospace text represents code, code objects, and command-line input. This includes: <ul style="list-style-type: none">• Java language source code and examples of file contents• Classes, objects, methods, properties, constants, and events• HTML documents, tags, and attributes |
| <i>Variables</i> | Monospace-plus-italic text represents variable code or command-line objects that are replaced by an actual value or parameter. |
| Interface | Bold text is used for interface options that you select (such as menu items) as well as keyboard commands. |
| <i>Files, components, and documents</i> | Italic text is used to highlight the following items: <ul style="list-style-type: none">• Pathnames, file names, and programs• The names of other documents referenced in this guide |

About Quest Software, Inc.

Now more than ever, organizations need to work smart and improve efficiency. Quest Software creates and supports smart systems management products—helping our customers solve everyday IT challenges faster and easier. Visit www.quest.com for more information.

Contacting Quest Software

| | |
|----------|---|
| Email | info@quest.com |
| Mail | Quest Software, Inc. World Headquarters 5 Polaris Way Aliso Viejo, CA 92656 USA |
| Web site | www.quest.com |

Refer to our web site for regional and international office information.

Contacting Quest Support

Quest Support is available to customers who have a trial version of a Quest product or who have purchased a commercial version and have a valid maintenance contract. Quest Support provides around the clock coverage with SupportLink, our web self-service. Visit SupportLink at: <http://support.quest.com>.

From SupportLink, you can do the following:

- Quickly find thousands of solutions (Knowledgebase articles/documents).
- Download patches and upgrades.
- Seek help from a Support engineer.
- Log and update your case, and check its status.

View the *Global Support Guide* for a detailed explanation of support programs, online services, contact information, and policy and procedures. The guide is available at: [http://support.quest.com/pdfs/Global Support Guide.pdf](http://support.quest.com/pdfs/Global%20Support%20Guide.pdf).

Quest Communities

Get the latest product information, find helpful resources, and join a discussion with the JProbe Quest team and other community members. Join the JProbe community at: <http://jprobe.inside.quest.com/>.

Installing JProbe Plugins for Eclipse

This chapter describes how to install JProbe Plugins for Eclipse and how to launch Eclipse with the plugins.

This chapter contains the following sections:

| | |
|--|----|
| System Requirements | 14 |
| Installing the JProbe Plugins in Eclipse | 14 |
| Installing the JProbe Plugins Using Pulse Explorer | 20 |

System Requirements

JProbe Plugins for Eclipse require the following environment:

- Eclipse 3.3.2, 3.4.x, or 3.5
- Sun JRE 6 (or any update to JRE 6)
- Graphical Editing Framework (GEF) with the same version number as Eclipse
 - Note** If you are using Pulse Explorer to manage multiple Eclipse software configurations, the GEF is automatically installed together with the Eclipse version you select.
- Quest JProbe 8.3

If you plan to run a Java EE application with JProbe in Eclipse, you also need to install the following versions of the WTP (Web Tools Platform) framework on your system:

- For Eclipse Classic 3.3.2: WTP complete 2.0.2.
- For Eclipse Classic 3.4.x: WTP 3.0.1.
- For Eclipse IDE for Java EE Developers: you do not need to install an additional WTP framework; it is already integrated with this Eclipse version.
- For Eclipse 3.5: WTP 3.1.

You can obtain the WTP files from <http://download.eclipse.org/webtools/downloads/>. For additional information, see “[Integrating JProbe to Run Java EE Applications](#)” on page 55.

Installing the JProbe Plugins in Eclipse

This section presents the procedures that a generic Eclipse user must follow in order to download and install the JProbe Plugins for Eclipse in his Eclipse environment. It also presents different methods of launching Eclipse with the newly installed plugins.

Important If you are using Pulse Explorer to manage multiple Eclipse software configurations, it is recommended that you install the JProbe Plugins for Eclipse as presented in section “[Installing the JProbe Plugins Using Pulse Explorer](#)” on page 20.

To install the JProbe Plugins for Eclipse in your Eclipse environment:

Note This procedure assumes that you have already installed an Eclipse environment and the Sun JRE on your machine.

- 1 Install the appropriate GEF version, as specified in “[System Requirements](#)” on page 14. For detailed instructions, see “[Installing the Graphical Editing Framework \(GEF\)](#)” on page 15.
- 2 If you plan to run a Java EE application with JProbe in Eclipse, install the appropriate version of the WTP framework, as specified in “[System Requirements](#)” on page 14.
- 3 Download and install the JProbe Plugins for Eclipse. For detailed instructions, see “[Downloading and Installing JProbe Plugins for Eclipse](#)” on page 16.
- 4 To use JProbe Plugins for Eclipse, launch Eclipse with the appropriate command line options. For detailed instructions, see “[Launching Eclipse with JProbe Plugins for Eclipse](#)” on page 18.

Installing the Graphical Editing Framework (GEF)

The GEF plugins are required by components that are used within the JProbe views. You may need to download the framework; it does not come with the basic Eclipse 3.3 distribution. Choose the version that matches the version number of your Eclipse install. For example, if you installed Eclipse 3.3.2, select GEF 3.3.2.

To obtain and install the GEF plugins:

- 1 Go to the Downloads page on the Eclipse Web site:
<http://download.eclipse.org/tools/gef/downloads/>
- 2 Locate the version of GEF that matches the version of your Eclipse installation. You may need to scroll down to the Archived Releases section and select the version link.
- 3 Select the link to download the GEF SDK. The file name of the zip file should have the form *GEF-SDK-#.##.zip*.
- 4 Extract the zip file to the parent directory of your Eclipse installation directory. For example, if Eclipse is installed in *C:\eclipse*, extract the zip file to *C:*.

Downloading and Installing JProbe Plugins for Eclipse

You can download and install the JProbe Plugins for Eclipse files via any of the following two methods:

- Using the latest *jprobe-eclipse-plugins.zip* file provided by Quest. For details, see “[Installing the JProbe Plugins for Eclipse from the ZIP File](#)” on page 16.
- Using the Update Manager wizard. For details, see “[Downloading and Installing Plugins Using the Eclipse Update Manager](#)” on page 16.

Installing the JProbe Plugins for Eclipse from the ZIP File

To install JProbe Plugins for Eclipse from the ZIP file:

- 1 Obtain the *jprobe-eclipse-plugins.zip* file containing the latest JProbe plugins for Eclipse.
- 2 Do one of the following:
 - If your Eclipse installation directory contains a *dropins* sub-directory, extract the zip file in that location. For example, if Eclipse is installed in *C:\eclipse*, extract the zip file to *C:\eclipse\dropins*.
 - If your Eclipse installation directory does not contain a *dropins* sub-directory, or if you have had problems with the Eclipse Update Manager in the past, then extract the zip file to the parent directory of your Eclipse installation directory. For example, if Eclipse is installed in *C:\eclipse*, extract the zip file to *C:*.

Downloading and Installing Plugins Using the Eclipse Update Manager

This procedure provides a basic overview of using the Eclipse Update Manager to download and install the JProbe Plugins for Eclipse feature. For more information about managing features, see the Eclipse online help. Click **Help > Help Contents** and select **Workbench User Guide > Tasks > Updating features with the update manager**.

Note This procedure applies only to Eclipse 3.5 environments, for Windows and Linux.

To install the plugins using the Eclipse Update Manager:

- 1 Launch Eclipse.
- 2 Click **Help > Software Updates**.
- 3 In the **Available Software** tab, click **Add Site**.
- 4 In the Add Site dialog box, specify the following site information:

- Name—for example, *JProbe*
- Location—for Eclipse 3.5, type in the following URL: <http://www.quest.com/jprobe/Eclipseupdate/35>.

5 Click **OK**.

The Quest JProbe site is added to the list of sites. Eclipse contacts the update site and populates the Quest JProbe entry with the feature that is available from the site.

6 Select the check box corresponding to the Quest JProbe update site, expand the tree to see that the feature is also selected, then click **Install**.

7 In the Install wizard, review the installation settings, and click **Next**.

8 Review and accept the license agreements, and click **Finish**.

Eclipse installs the required software and, upon completion, prompts you to restart your environment. Some versions of Eclipse may not correctly set up the JProbe settings in *eclipse.ini*. Therefore, you need to modify this file before restarting Eclipse.

9 Click **No**, then close Eclipse.

To use JProbe Plugins for Eclipse, you need to launch Eclipse with some command line options. For more information, see “[Launching Eclipse with JProbe Plugins for Eclipse](#)” on page 18.

Launching Eclipse with JProbe Plugins for Eclipse

You can launch JProbe Plugins for Eclipse either by editing the *eclipse.ini* file or from the command line. Regardless of the method, you need to launch Eclipse with the following options:

| | |
|----------------------------|---|
| <code>-vm</code> | Set to Sun JRE 6 (or any update to JRE 6). |
| <code>-vmargs</code> | Set the heap size. |
| <code>-Xms256m</code> | |
| <code>-Xmx1024m</code> | |
| <code>-Djprobe.home</code> | Set to your JProbe installation directory. JProbe plugins use this setting to locate the JProbe Analysis Engine and to access JProbe licensing. |
| | Note This setting is not required if you installed JProbe via the Eclipse 3.5 Update Manager. |

Tip On Windows, you can create (or edit) an Eclipse shortcut to include these options. Create a shortcut as usual and name it something like “Launch Eclipse with JProbe”. Right-click the shortcut and select **Properties**. Edit the Target field to include the command line options.

Launching Eclipse by Editing the INI File

The options in the `<ECLIPSE_HOME>/eclipse.ini` text file control the Eclipse startup. This file contains command-line options that are added to the command line when Eclipse is started. If `ECLIPSE_HOME` is not defined, the default *eclipse.ini* file in the Eclipse installation directory is used.

Note the following:

- Each option and each argument to an option must be on its own line.
- All lines after `-vmargs` are passed as arguments to the Java VM, so all arguments and options for Eclipse must be specified before `-vmargs`.

To launch Eclipse with the JProbe plugins by editing the *.ini* file:

- 1 Open the *eclipse.ini* file in a text editor.
- 2 Add or update the following options and arguments:

`-vm`

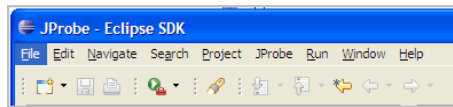
```
C:\Program Files\<java_home>\bin\javaw
-vmargs
-Xms256m
-Xmx1024m
-Djprobe.home=<jprobe_home>
```

Note This setting is not required if you installed JProbe via the Eclipse 3.5 Update Manager.

- 3 Save the file and start Eclipse as usual.

A JProbe menu item appears on the main menu.

Tip If the JProbe menu does not appear on the menu bar, use the `-clean` command line option described in “[Launching Eclipse from the Command Line](#)” on page 19.



You can now get started with using the JProbe Plugins in Eclipse. For a quick overview, click **JProbe > Quick Start Page** on the menu bar.

Launching Eclipse from the Command Line

To launch Eclipse with the JProbe plugins from the command line:

- 1 From the command line, enter the following commands all on one line:

```
c:\eclipse\eclipse -vm <path_to_javaw> -vmargs
-Xms256m -Xmx1024m -Djprobe.home=<path_to_JProbe_dir>
```

Note Anything after `-vmargs` is passed to the Java VM rather than Eclipse.

For example:

```
c:\eclipse\eclipse
-vm "c:\Program Files\java\jdk1.6.0\bin\javaw.exe"
-vmargs -Xms256m -Xmx1024m -Djprobe.home="c:\Program
Files\JProbe 8.3"
```

When Eclipse launches, there is a JProbe menu item on the main menu.

- 2 If you do not see the JProbe menu item, use the `-clean` command line option.
 - a Exit Eclipse.
 - b Edit the command line for Eclipse to include the `-clean` option.

For example:

```
c:\eclipse\eclipse -clean
-vm "c:\Program files\java\jdk1.6.0\bin\javaw.exe"
-vmargs -Xms256m -Xmx1024m -Djprobe.home="c:\Program
Files\JProbe 8.3"
```

c Launch Eclipse again.

You can now get started with using the JProbe Plugins in Eclipse. For a quick overview, click **JProbe > Quick Start Page** on the menu bar.

Installing the JProbe Plugins Using Pulse Explorer

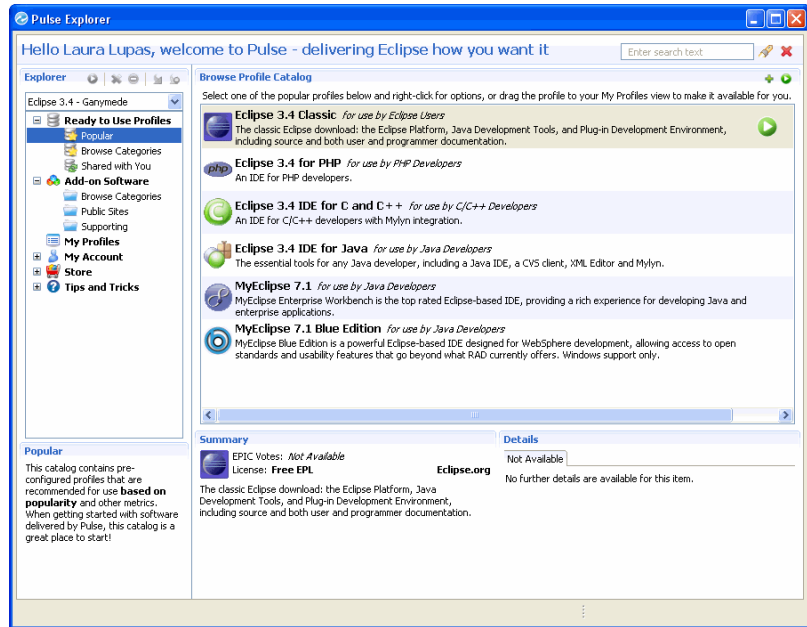
This section presents how to install the JProbe Plugins for Eclipse using the Pulse Explorer.

To install the JProbe Plugins for Eclipse using the Pulse Explorer:

Note This procedure assumes that you have already installed the Pulse Explorer to manage multiple Eclipse configurations on your machine. You can download and install the Pulse software from the Genuitec Web site: www.poweredbypulse.com.

1 Start the Pulse Explorer application.

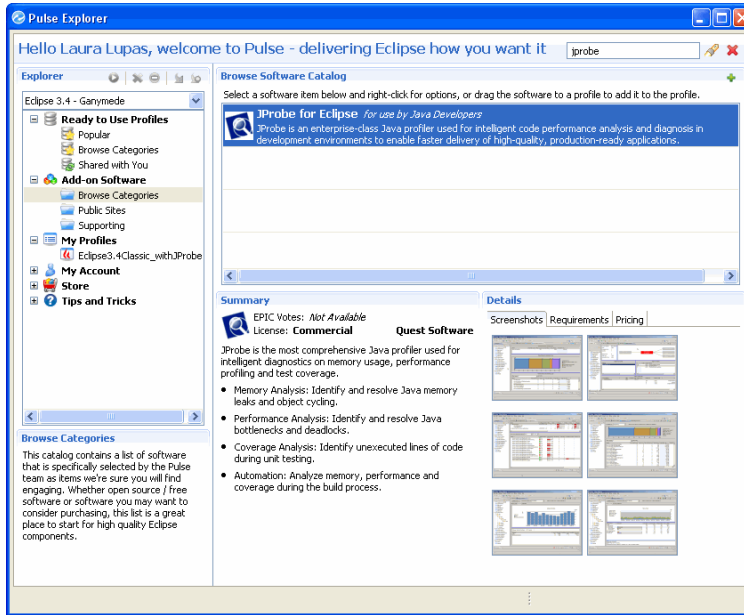
The Pulse Explorer graphical user interface appears, showing the popular profiles available for use on your machine. This information is also displayed by clicking **Ready to Use Profiles > Popular** in the Explorer pane.



- 2 Create a new profile for the Eclipse environment in which you are planning to integrate the JProbe Plugins for Eclipse. If you already have a profile, skip this step and continue with [step 3](#).
 - a In the Browse Profile Catalog pane, select the desired Eclipse version.
 - b Drag and drop it over the **My Profiles** in the Explorer pane.
 - c In the Add to My Profiles dialog box, specify a name for the new profile you are creating, then click **OK**.

The new profile's name appears in the Explorer pane under **My Profiles**.
- 3 Add the JProbe Plugins for Eclipse to your profile.
 - a In the Explorer pane, click **Add-on Software > Browse Categories**.
 - b Type `jprobe` in the search field located in the upper right corner of the Pulse Explorer interface, and press the **Enter** key.

Pulse locates the latest version of the JProbe Plugins for Eclipse on the Quest Web site and displays this software on the Browse Software Catalog pane.



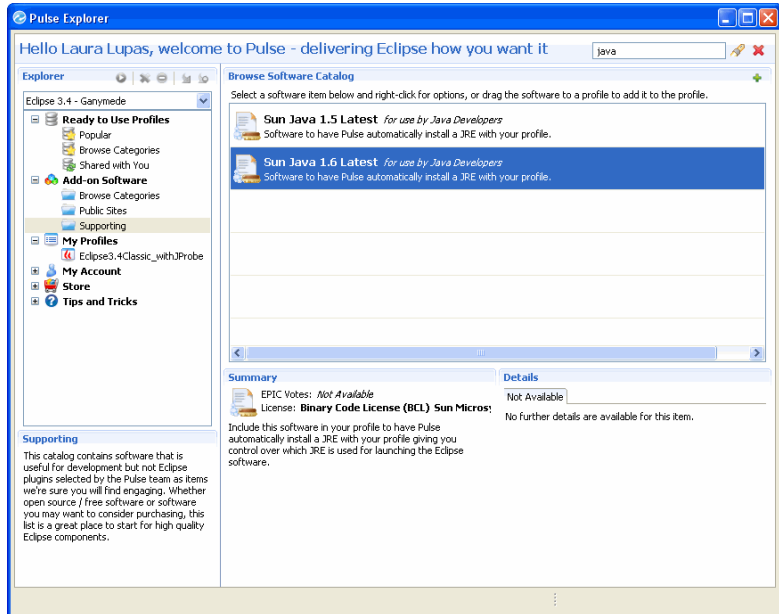
- c Add the plugins to your profile by dragging and dropping the JProbe for Eclipse option over your profile in the Explorer pane.
 - d Click **OK** in the confirmation dialog box that appears at the end of the operation.
- 4 Complete your profile by adding a JRE.

Note Sun Java 1.6 (or later) is required.

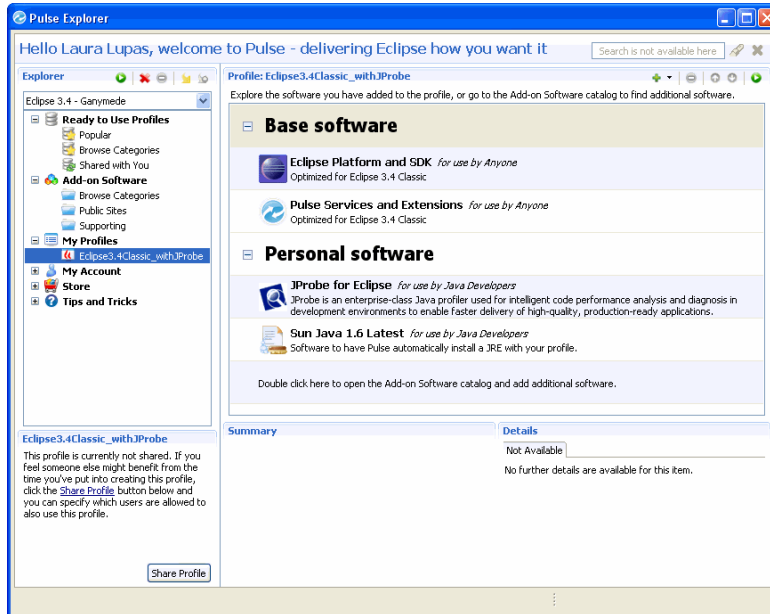
- a In the Explorer pane, click **Add-on Software > Browse Categories**.
- b Type `java` in the search field located in the upper right corner of the Pulse Explorer interface, and press the `Enter` key.

Pulse locates the JRE versions available for your selection and displays them on the Browse Software Catalog pane.


- c Select the Sun Java 1.6 (or later) from the list, as recommended in “[System Requirements](#)” on page 14.



- d Add the JRE to your profile by dragging and dropping the Sun Java 1.6 Latest option over your profile in the Explorer pane.
 - e Click **OK** in the confirmation dialog box that appears at the end of the operation.
- 5 Review your profile, by clicking its name in the Explorer pane.
- The Profile pane displays the base and personal software added to your profile.



- 6 Right-click on the profile name in the Explorer pane and select **Edit Launch Arguments**.
- 7 Add the following arguments in the VM arguments text box:


```
-Xms256m -Xmx1024m
```
- 8 Install your profile then run it as a new instance.
 - a Click the  icon on the Profile pane's toolbar.

Pulse validates your profile in the background.
 - b In the Accept Software Licenses dialog box, select the *I acknowledge and accept ALL of the above licenses* check box, then click **Next**.
 - c Optional: select a location different than the default one for installing the software in your profile. Click **Configure**, use the **Browse** buttons to navigate to the new locations, then click **Next**.
 - d Click **Install**.

Pulse downloads from the Web the selected software and installs them in the specified location. When the operation is complete, Pulse launches Eclipse.

- e Select a location for the Eclipse workspace, then click **OK**.

The Eclipse user interface appears. The JProbe menu showing on the menu bar indicates that the JProbe Plugins for Eclipse are installed and ready to use.

Getting Started with JProbe Plugins for Eclipse

When Eclipse is launched with the JProbe plugins, a JProbe perspective is created. You work with JProbe snapshots from within this perspective.

This chapter presents how to set up a JProbe project and import snapshot files.

This chapter contains the following sections:

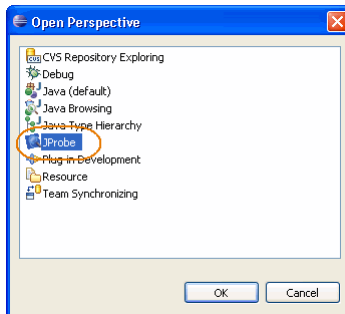
| | |
|---|----|
| Opening the JProbe Perspective | 28 |
| Creating a Project for JProbe Snapshots | 29 |
| Importing Heap Dumps | 31 |
| Importing Snapshots | 33 |
| Creating a Project for the JProbe Demo Code | 34 |
| Creating and Managing Categories | 36 |
| Managing Licenses | 44 |

Opening the JProbe Perspective

For information about perspectives, consult the Eclipse online help.

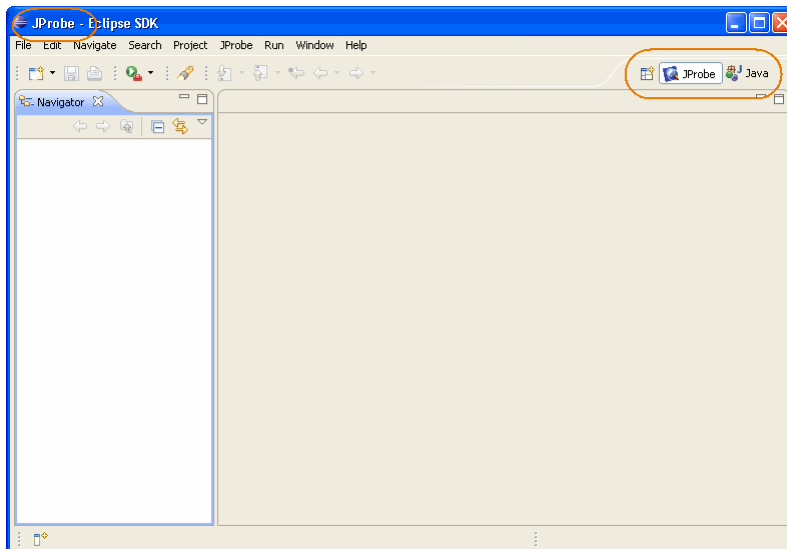
To open the JProbe perspective:

- 1 Select **Window > Open Perspective > Other**.
- 2 Select **JProbe**.







- 3 Click **OK**.

JProbe is displayed in the title bar and on the perspective shortcut bar.



To return to the JProbe perspective after navigating away from it:

- If JProbe is on the shortcut bar, click    **JProbe**.
- Otherwise, click  **Open Perspective** and select **Other** > **JProbe**, then click **OK**.

Creating a Project for JProbe Snapshots

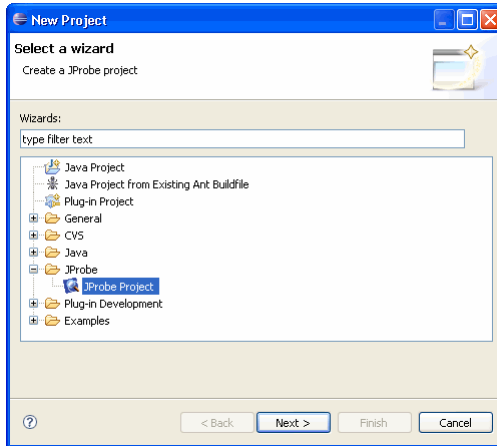
You may want to create one project for all snapshots, or multiple projects for different types of snapshots. Use one of the following methods, depending on your project needs:

- [To create a project for snapshots using the JProbe wizard:](#)
- [To create a project for snapshots from the Attach to Running JProbe Session dialog box:](#)

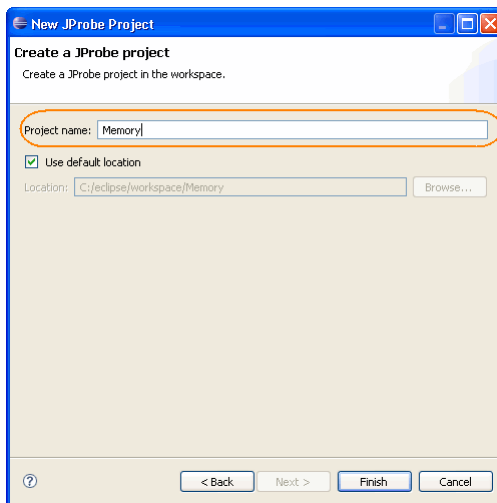
To create a project for snapshots using the JProbe wizard:

- 1 In the JProbe perspective, select **File** > **New** > **Project**.
- 2 Select **JProbe** > **JProbe Project**.

Note If the JProbe perspective and menu are visible, but the JProbe project type is not available, check the **JProbe** menu. If it contains only the **Quick Start Page** item, please check the JRE version you used to launch Eclipse. You must use a Java 1.6 JRE to run the JProbe plugins for Eclipse. For detailed installation instructions, see the *JProbe Installation Guide*. The same situation occurs if the dependencies for the JProbe plugins are not met (for example, if third-party plugins, like GEF, are not installed). For a detailed list of system requirements, see “[System Requirements](#)” on page 14.



- 3 Click **Next**.
- 4 Provide a name for the project, such as **Memory**.



- 5 Provide a location for the project or accept the default location.
- 6 Click **Finish**.

The project name appears in the Navigator.

To create a project for snapshots from the Attach to Running JProbe Session dialog box:

- 1 In the Attach to Running JProbe Session dialog box, click **New JProbe Project**.
- 2 Provide a name for the project, such as Memory.
- 3 Provide a location for the project or accept the default location.
- 4 Click **Finish**.

The project name appears in the Navigator.

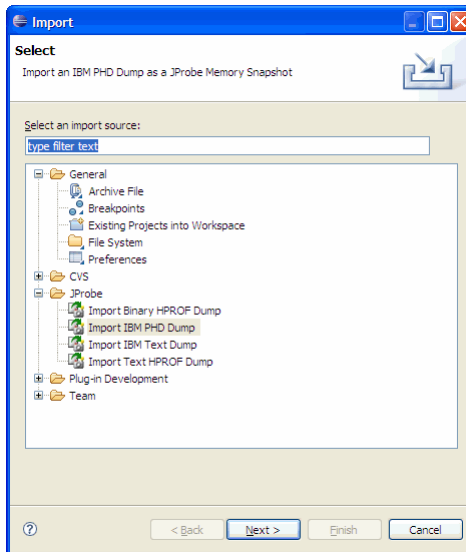
Importing Heap Dumps

The JProbe project is the default folder into which you import heap dumps. You may want to create a separate JProbe project for heap dumps.

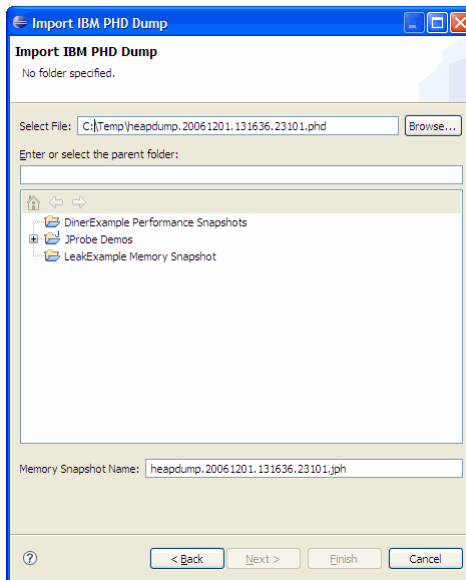
To import a heap dump:

- 1 In the Navigator, select the JProbe project that you created.
- 2 Click **File > Import**.
- 3 Expand the **JProbe** folder.

Note If the JProbe perspective and menu are visible, but the JProbe project type is not available, check the **JProbe** menu. If it contains only the **Quick Start Page** item, please check the JRE version you used to launch Eclipse. You must use a Java 1.6 JRE to run the JProbe plugins for Eclipse. For detailed installation instructions, see the *JProbe Installation Guide*. The same situation occurs if the dependencies for the JProbe plugins are not met (for example, if third-party plugins, like GEF, are not installed). For a detailed list of system requirements, see "[System Requirements](#)" on page 14.



- 4 Select the type of heap dump you want to import and click **Next**.



- 5 Click **Browse** and navigate to the heap dump file that you want to open.

- 6 Click **Open**, then **Finish**.

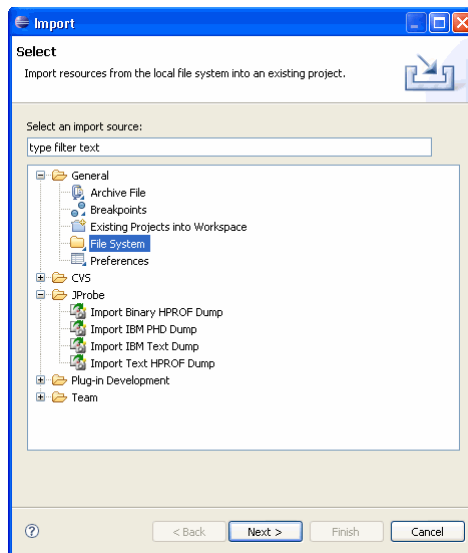
The heap dump file is converted to a JProbe snapshot and listed in the Navigator.

Importing Snapshots

You may want to import JProbe snapshots from the standalone JProbe console, from JProbe's automation tools, or from another developer who is using JProbe. The JProbe project is the default folder into which you import snapshots. You may want to create a separate JProbe project for imported snapshots.

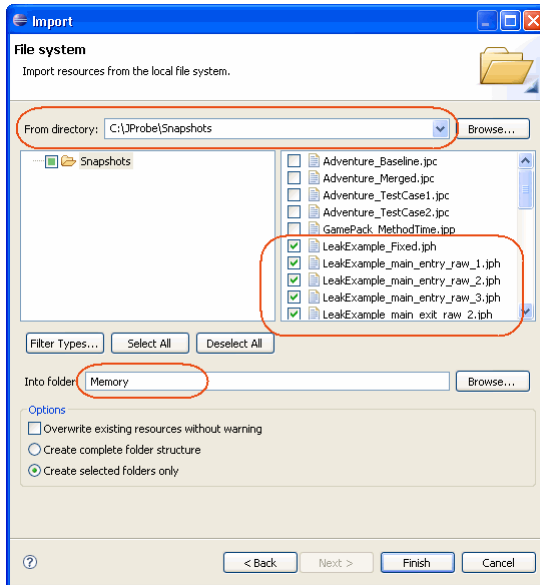
To import snapshots:

- 1 In the Navigator, select the JProbe project that you created.
- 2 Click **File > Import**.
- 3 Select **General > File System**.



- 4 Click **Next**.
- 5 Click the **Browse** button beside the From Directory text field and browse to the directory containing the snapshot files.
- 6 Select the snapshots to import.

- All snapshots—select the check box beside the directory name or click **Select All**.
 - Some snapshots—select the check boxes beside the snapshots.
- 7 By default, snapshots are imported to the currently selected JProbe project. To change the project, specify the project name in the **Into folder** text box.



- 8 Click **Finish**.

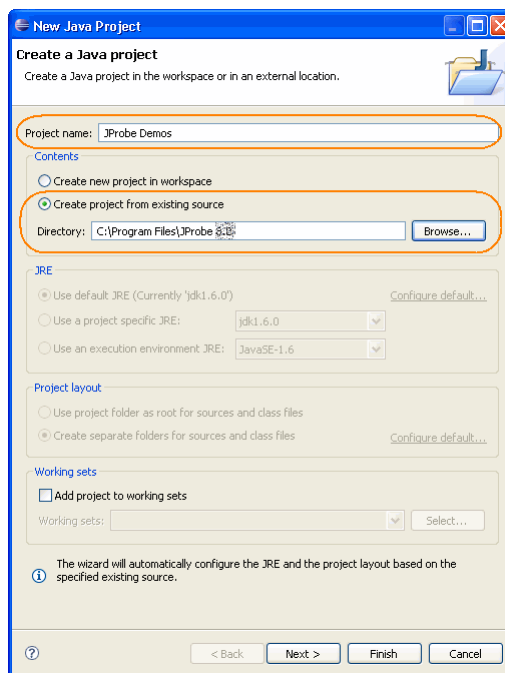
The snapshots are listed under the project in the Navigator. For information on viewing snapshot data, see “[Viewing Snapshot Data](#)” on page 73.

Creating a Project for the JProbe Demo Code

A set of JProbe tutorials and demo applications are available via the JProbe community (<http://jprobe.inside.quest.com/>). If you want to view in Eclipse the snapshots that you took while working through a tutorial, you should create a project in Eclipse for the demo code, so that Eclipse can find the demo code and display source code in the Source view.

To create a project for demo code:

- 1 Select **File > New > Project**.
- 2 Select **Java Project** and click **Next**.
- 3 Provide a name for the project, such as **JProbe Demos**.
- 4 Under Contents, select **Create project from existing source**.
- 5 From the Directory field, browse to your JProbe installation directory; for example, *C:\Program Files\JProbe 8.3*.

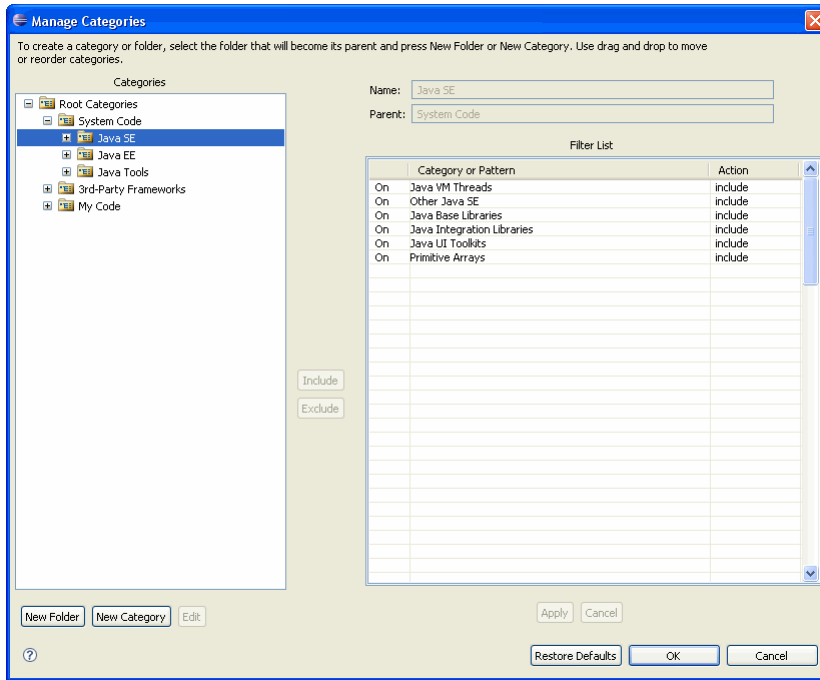


- 6 Click **Finish**.
- 7 You may be prompted to switch to the Java perspective. If so, click **No**.
Note If you disabled the prompt, you may end up in the Java perspective. If so, switch back to the JProbe perspective

The project name appears in the Navigator. The code is under demos.

Creating and Managing Categories

Categories provide a powerful tool for organizing the data that JProbe collects about your application. JProbe provides some preset categories, such as Java SE and Java EE, but you can also create your own. Categories are used in the Memory analysis tool to group instances.



Adding Categories

You add code to categories using filters. For information about filters, see “[Syntax for Filters](#)” on page 37.

To add a category:

- 1 From the main menu, select **JProbe > Manage Categories**.
- 2 Select the folder in which the category logically belongs and click **New Category**.

- 3 Enter a name in the Name field.
- 4 To add packages to this category, double-click the cell and enter the package name.
The package is displayed in the cell and the Action is set to Include.
- 5 To exclude some code from this category, enter the package, class, or method (no signature) in the cell and set the Action to **Exclude**.
For details about filters, see “[Syntax for Filters](#)” on page 37.
- 6 To add existing folders or categories to this category, select the folder or category and click the **Include** button.
Note **Order is important.** Filters lower in the table override filters defined higher in the table. If you need to rearrange the filters, click and hold the row button for the filter you want to move and drag it to its new location. Right-click a row button to insert or delete a row.
- 7 Click **Apply**.
- 8 Repeat to create additional categories.
- 9 Click **OK**.

Syntax for Filters

Filters are case-sensitive and blank spaces are not allowed. You can use an asterisk (*) as a wild card; it matches any character, including the package separator. If you specify the code in a shorthand or partial form, JProbe automatically changes it to canonical form. The change may not be exactly what you desire, so be sure to verify it. JProbe does not check to see whether the filter actually matches anything in your code. For example, if you make a typing error in your package name, nothing in your code will match the filter and so the filter is not effective.

The following table contains common syntax and describes how JProbe interprets it:

| If you type this: | Canonical form is this: | And filters are applied to: |
|-------------------|-------------------------|--|
| * | *.*.*() | All methods in all classes in all packages (including the unnamed package) |
| .* | .*.*() | All methods in all classes in the unnamed package only |

| If you type this: | Canonical form is this: | And filters are applied to: |
|--------------------------|----------------------------|--|
| <code>C</code> | <code>*.C.*()</code> | All methods in classes named <code>C</code> in any package |
| <code>.C</code> | <code>.C.*()</code> | All methods in the class named <code>C</code> in the unnamed package only |
| <code>M()</code> | <code>*.*.M()</code> | Method <code>M</code> in all classes in all packages |
| <code>P.String*</code> | <code>P.String*.*()</code> | All methods in any class in the package <code>P</code> whose name starts with <code>String</code> and all methods in any class in a package or subpackage of <code>P.String*</code> (matches both <code>P.StringArt.draw()</code> and <code>P.Stringent.Check.English.spelling()</code>) |
| <code>P.*.C.do*()</code> | <i>no change</i> | Any method whose name starts with <code>do</code> in any class named <code>C</code> in any subpackage of <code>P</code> , except <code>P</code> directly (matches <code>P.SubPkg.C.doGet()</code> , but does not match <code>P.C.doNotMatch()</code>) |
| <code>Pre*</code> | <code>*.Pre*.*()</code> | All methods in all classes starting with <code>Pre</code> in any package, as well as any class in a subpackage whose name starts with <code>Pre</code> (matches both <code>Pkg.Prepare.m()</code> and <code>Pkg.Pretend.C.m()</code>) |
| <code>s*y</code> | <code>*.s*y.*()</code> | All methods in classes whose name ends with <code>y</code> and either starts with <code>s</code> or belongs to a subpackage that starts with <code>s</code> (matches both <code>java.rmi.server.RMIsocketFactory</code> , <code>com.quest.say</code> , and <code>java.security.cert.X509CRLentry</code>) |

| If you type this: | Canonical form is this: | And filters are applied to: |
|--|----------------------------|--|
| <code>foo.bar</code> | <code>foo.bar.*()</code> | All methods in the class <code>bar</code> in package <code>foo</code> . This is likely not what you wanted. See the next example. |
| <code>foo.bar.*</code> | <code>foo.bar.*.*()</code> | All methods in all classes in <code>foo.bar</code> or any of its subpackages |
| <code>foo.bar. String.t*e()</code> | <i>no change</i> | Methods whose names start with <code>t</code> and end in <code>e</code> in <code>foo.bar.String</code> (matches both <code>toUpperCase()</code> and <code>toLowerCase()</code>) |

Adding Folders

To add a folder:

- 1 From the main menu, choose **JProbe > Manage Categories**.
- 2 Right-click a folder and choose **New Folder**.
- 3 Enter a name in the Name field.
- 4 Click **Apply**.
- 5 Repeat to create additional folders.
- 6 Click **OK**.

Editing Categories

To edit a category:

- 1 From the main menu, choose **JProbe > Manage Categories**.
- 2 Select a category and click **Edit**.
- 3 Make changes to the filter list.
Note To quickly remove categories or patterns from the list, select the item and press the **Delete** key.
- 4 Click **Apply**.
- 5 Click **OK**.

Renaming Categories or Folders

To rename a category or folder:

- 1 From the main menu, select **JProbe > Manage Categories**.
- 2 Right-click a category and choose **Rename**.
- 3 Specify the new name and click **OK**.
- 4 Click **OK**.

Deleting Categories or Folders

To delete a category or folder:

- 1 From the main menu, select **JProbe > Manage Categories**.
- 2 Right-click the category or folder and choose **Delete**.

Managing JProbe Preferences

This section presents how to set preferences that affect how JProbe operates and displays data. For detailed information, refer to the following categories:

- [General JProbe Preferences](#)
- [CSV Export Preferences](#)
- [Launch Configuration Preferences](#)
- [Coverage Preferences](#)
- [Memory Preferences](#)
- [Performance Preferences](#)

General JProbe Preferences

To set the General JProbe Preferences:

- 1 On the Eclipse menu bar, click **Window > Preferences**.
- 2 In the Preferences dialog box, click **JProbe**.

The general preferences are displayed in the right pane:

- Show Package Names—by default, wherever a class name appears in a view, the package name is prepended to it.
 - Show Method Signatures—by default, when a method is identified, the method signature is appended to it.
 - Prompt me whenever I am about to compare snapshots from different applications—by default, this option is enabled.
- 3 To change any of these default operating settings, clear its corresponding check box, then click **Apply**.

CSV Export Preferences

To select whether or not headers are included in the exported data:

- 1 On the Eclipse menu bar, click **Window > Preferences**.
- 2 In the Preferences dialog box, click **JProbe > CSV Export**.

The CSV settings are displayed in the right pane:

- Export Column Headings—by default, this option is enabled.
- 3 To change this default setting, clear its corresponding check box, then click **Apply**.

Launch Configuration Preferences

To set the preferences for launching the JProbe configuration files:

- 1 On the Eclipse menu bar, click **Window > Preferences**.
- 2 In the Preferences dialog box, click **JProbe > Launch Configuration**.

The launch configuration preferences are displayed in the right pane:

- Automatically generate filters from launch configuration settings—by default, data collection filters are created for your application, based on the configuration settings defined.
- Switch to JProbe perspective at launch—by default, Eclipse switches to the JProbe perspective when you launch an application.
- Open the attach to running JProbe session dialog at launch—by default, the Attach to JProbe Session dialog box appears when you launch an application with JProbe analysis.

- 3 To change any of these default launching settings, clear its corresponding check box, then click **Apply**.

Coverage Preferences

To set JProbe preferences specific to a coverage analysis:

- 1 On the Eclipse menu bar, click **Window > Preferences**.
- 2 In the Preferences dialog box, click **JProbe > Coverage**.

The coverage preferences are displayed in the right pane:

- Filter out Catch Blocks—by default, all catch blocks are included in the metrics calculated.
 - Missed Lines Color—sets default color for any missed line.
 - Missed Conditions Color—sets default color for any missed condition.
 - No Data for Line Color—sets default color for cases when no data is available for a line.
 - Filtered Catch Block Color—sets default color for the catch blocks filtered out.
- 3 To change these default coverage settings, see sections [Filter Out Catch Blocks](#) and [Customizing Colors Used to Display Results](#).

Filter Out Catch Blocks

The data for catch blocks is included in the following metrics:

- % Missed Lines
- Missed Lines
- Total Lines
- % Missed Conditions
- Missed Conditions
- Total Conditions

Catch blocks, however, are often impossible to test. If you choose not to test them, you may find it easier to interpret your results without the catch block data inflating the missed values. You can set a filter to remove the data from the calculations.

To remove catch block data from calculations:

- In the Coverage Preferences pane, clear the Filter out Catch Blocks check box.

Exceptions

Although it is not generally considered good coding practice, it is possible that a line can have both normal statements and catch block statements.

Catch block filters do not apply in the followings situations:

- Try and catch appear in the same line of code.

```
try {Thread.sleep(1000);} catch (Exception e) {}
```

If `Thread.sleep(1000)` is executed, coverage is 100%; if not, coverage is 0%.

- Executed code and catch appear in the same line of code.

```
try {
    System.out.print(obj.toString());
    System.out.println();} catch (Exception e) {
    System.exit(1);}
```

If `System.out.print()` is executed, the coverage is 100% regardless of whether or not it throws an exception.

Customizing Colors Used to Display Results

You can change the colors used to represent hit lines, missed lines, missed conditions, catch blocks that have been filtered out, and unexecuted code.

To set colors:

- 1 In the Coverage Preferences pane, click the color beside the preference's name.
- 2 In the Color dialog box, select a new color, and then click **OK**.

The selected color is now displayed in the Coverage Preferences pane.

Memory Preferences

To set JProbe preferences specific to a memory analysis:

- 1 On the Eclipse menu bar, click **Window > Preferences**.
- 2 In the Preferences dialog box, click **JProbe > Memory**.

The memory preferences are displayed in the right pane:

- Memory Unit—defines the unit for measuring heap data displayed in memory views: Bytes (default option), Kilobytes, and Megabytes.
- 3 To change the default memory settings, select an option button from the list.

Performance Preferences

To set JProbe preferences specific to a performance analysis:

- 1 On the Eclipse menu bar, click **Window > Preferences**.
- 2 In the Preferences dialog box, click **JProbe > Performance**.

The performance preferences are displayed in the right pane:

- Show Percentage Time Values in Methods View Trees—defines if the results are displayed in the Methods View as percentages or as actual time values.
 - Time Unit—defines the unit for measuring time intervals displayed in performance views: *Nanoseconds*, *Microseconds*, *Milliseconds* (default option), and *Seconds*.
 - Time Precision—defines the precision used when calculating the time units displayed in performance views: no decimal places (default option), one, two, three, four, or five decimal places.
 - JDBC Truncation Style—defines the JDBC truncation method: *Truncate at the Beginning of Connection and Statement Strings*, *Truncate in the Middle of Connection and Statement Strings*, or *Truncate at the End of Connection and Statement Strings*.
- 3 To change the default performance settings, select an option button from the lists.

Managing Licenses

Before you can use JProbe, you must provide a valid license file. JProbe is then unlocked for purchase or evaluation, depending on the license file you provide.

JProbe has four types of licensing models available: node locked, per seat, concurrent, and enterprise. For more information, review the JProbe licensing options on the Quest Web site at: http://www.quest.com/jprobe/licensing_info.aspx.

To define or modify the license file to use with JProbe:

- 1 From the main menu, select **JProbe > Manage Licenses**.

The **JProbe Licensing** dialog box appears.

2 Select the license model you want to use by clicking the correspondent tab:

- Node Locked / Per Seat Licenses
- Concurrent Licenses

Note To obtain a JProbe Trial Licence, click **Obtain License**. The JProbe® Trial License Web page appears in a browser window.

3 Fill in the required information as specified in the GUI.

For more information about using the license server, see the *Quest Software License Server Administrator's Guide*.

Integrating JProbe into Eclipse

This chapter describes how to integrate JProbe into an Eclipse environment. The integration process varies, depending on the type of application that you want to analyze with JProbe.

When Eclipse is launched with the JProbe plugins, several menu options are added to the standard Eclipse menu bar. These menus allow you to run different types of application with JProbe, view the runtime session in the JProbe perspective, and analyze the results with JProbe.

This chapter contains the following sections:

| | |
|--|----|
| Integrating JProbe to Run Java SE Applications | 48 |
| Integrating JProbe to Run Java EE Applications | 55 |

Integrating JProbe to Run Java SE Applications

You can integrate JProbe with Eclipse to run the following types of Java SE applications:

- Java Application
- Eclipse Application
- JUnit
- JUnit Plug-in Test
- OSGi Framework

The integration process includes the following steps:

- 1 Configuring your Java SE application. Follow the instructions in one of these procedures, as needed:
 - [Configuring a Java Application in Eclipse](#)
 - [Configuring an Eclipse Application in Eclipse](#)
 - [Configuring JUnit in Eclipse](#)
 - [Configuring JUnit Plug-in Test in Eclipse](#)
 - [Configuring an OSGi Application in Eclipse](#)
- 2 Opening the JProbe perspective in Eclipse. For more information, see “[Opening the JProbe Perspective](#)” on page 28.
- 3 Launching your Java SE application with JProbe in Eclipse. For more information, see “[Launching a Java SE Application with JProbe](#)” on page 54.

Configuring a Java Application in Eclipse

Before running a Java application with JProbe in Eclipse, you need to configure the Java application in your environment.

Note If your Java application is already configured in Eclipse, you can ignore this section.

Configuring a Java application in Eclipse includes the following steps:

- 1 [Creating a Java Project](#).
- 2 [Creating a Java Package](#).

- 3 [Importing a Java Application into Your Project.](#)
- 4 [Creating a New Run Configuration.](#)

Creating a Java Project

To create a Java project:

- 1 Select **File > New > Project**.
The New Project (Select a wizard) dialog box appears.
- 2 Select **Java > Java Project**, then click **Next**.
The New Project (Create a Java Project) dialog box appears.
- 3 Fill in the **Project name** text box.
- 4 Select the **Create new project in workspace** check box and click **Next**.
The New Java Project (Java Settings) dialog box appears.
- 5 Click **Finish**.
The Java project is now created.

Note You can view the newly created project in the Package Explorer view in the Java perspective. For details about how to open a Java perspective, refer to Eclipse documentation.

Creating a Java Package

To create a Java package:

- 1 In the Java perspective, select **File > New > Package** from the menu bar.
The New Java Package (Create a new Java package) dialog box appears. The Source folder text box is already filled with the name of your *<Java project>/src*.
- 2 Type the name of your package in the Name text box.
Note For example, to perform JProbe's Network tutorial, you need to create two packages and name them *demos.memory.network* and *demos.memory.sim*.
- 3 Click **Finish**.
The Java package is now created.

Note You can view the newly created package in the Package Explorer view in the Java perspective.

Importing a Java Application into Your Project

To import a Java application into your project:

- 1 Select **File > Import**.

The Import (Select) dialog box appears.

- 2 Click **General > File System** and click **Next**.

The Import (File system) dialog box appears.

- 3 In the **From directory** box, select the directory where your Java application resides.

Note For example, to import JProbe's Network application, you need to choose the `<jprobe_home>\demos\memory\network` directory path, then the `<jprobe_home>\demos\memory\sim` directory path.

The lower boxes are populated with the files existing in the selected directory.

- 4 Select the check boxes beside your Java application classes and images.

Note For example, to import the Network example classes, you would select all the .java files and image files in the `<jprobe_home>\demos\memory\network` and the `<jprobe_home>\demos\memory\sim` directories.

- 5 In the **Into folder** box, select a directory in which you want to import the Java application.

- 6 Select any additional importing settings from the Options section:

- Overwrite existing resources without warning
- Create complete folder structure
- Create selected folders only

- 7 Click **Finish**.

The Java application is imported into your project.

Note You can view the newly created package in the Package Explorer view in the Java perspective.

- 8 Compile your application to ensure that it has no errors.

Creating a New Run Configuration

To create a new run configuration for your Java application:

- 1 Select **Run > JProbe Configurations**.

The Run Configurations dialog box appears.

- 2 In the left pane, select **Java Application**, then click **New launch configuration**



The Run Configurations (Create, manage, and run configurations) dialog box appears.

- 3 In the **Name** text box in the right pane, type the configuration name for your Java application.
- 4 On the **Main** tab, choose your Java Project and Main class.

Note For example, to perform JProbe's Network tutorial, you need to choose `demos.memory.network.Network` as the main class.

- 5 Click **Apply**.

The new run configuration is created and appears in the left pane under Java Application.

- 6 Click **Close**.

You can now launch your application with the JProbe analysis. For more information, see “[Launching a Java SE Application with JProbe](#)” on page 54.

Configuring an Eclipse Application in Eclipse

Before running an Eclipse application with JProbe in Eclipse, you need to configure the Eclipse application in your environment.

Note If your Eclipse application is already configured in Eclipse, you can ignore this section.

Configuring an Eclipse application in Eclipse includes the following steps:

- 1 [Creating an Eclipse Plugin Project](#).
- 2 [Creating the Eclipse Plugin Run Configuration](#).

Creating an Eclipse Plugin Project

To create an Eclipse plugin project:

- 1 Select **File > New > Project**.

The New Project (Select a wizard) dialog box appears.

- 2 Select **Plug-in Development > Plug-in Project** and click **Next**.

The New Plug-in Project dialog box appears.

- 3 Type the name of the project in the **Project name** text box (for example, *com.jprobe.test*)
- 4 Click the **Create a Java project** check box.
- 5 Select the target platform for this plugin and click **Next**.
The New Plug-in Project (Plug-in Content) dialog box appears.
- 6 Edit the **Plug-in Properties** and **Plug-in Options** as needed and click **Next**.
The New Plug-in Project (Templates) dialog box appears.
- 7 If you want to use an existing template for your project, click the **Create a plug-in using one of the templates** check box.
The list of available template options becomes available.
- 8 Select the template you want to use (for example, *Hello, World*) and click **Next**.
The Sample Action Set dialog box appears.
- 9 Edit the **Action Class Name** and **Message Box Text** properties as needed and click **Finish**.


This type of project is associated with the Plug-in Development perspective. If this perspective is not already open in Eclipse, click **Yes** to open it now.

The Eclipse plugin project is created.

Note You can view the newly created project in the Package Explorer view in the Plug-in Development perspective.

Creating the Eclipse Plugin Run Configuration

To create a run configuration for your Eclipse plugin:

- 1 Select **Run > Run Configurations**.
The Run Configurations dialog box appears.
- 2 In the left pane, select **Eclipse Application**, then click **New launch configuration** .
The Run Configurations (Create, manage, and run configurations) dialog box appears.
- 3 In the right pane, type the name of the configuration for your Eclipse application in the **Name** text box. Leave the rest of the settings at their default values.
- 4 Click **Apply**.

The new run configuration is created and appears in the left pane under Eclipse Application.

- 5 Click **Run** to run the plugin inside Eclipse and verify that the configuration is working.

A new Eclipse window opens, displaying an additional user's plugin.

- 6 Click **Close**.

Configuring JUnit in Eclipse

Before running a JUnit application with JProbe in Eclipse, you need to configure the JUnit application in your environment.

The configuration process is similar to configuring Java applications in Eclipse. For more information, see [“Configuring a Java Application in Eclipse”](#) on page 48.

This guide assumes that you have already configured your JUnit application in your system prior to JProbe integration.

Configuring JUnit Plug-in Test in Eclipse

Before running a JUnit Plug-in Test application with JProbe in Eclipse, you need to configure the JUnit Plug-in Test application in your environment.

The configuration process is similar to configuring Java applications in Eclipse. For more information, see [“Configuring an Eclipse Application in Eclipse”](#) on page 51.

This guide assumes that you have already configured your JUnit Plug-in Test application in your system prior to JProbe integration.

Configuring an OSGi Application in Eclipse

Before running an OSGi application with JProbe in Eclipse, you need to configure the OSGi application in your environment.

The configuration process is similar to configuring Eclipse applications in Eclipse. For more information, see [“Configuring an Eclipse Application in Eclipse”](#) on page 51.

This guide assumes that you have already configured your OSGi application in your system prior to JProbe integration.

Launching a Java SE Application with JProbe

To launch a Java SE application with JProbe:

- 1 In the JProbe perspective, select **Run > JProbe Configurations**.
The JProbe Configurations dialog box appears.
- 2 In the left pane, choose one of the run configurations for your Java SE application: **Eclipse Application**, **Java Application**, **JUnit**, **JUnit Plug-in Test**, or **OSGi Framework**.
The name of the configuration appears in the right pane, in the **Name** text box.
- 3 In the right pane, on the **JProbe** tab choose the type of analysis to be performed while your application runs: **Memory**, **Performance**, or **Coverage**.
Note For example, to perform JProbe's Network tutorial, you would choose the Memory check box.
The lower part of the JProbe tab changes, depending on the type of analysis selected.
- 4 Configure additional analysis settings, as needed.
Note For example, to perform JProbe's Network tutorial, on the **Filters** tab you would click in the first row's **Action** cell and select **traces**.
- 5 For Eclipse applications only, specify any JProbe-specific advanced configuration settings (for example, `-jp_snapshot_basename=<name>`, `jp_snapshot_dir=<directory_path>`, or `jp_console_port=52991`) by adding them to the **VM arguments** text box on the **Arguments** tab.
Note These options are specific to JProbe and should be used only when running the application inside Eclipse with JProbe. Remove these options from the configuration settings before running the application in debug or run dialog with Eclipse. Failing to do so will result in error messages in the Eclipse Console view.
- 6 Click **Apply**.
The configuration settings for running your Java SE application with JProbe are saved.
- 7 In the JProbe Configurations dialog box, click **Run JProbe**.
Your Java SE application is now launched with the JProbe analysis.
Note For example, in the case of JProbe's Network tutorial, the Network Simulation dialog box opens.
JProbe searches for currently running engines, and displays this list in the Attach to Running JProbe Session dialog box.

- 8 Select an engine from the list of currently running engines, or enter in the **Host Name/ IP Address** and **Port #** fields the host name and port number of the computer running your application under JProbe. Click **OK**.

Tip Ensure that the port number is the one that was indicated in the Console tab when you launched the application.

- 9 In the *Add Snapshots To Project* section, choose an existing JProbe project for the snapshots captured during this session (by selecting its name from the list), or define a new one (by clicking **New JProbe Project**).

Note For details about creating a JProbe project for your snapshots, see “[Creating a Project for JProbe Snapshots](#)” on page 29.

- 10 Click **OK**.

The JProbe Console connects to the JProbe Analysis Engine running with the application. The runtime session view for the analysis type is displayed.

- 11 In Eclipse, check the messages in the Console view.

Note An empty command window may open. You can minimize this window, but do not close it, as it is required for JProbe to operate correctly.

The JProbe Connection Manager should now be available for Console connections on the default port (52991).

You can now create a JProbe project and attach JProbe to the running session. For more information, see “[Attaching JProbe to the Application](#)” on page 63.

Integrating JProbe to Run Java EE Applications

If the TPTP framework (Test and Performance Tools Platform) is installed in the Eclipse IDE, you need to disable the TPTP Profiling for Web Applications feature.

To disable the TPTP profiling for Web Applications plugin:

- 1 In Eclipse, choose **Help > Software Updates > Manage Configurations**.
The Product Configuration dialog box appears.
- 2 Click **Show Nested Features** to view all the nested features installed on your Eclipse IDE.
- 3 Choose **TPTP Profiling for Web Applications** and click its corresponding **Disable** link.
- 4 Click **OK**.

- 5 To disable TPTP profiling for Web applications, restart Eclipse.

Integrating JProbe into Eclipse to run Java EE applications involves the following steps:

- 1 [Creating an Application Server](#)
- 2 [Testing the Application Server](#)
- 3 [Setting the Server Timeout Delay](#)
- 4 [Running the Application Server with JProbe](#)

Creating an Application Server

By creating a server you create a pointer from the workbench to an existing installation of an application server. JProbe supports several types of application servers; for a complete list, see the *JProbe Installation Guide*.

To create an application server in Eclipse:

- 1 Open the Java EE perspective.
Note For information about perspectives, open the Eclipse online help and select **Workbench User Guide > Tasks > Working with perspectives**.
- 2 Select **File > New > Other**.
The New (Select a wizard) dialog box appears.
- 3 Select **Server > Server** and click **Next**.
The New Server (Define a New Server) dialog box appears.
- 4 In the **Server's host name** text box, type the fully qualified DNS name or the IP address of the host machine where the server is running. The default address is localhost.
- 5 Select the application server type from the **Select the server type** list (for example, **Apache > Tomcat v6.0 Server**).
- 6 Type the server name in the **Server name** text box and click **Next**.
- 7 Follow the instructions in the wizard to specify the details of the server that you want to create. This information is specific to the selected server.
- 8 In the **Add and Remove Projects** dialog box, modify the projects that are configured on this server.

The new server is now created and appears in the Server column in the Servers view.

Testing the Application Server

After creating an application server, it is a good idea to check that it starts and stops correctly.

To test the server in Eclipse:

- 1 In the **Servers** view, right-click the name of the server and choose **Start**.
The server starts within Eclipse and the new status (Started) is displayed in the State column. The Console view shows details about the status of the processes executed.
For some application servers, an additional window opens. Closing this window does not stop the server, which continues to run in the background.
- 2 In the Servers view, right-click the name of the running server and choose **Stop**.
The server is stopped and its new status (Stopped) is displayed in the State column. The Console view shows details about the status of the processes executed.

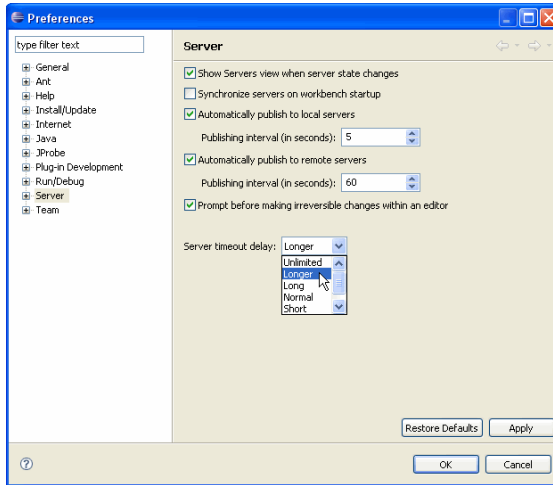
Setting the Server Timeout Delay

A Java EE application can run correctly with JProbe in Eclipse only when the timeout delay of the application server has been set to a specific value.

The timeout delay for all application servers running on an Eclipse 3.3 environment can be set globally.

To set the timeout delay for application servers running on an Eclipse 3.3 environment:

- 1 Select **Window > Preferences**.
The Preferences dialog box appears.
- 2 Click **Server** in the left pane.
The right pane displays the server configuration settings.
- 3 Set the Server timeout delay to **Long**, **Longer**, or **Unlimited** (the latter two options are recommended).



4 Click **Apply**.

This timeout delay applies to all application servers installed in your Eclipse 3.3 environment.

5 Click **OK**.

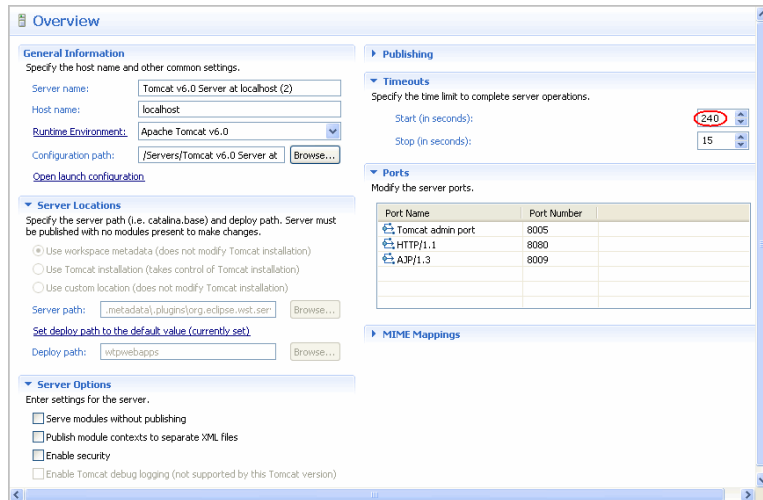
The timeout delay for application servers running on Eclipse 3.4 or Eclipse IDE for Java EE Development environments need to be set individually for each server.

To set the timeout delay for an application server running on an Eclipse 3.4 or Eclipse IDE for Java EE Development environment:

1 In the Servers view, right-click the name of the server and choose **Open**.

The server Overview view opens.

2 In the Timeouts section, set the **Start (in seconds)** field to a value between 200 - 300 seconds (the recommended value is 240 sec.).



- 3 Select **File > Save**.
- 4 Repeat steps 1 to 3 for each application server that you want to run with JProbe.

Running the Application Server with JProbe

To run an application server with JProbe:

- 1 In the Servers view, right-click the name of the server and select **Profile**.
The Run Server with JProbe dialog box opens.
- 2 On the **JProbe** tab, select the type of analysis to be performed while running your application: **Memory**, **Performance**, or **Coverage**.
- 3 On the **Filters** tab, define filters (if needed).
- 4 On the **Advanced Options** tab, specify any JProbe-specific advanced configuration settings (for example, set **JProbe Option Name** to `jp_console_port` and **Value** to `52991`).
- 5 Click **Finish**.

The application server now starts with JProbe and the new status (Profiling) is displayed in the State column. The Console view shows details about the status of the processes executed.

Note If you decide to cancel the operation (by clicking **Cancel**) instead of finishing it, the server starts in Eclipse, without JProbe. The status of the operation displayed in the State column is also Profiling, which may be misleading.

For some application servers, an additional window opens. You can minimize this window, but do not close it, as it is required for JProbe to operate correctly.

Note You can have only one application server started with JProbe in Eclipse at any given time.

- 6 Verify the connection port in the server Console view. If you clicked Finish in [step 5](#), the JProbe Connection Manager should now be available for console connections on the default port (52991).

With the exception of the Apache Tomcat server, all servers supported by JProbe use *jplauncher.exe* to start the application server and have three Console views associated with the process:

- *javaw.exe*, which displays the version of Java executable
- *jplauncher.exe*, which displays the port number available for the JProbe Connection Manager
- *JProbe Server Plugin Output*, which allows you to monitor the output/debug messages of JProbe server plugin

If you are running a Tomcat server with JProbe, two Console views can be opened:

- *javaw.exe*, which displays the port number available for the JProbe Connection Manager
- *JProbe Server Plugin Output*, which allows you to monitor the output/debug messages of JProbe server plugin

When multiple views are available to display, choose the Console of interest from the **Display Selected Console** list or icon on the lower toolbar.

You can now create a JProbe project and attach JProbe to the running session. For more information, see [“Attaching JProbe to the Application”](#) on page 63.

Running a JProbe Analysis in Eclipse

This chapter describes how to run your Java application with JProbe in Eclipse, which allows you to identify problems with your application and investigate those problems. You can then improve your code using the Eclipse development environment and re-run the JProbe analysis session with the improved code to see the improvement in memory usage, performance, or test case coverage.

This chapter contains the following sections:

| | |
|---|----|
| Running a Java Application with JProbe in Eclipse | 62 |
| Viewing Data in a Runtime Session | 65 |
| Viewing the Execution Log | 72 |
| Viewing the Session History | 73 |
| Viewing Snapshot Data | 73 |

Running a Java Application with JProbe in Eclipse

After you have configured an application, started it, and stopped it, you are ready to launch your application with JProbe. This is done in two steps:

- [Launching the Application](#)
- [Attaching JProbe to the Application](#)

Alternatively, you can start the JProbe session monitoring first, run it in the background, and then launch your application. For details, see [Running a JProbe Session in the Background](#).

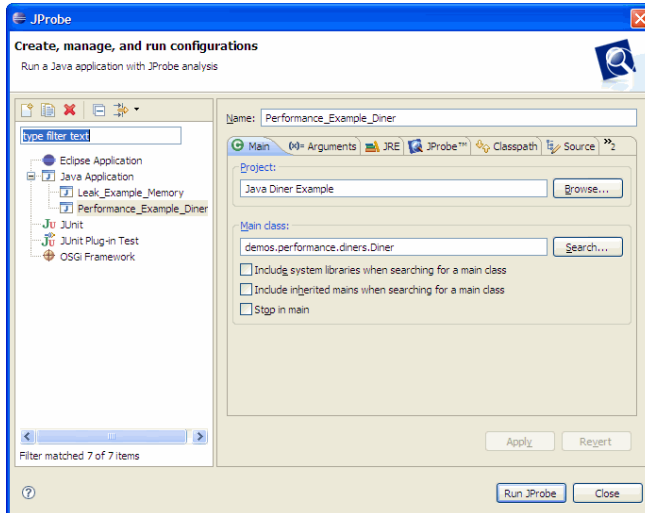
Launching the Application

This procedure assumes that you have already configured an application, as described in “[Integrating JProbe into Eclipse](#)” on page 47.

To launch an application with JProbe:

- 1 Click **Run > JProbe Configurations**.

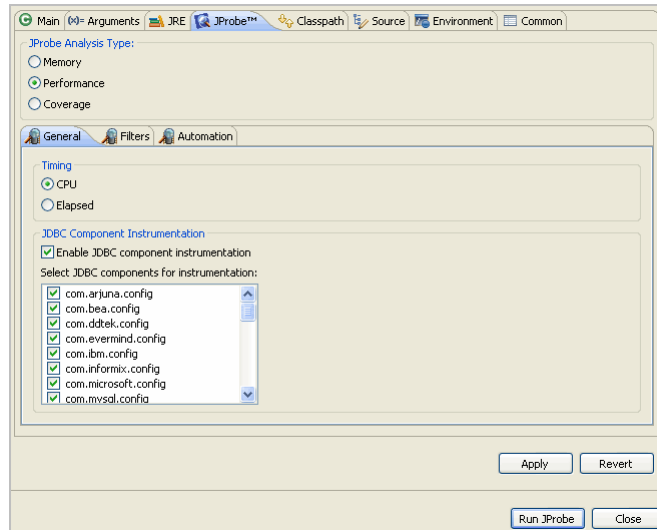
The JProbe window opens.



- 2 In the left pane, select the application that you want to run.

The text boxes in the Main tab are populated with the project name and main class.

- 3 If you want to make any changes to this configuration, click the **JProbe** tab.



- 4 Click **Run JProbe**.

The application starts and the Console view appears.

Note Take note of the port number in the Console view. You will need it in the next procedure.

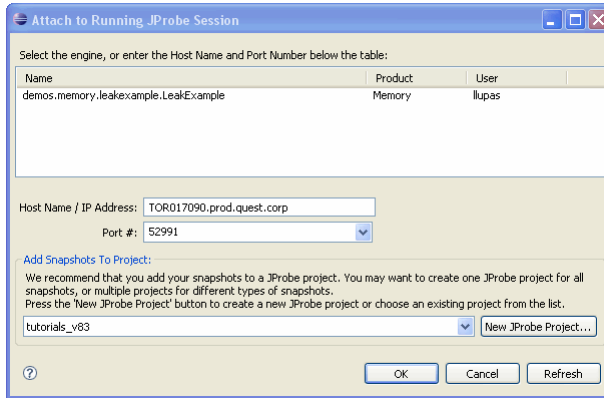
Attaching JProbe to the Application

After the application is running, you attach JProbe to it and monitor the runtime data. You can take snapshots while the session is running.

To attach JProbe to a running application:

- 1 If the Attach to Running JProbe Session dialog box is not already open, click **JProbe > Attach to Session**.

The Attach to Running JProbe Session dialog box appears.



- 2 Select an engine from the list of currently running engines, or enter in the **Host Name/ IP Address** and **Port #** fields the host name and port number of the computer running your application under JProbe. Click **OK**.

Tip Ensure that the port number is the one that was indicated in the Console tab when you launched the application. Only engines currently not connected to a Console are shown.

- 3 In the *Add Snapshots To Project* section, choose an existing JProbe project for the snapshots captured during this session (by selecting its name from the list), or define a new one (by clicking **New JProbe Project**).

Note For details about creating a JProbe project for your snapshots, see [“Creating a Project for JProbe Snapshots”](#) on page 29.

- 4 Click **OK**.

The runtime session view appears. You can now begin to monitor what is happening in your application in the runtime session views. For details, see [“Viewing Data in a Runtime Session”](#) on page 65.

Running a JProbe Session in the Background

You can start a JProbe session connection monitor before launching an application, and run it in the background.

To run a JProbe session in the background:

- 1 Click **JProbe > Attach to Session**.

The Attach to Running JProbe Session dialog box appears.

2 Click **OK.**

The JProbe Session Connection Indicator dialog box appears. JProbe tries to connect to the localhost (by default, on port number 52991).

3 Click **Run in Background.**

The JProbe session monitoring starts and runs in the background.

Tip You can check the status or terminate this task (later on) from the Eclipse Progress view (click **Window > Show View > Other** and select the **General > Progress** option).

Viewing Data in a Runtime Session

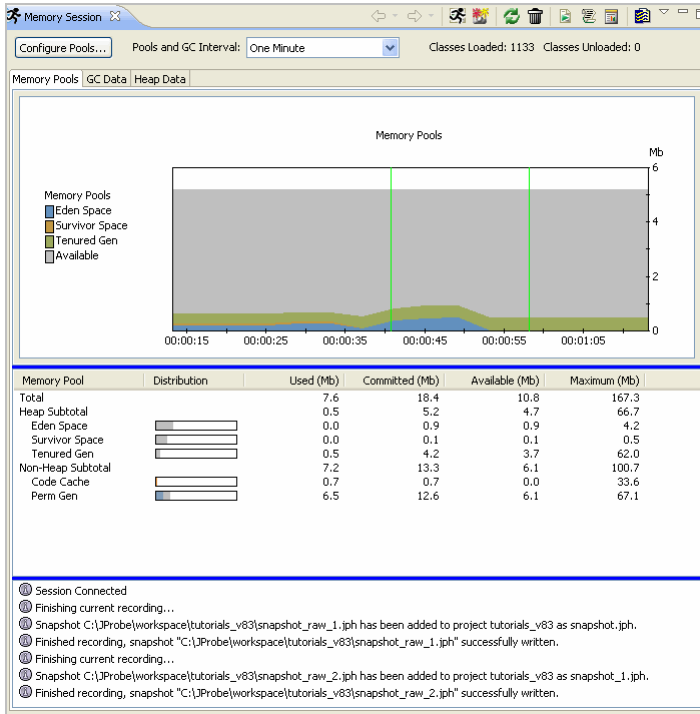
When you attach JProbe to a running session, the Session view opens automatically and displays the information that JProbe is collecting. This view is different for each of the analysis types.

Memory Session Runtime View

While you are running a Memory session, the runtime view displays memory pools and changes in counts in the Java heap, as well as garbage collection data. The information is displayed in three tabs: Memory Pools, GC Data, and Heap Data.

Memory Pools

The data displayed in the Memory Pools tab varies, depending on the JVM and the options that you are using. For JVMs that use generational garbage collectors, heap pools typically represent the generations. For example, in the Sun JVMs, with some options (as shown below), Tenured Gen is the old generation, while Eden Space and Survivor Space together make up the new generation.



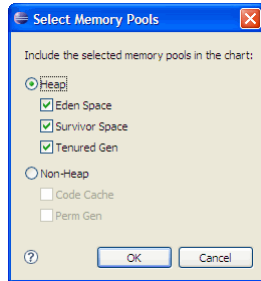
The table displays both heap and non-heap data, but you can control which memory pools are displayed in the chart. You can also change the data display interval for the memory pools and garbage display.

Note Changing the Memory Pools settings does not affect the polling interval, which is always 5 seconds.

To configure memory pools:

- 1 Click **Configure Pools**.

The Select Memory Pools dialog box appears.



- 2 Select the memory pools that you want to display and click **OK**.

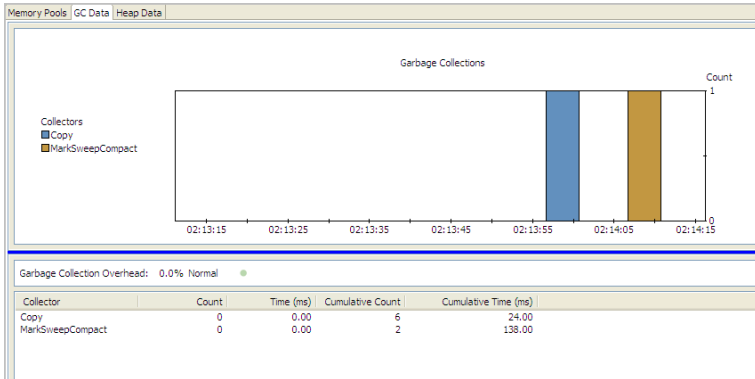
The graph is refreshed to display the pools that you chose.

To change the Memory and GC data display interval:

- Select a different interval from the **Pools and GC Interval** list.

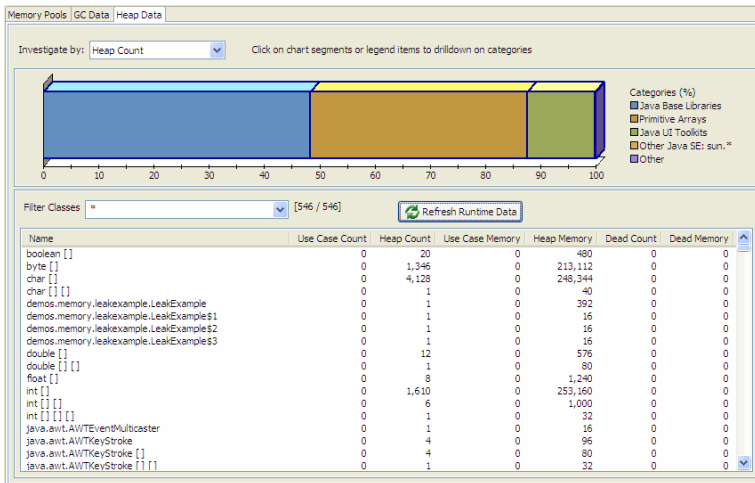
GC Data

The upper pane of the GC Data tab displays garbage collections in chart form, showing which collector was invoked and how many times it was invoked. The lower pane summarizes this, along with how much time each collector spent. The Garbage Collection Overhead value indicates the percentage of time that the JVM was exclusively performing garbage collection, compared to the elapsed time the program has been running.



Heap Data

The Heap Data tab displays the same types of information as the Instances view. It is populated when you click the **Refresh Runtime Data** button.



For more detailed information about the data displayed in these views, see the section on “Exploring the Memory Runtime Summary View” in the *JProbe User Guide*.

After you have captured the data in a snapshot (either manually or by using a trigger), you can use the heap investigation tools, starting with the Instances view, to understand the references among classes in your application. For details, see “Exploring the Instances View” in the *JProbe User Guide*.

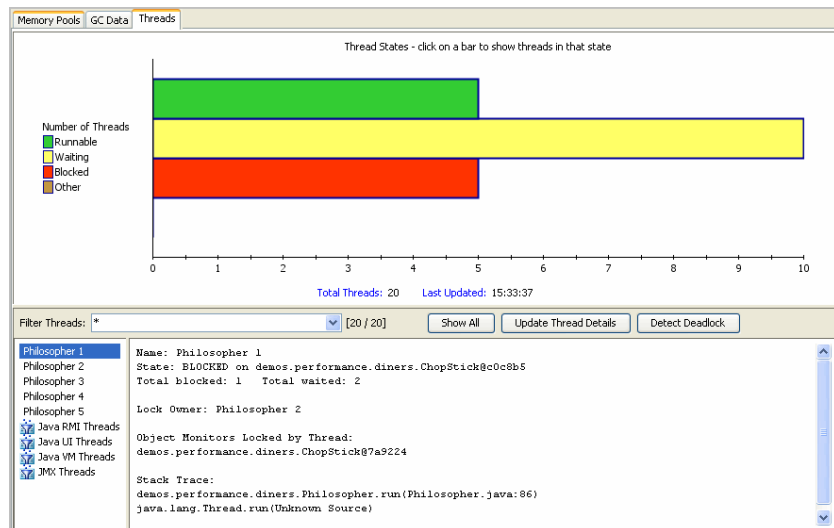
Performance Session Runtime View

While you are running a Performance session, the runtime view displays memory pools, garbage collection data, and threads. The information is displayed three tabs: Memory Pools, GC Data, and Threads. The Memory Pools and GC Data tabs are the same as in a Memory session. For information about them, see “[Memory Session Runtime View](#)” on page 65.

Note When you click **Take an HPROF Heapdump** on the toolbar, an HPROF dump is taken, converted to a JProbe snapshot, and added to the Snapshot Navigator. You can then analyze it using the JProbe Memory analysis tool. To use this feature, you must run your application with a Sun JVM.

Threads

The Threads tab contains a graph and a list, which are updated every four seconds.



The graph displays how many threads are in a given state at a certain time. The threads are grouped as presented in the following table.

| Group | Thread State | Thread Description |
|-------|--------------|--------------------|
|-------|--------------|--------------------|

| | | |
|----------|---------------|--|
| Runnable | RUNNABLE | A thread executing in the Java virtual machine. |
| Waiting | TIMED_WAITING | A thread that is waiting for another thread to perform an action for up to a specified waiting time. |
| | WAITING | A thread that is waiting indefinitely for another thread to perform a particular action. |
| Blocked | BLOCKED | A thread that is blocked waiting for a monitor lock. |
| Other | NEW | A thread that has not started yet. |
| | TERMINATED | A thread that has exited. |

Clicking on a bar or a line in the legend reduces the list of threads in the lower pane, so that only threads in that group are displayed. You can return to the full list by clicking the **Show All** button.

The total number of threads is displayed beneath the graph, along with the time the graph was last updated. Historical data is not maintained for the graph.

The lower pane displays a list of all known threads, grouped into categories. This list is sorted in ascending order (threads followed by categories). When you click a thread on the list, details for the selected thread are displayed on the right pane, including:

- Name of the thread.
- Thread state.
- The object upon which the thread is blocked, when applicable.
- The thread that owns the object that the thread is blocked, when applicable.
- Stack trace of the thread.
- List of object monitors locked by the thread.
- List of synchronizers locked by the thread.

You can reduce the list by clicking a bar in the graph, or by typing a filter into the **Filter Threads** field. The filter is a simple string filter that matches against thread names, not category names. Categories can be expanded by right-clicking on them and selecting **Expand Category**. Threads can be collapsed back into categories by right-clicking on them and selecting **Collapse Category**.

Thread details are only updated on demand. Users can update the details for the currently selected thread by pressing the **Update Thread Details** button or by double-clicking a thread name. The button is disabled when no threads are selected or a category is selected.

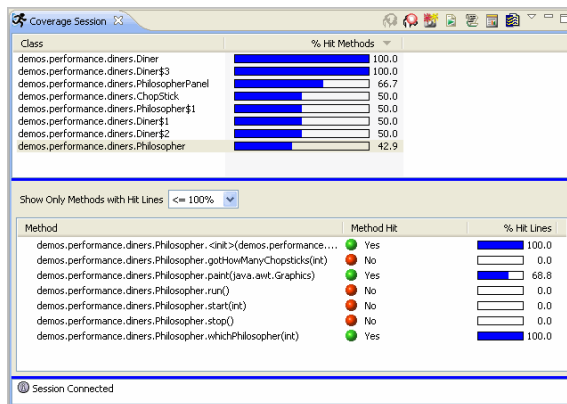
You can detect deadlocks on demand, by pressing the **Detect Deadlock** button. If a deadlock is detected, the threads list displays the threads involved in the deadlock. If no deadlock is detected, a message is displayed to that effect.

For more detailed information about the data displayed in the Performance runtime views, see the section on “Exploring the Performance Runtime Summary View” in the *JProbe User Guide*.

After you capture the data in a snapshot, you can use the performance investigation tools, starting with the Call Graph and Call Tree, to understand the performance issues in your program. For details, see the *JProbe User Guide*.

Coverage Session Runtime View

While you are running a Coverage session, the runtime browser provides a real-time view of test coverage for your application. It displays data in terms of hits; that is, the classes, methods, and lines of code that were exercised by your test case.



The upper table displays a list of classes hit by the test case and shows the percentage of methods that were hit within each class. When you select a class, the lower table shows the names of the methods in the class, whether or not the method was hit, and the percentage of lines that were hit. You can filter this table by percentage of hit lines.

To filter methods in the lower table:

- Select a different percentage from the list.

The table refreshes to display only those methods that have a percentage of hits below the chosen threshold.

For more detailed information about the data displayed in the Coverage views, see the section on “Exploring the Coverage Analysis Views” in the *JProbe User Guide*.

Viewing the Execution Log


When you are running a session, you can view the execution log from the session runtime view. The same information is also saved to every snapshot taken during the session.


If you are viewing a snapshot, you can open the execution log that was created when the snapshot was taken.

The following table summarizes the information in the execution log:

| Tab Name | Description |
|------------------|--|
| General | Contains information about the JProbe version, the application, and the environment in which they run. For a snapshot, it also contains the snapshot name and file size. |
| Actual | Contains the options that were passed to the JProbe Analysis Engine, expressed as command line JProbe options. These may be different than the options that were requested in the configuration. |
| Requested | Contains the options that were set in the configuration, expressed as command line JProbe options. |
| JVM | Contains information about the Java executable that is running the application. |

To view the execution log:

- 1 Do one of the following:
 - In the session runtime view, select **Runtime Execution Log** .

- In the Navigator, right-click a snapshot and select **JProbe > Execution Log** .
- 2 Review the information.
 - 3 Click **OK**.

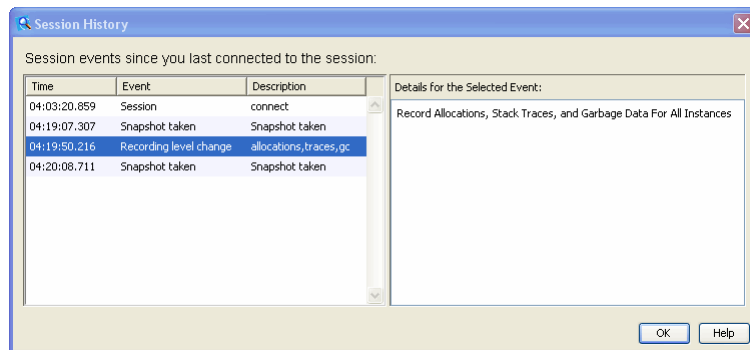
Viewing the Session History

When running a session, you can access the Session History from the Memory Session view toolbar.

The Session History dialog presents events that occurred since the Console has connected to the session (for example, the “start recording”, “stop recording”, “snapshot taken” events). A timestamp and a short description are provided in the left pane for each event. Additional details are presented in the right pane for the selected events. For example, the “start recording” event shows the details of the recording level and a list of filters (which may appear abbreviated, depending on their length), when applicable. Also, for example, the “snapshot taken” event shows the source of the request.

To view the session history:

- In the Memory Session view, click **View Session History**  on the toolbar.



Viewing Snapshot Data

You can take snapshots at any time while an application is running, or you can use triggers to take snapshots. A snapshot is always taken when you stop running an

application in JProbe. These snapshots are in the Navigator; you can open them and examine the data in several JProbe views.

To open a snapshot in a JProbe view:

- 1 Right-click the snapshot and select **JProbe > Open Snapshot**.

or

Double-click the snapshot.

The snapshot data is displayed in the selected view.

- 2 Interact with the view as you would in the JProbe Console.

For information about the Memory, Performance, and Coverage views, see the *JProbe User Guide*.

Index

A

about JProbe 8

about Quest Software 10

adding

categories 36

folders 39

application server

creating 56

running with JProbe 59

setting timeout delay 57

testing 57

attaching JProbe to a Java application 63

C

categories

adding 36

creating and managing 36

deleting 40

editing 39

renaming 40

configuring

Eclipse applications in Eclipse 51

Java applications in Eclipse 48

JUnit in Eclipse 53

JUnit Plug-in Test in Eclipse 53

OSGi applications in Eclipse 53

contacting Quest 11

creating

a project for JProbe snapshots 29

application servers 56

categories 36

Eclipse Plug-in project 51

Eclipse Plug-in run configuration 52

Java package 49

Java project 49

Java run configuration 50

D

deleting

categories 40

folders 40

documentation

core 9

feedback 9

suite 9

downloading JProbe Plugins for Eclipse 16

downloading plugins

from the ZIP file 16

using the Update Manager 16

E

Eclipse applications

configuring in Eclipse 51

creating an Eclipse Plug-in project 51

creating Eclipse Plug-in run configuration 52

editing categories 39

execution log, viewing 72

F

filters, syntax 37

folders

- adding 39
- deleting 40
- renaming 40

G

GC Data 67

GEF plugin 15

Graphical Editing Framework plugin 15

H

Heap Data 68

heap dumps, importing 31

I**importing**

- heap dumps 31
- Java application in your project 50
- snapshots 33

installing

- GEF plugin 15
- JProbe Plugins for Eclipse
 - generic Eclipse users* 14
 - Pulse users* 20

integrating JProbe

- for running Java EE applications 55
- for running Java SE applications 48
- into Eclipse 47

J**Java application**

- attaching JProbe to a 63
- configuring in Eclipse 48
- creating a Java package 49
- creating a Java project 49
- creating run time configuration 50

importing in your project 50

launching with JProbe in Eclipse 62

running with JProbe in Eclipse 62

Java SE applications, launching with JProbe 54

JProbe analysis, running in Eclipse 61

JProbe community 11

JProbe integration

- for running Java EE applications 55
- for running Java SE applications 48
- into Eclipse 47

JProbe Plugins for Eclipse, downloading 16

JProbe session

- running in the background 64

jprobe-eclipse-plugins.zip

- downloading JProbe Plugins for Eclipse 16

JUnit Plug-in Test, configuring in Eclipse 53

JUnit, configuring in Eclipse 53

L**launching Eclipse 18**

- by editing the INI file 18
- from Pulse 24
- from the command line 19
- with JProbe Plugins for Eclipse 18

launching Java application, with JProbe in Eclipse 62

launching Java SE

- application with JProbe 54

licenses, managing 44

M**managing**

- categories 36
- JProbe preferences 40
- licenses 44

Memory Pools 65

O

opening the JProbe perspective 28

OSGi applications, configuring in Eclipse 53

P

preferences, managing 40

project

creating for JProbe snapshots 29

JProbe demo code 34

R**renaming**

categories 40

folders 40

run configuration

for Eclipse applications 52

for Java applications 50

running

a Java application with JProbe in Eclipse 62

a JProbe analysis in Eclipse 61

application servers with JProbe 59

running JProbe session in the background 64

runtime session, viewing data 65

S

session history, viewing 73

setting, server timeout delay 57

snapshot data, viewing 73

snapshots

creating a project for 29

importing 33

support 11

syntax for filters 37

system requirements 14

T**tabs**

GC Data 67

Heap Data 68

Memory Pools 65

Threads 69

technical support 11

testing, application servers 57

text conventions 10

threads 69

timeout delay, setting 57

U**Update Manager**

downloading JProbe Plugins for Eclipse 16

V**viewing**

execution log 72

session history 73

snapshot data 73

viewing data in a runtime session

Coverage 71

Memory 65

Performance 69

