Profiling LCSim with Netbeans

Netbeans

If you don't have Netbeans on your machine already, download and install it. You should get the Netbeans 6.0.1 release, thought any 6.x release will probably suffice.

Install the Netbeans Maven plugin.

Under Tools -> Plugins -> Available Plugins, check the box next to Maven. Click Install.

Profiler Configuration

The profiler needs to be calibrated before it can be used. Select Profiler -> Advanced Commands -> Run Profiler Calibration.

Some information is required to connect the profiler to our application.

Select Profiler -> Attach Profiler to pull up the Attach Profiler window.

🗸 Attach Profiler					
Attach to: <external application=""></external>					
Monitor	Analyze Performance				
• <u>Analyze Performance</u> • Create Custom	 Entire application Part of application Eilter: Exclude Java core classes Show filter value Edit filter sets 				
Memory	Use defined Profiling Points <u>Show active Profiling Points</u> Overhead: <u>Advanced settings</u>				
Attach Mo <u>d</u> e: Local direct attach to Java Application, <u>change</u>					
	▶-▶ Attach Cancel <u>H</u> elp				

Click on the Change link at the bottom of the window to bring up the Attach Wizard.

🗙 Attach Wizard		
Steps	Select Target Type	
 Select Target Type Review Attach Settings Manual Integration 	Type of the attach target T <u>a</u> rget Type Application	•
	Attach method	Attach invocation
	● <u>L</u> ocal ○ <u>R</u> emote	● <u>D</u> irect ○ Dynamic (JDK 1.6)
	Choose Local to profile local App Choose Direct attach to start JVM profiler connects. Choose Dynar attach the profiler at any time.	plication. Choose Remote to profile remote Application. If of the Application and block its exectution until the nic attach to start the Application normally and then
	< <u>B</u> ack	Next > Einish Cancel Help

Make sure Local and Direct are selected. Click the Next button twice.

Attach Wizard				
Steps	Manual Integration			
1. Select Target Type 2. ReviewAttach Settings 3. Manual Integration	Select the Java platform that will run your Java Application Java SE 5.0_04 and newer			
	Follow these instructions to manually integrate the Profiler with Java Application			
	Step 1: Run the application using Java SE 5 (JRE or JDK). When starting the application, provide the extra startup option to the java command: -agentpath:/Home/apps/netbeans-6.0.1/profiler2/lib/deployed/jdk15/linux/libprofilerinterface.so=/Home/apps/netbeans-6.0.1/profiler2/lib,5140 Step 2: The JVM will start, but will not proceed with application execution until you connect the profiler.			
	Hint: When profiling CPU, you should set a meaningful instrumentation filter and/or select Part of Application option to reduce profiling overhead.			
	< <u>B</u> ack Next > Einish Cancel Help			

Copy and paste the entire string beginning with "-agentpath" to a local scratch file. This will be needed to connect the running code to the profiler.

Project Checkout and Setup

Checkout GeomConverter and org.lcsim from cvs.

The Maven 2 POM file does not have the standard name in either of these projects. In Linux, you can create a link with the correct name.

Do this before opening the project in Netbeans.

The Create Shortcut command may do the same thing in Windows, though I have not tested this.

Open LCSim by selecting Open Project. Click on the LCSim root directory and select Okay.

You should see LCSim in your Netbeans project window now.



Right click on LCSim and select Set as Main Project.

Creating the Main Routine

The profiler will only run on classes that have a main routine.

I created a class called *ProfilingTest* in the package org.lcsim. Technically, this class could be called anything and could be located anywhere under the **src** directory in lcsim.

✓ New Java Class	
Steps 1. Choose File Type 2. Name and Location	Name and Location Class Name: ProfilingTest
	Project: org DOT Icsim (jar) Location: Source Packages Package: org.lcsim Created File: /a/surrey01/vol2/g.lcd.mc/prj/users/jeremym/cvs/lcsim/dev/src/org/lcsim/ProfilingTest.java
	< <u>B</u> ack Next > <u>Finish</u> Cancel <u>H</u> elp

Add a main routine to this class with some test code. Here is my version of ProfilingTest.java.

```
package org.lcsim;
public class ProfilingTest
{
    public static void main(String args[])
    {
        System.out.println("hello profiler");
    }
}
```

Right click on LCSim again and select Properties -> Run.

Paste the string you saved earlier into the VM Options field.

The Run settings should now look similar to this.

¥	org DOT Icsim (jar)		
	Lategories: General Sources Run Actions	<u>M</u> ain Class: Arguments:	org.lcsim.ProfilingTest Browse
	Actions	Working Directory: <u>V</u> M Options: (e.g.	-agentpath:/Home/apps/netbeans-6.0.1/profiler2/lib/deploye -Xms10m)
			OK Cancel

Netbeans will now wait for a profiler connection before running your main routine.

Connecting to the Profiler in Netbeans

Run the test code by right clicking on the LCSim project and selecting Run.

If all was setup correctly, you should see something like the following in your log window.

```
Profiler Agent: Initializing...
Profiler Agent: Options: >/Home/apps/netbeans-6.0.1/profiler2/lib,5140<
Profiler Agent: Initialized succesfully
Profiler Agent: JNI On Load Initializing...
Profiler Agent: JNI OnLoad Initialized succesfully
Profiler Agent: Waiting for connection on port 5140 (Protocol version: 8)</pre>
```

This means that you can now launch the profiler using **Profile -> Attach Profiler**, and your main routine will be executed and profiled according to your selections.

Connecting to the Profile from the Command Line

The profiler can also be used to connect to compilations run from the command line.

The following command runs the previously created *ProfilingTest* main from the command-line.

```
java -agentpath:/Home/apps/netbeans-6.0.1/profiler2/lib/deployed/jdk15/linux/libprofilerinterface.so=/Home/apps
/netbeans-6.0.1/profiler2/lib,5140 \
-jar ./target/executable-netbeans.dir/lcsim-1.2.jar org.lcsim.ProfilingTest
```

After this statement is executed, you should see the same log message indicating that the program is waiting for a connection.

Now execute Profile -> Attach Profiler and the command line program will be profiled.