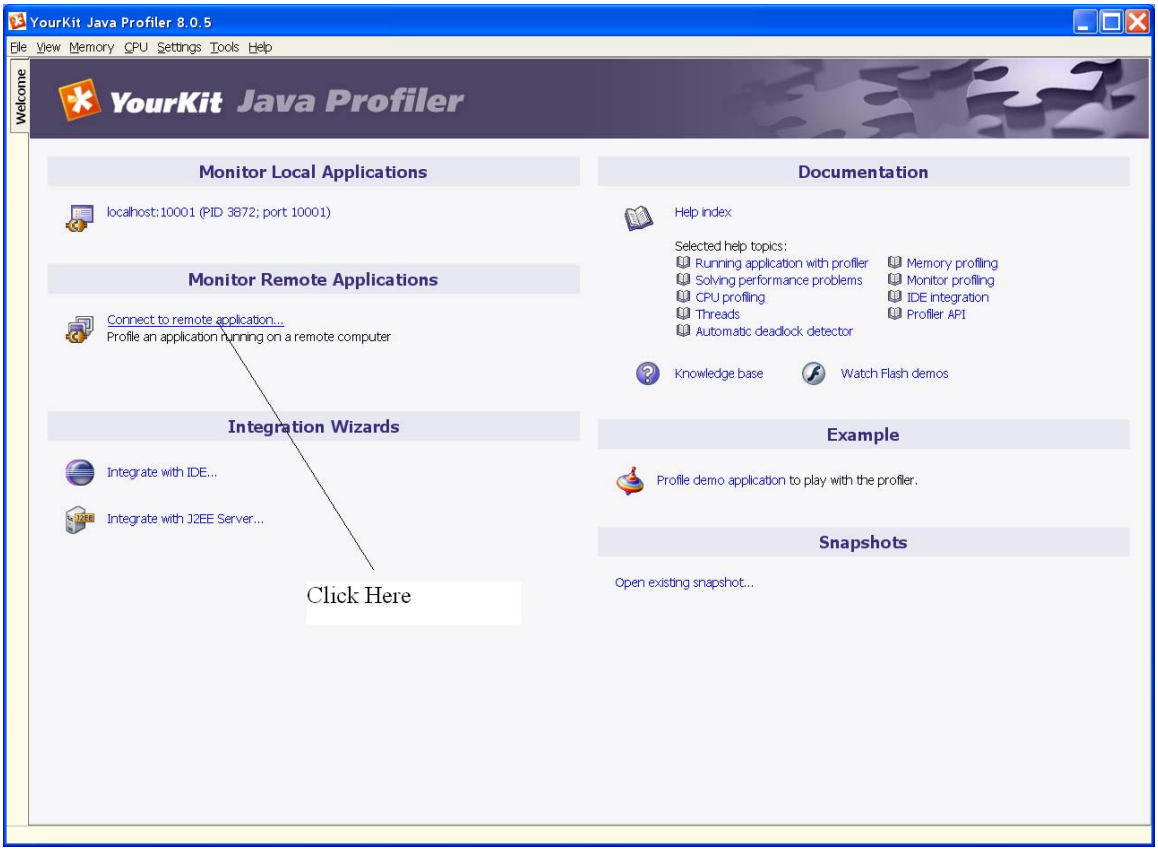
This is a tutorial walking through how I profile java components within MATLAB 2007a using YourKit java profiler. You can download a trial version at http://www.yourkit.com/.

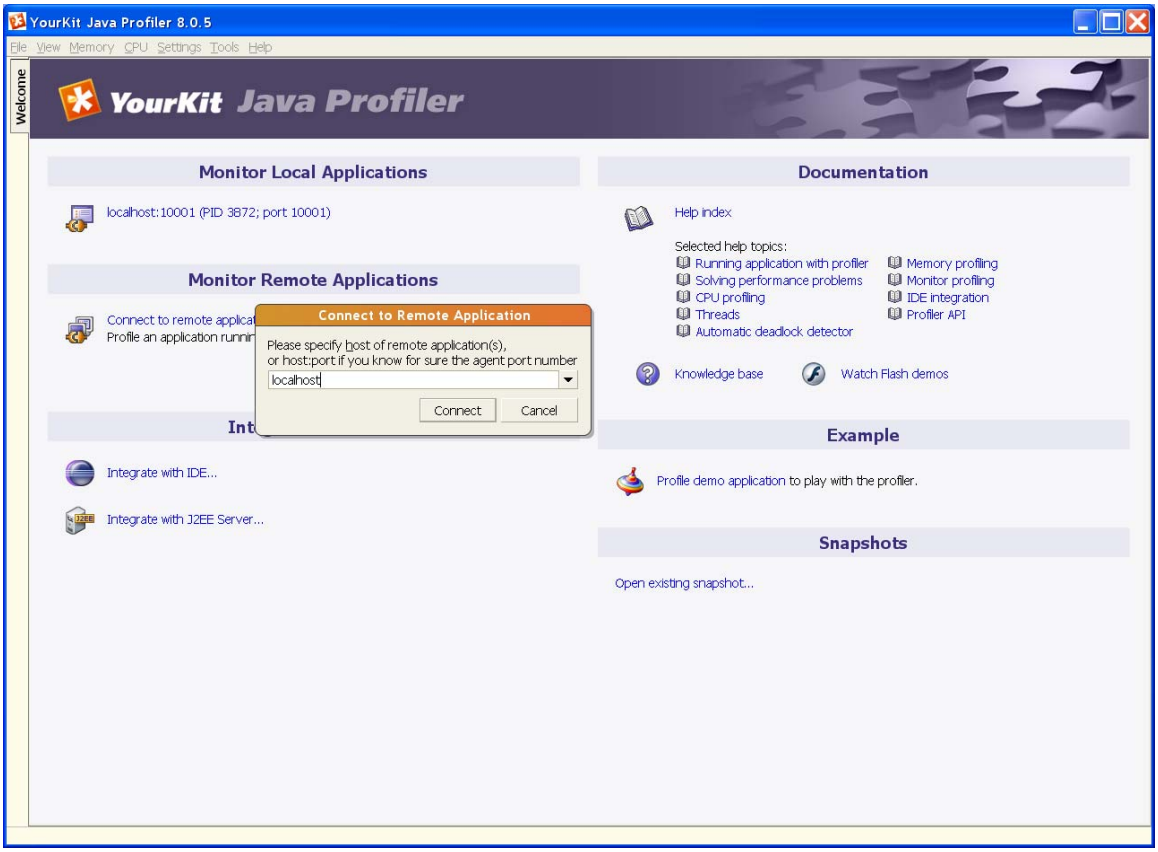
First add to your environmental variable PATH C:\Program Files\YourKit Java Profiler 8.0.5\bin\win32 which contains the file yjpagent.dll.

Then add a line to your java.opts file "-agentlib:yjpagent=disablej2ee,noj2ee" and then launch MATLAB.

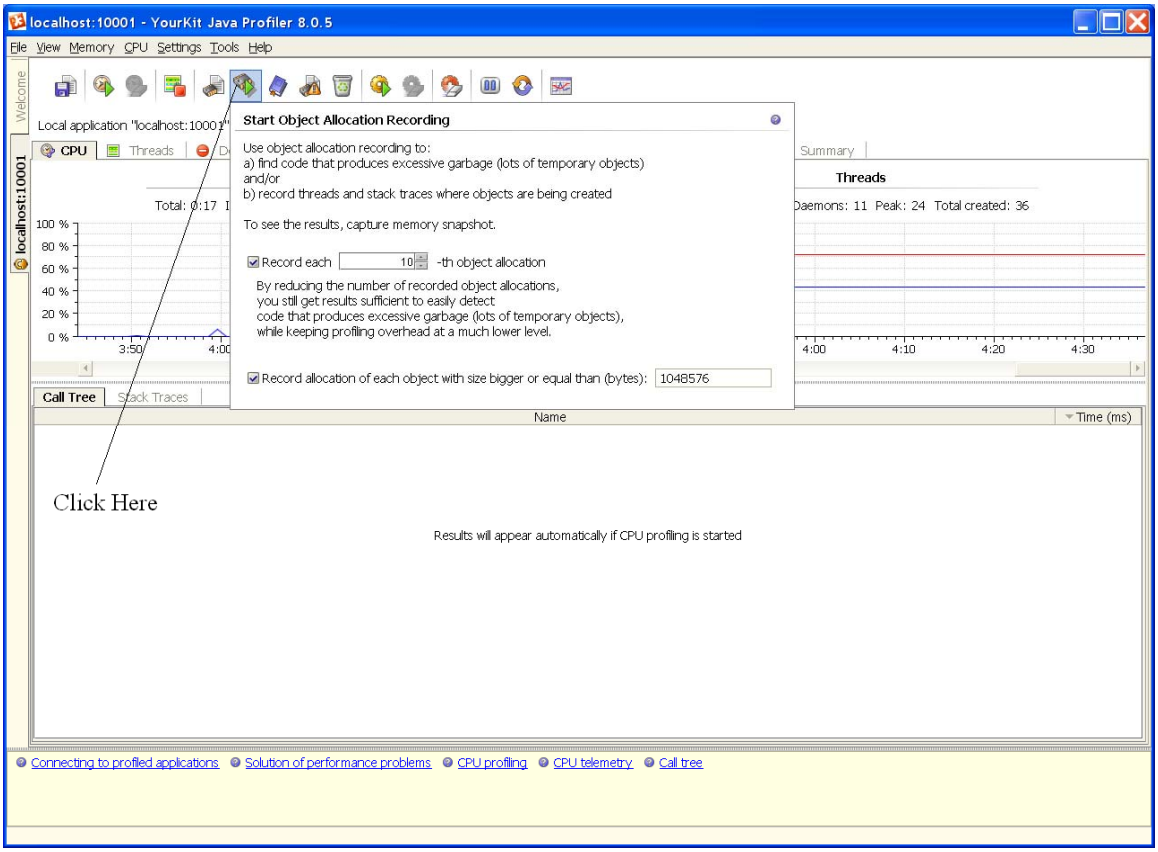
Next launch YourKit (8.0.5), you'll see the Welcome (side) tab.



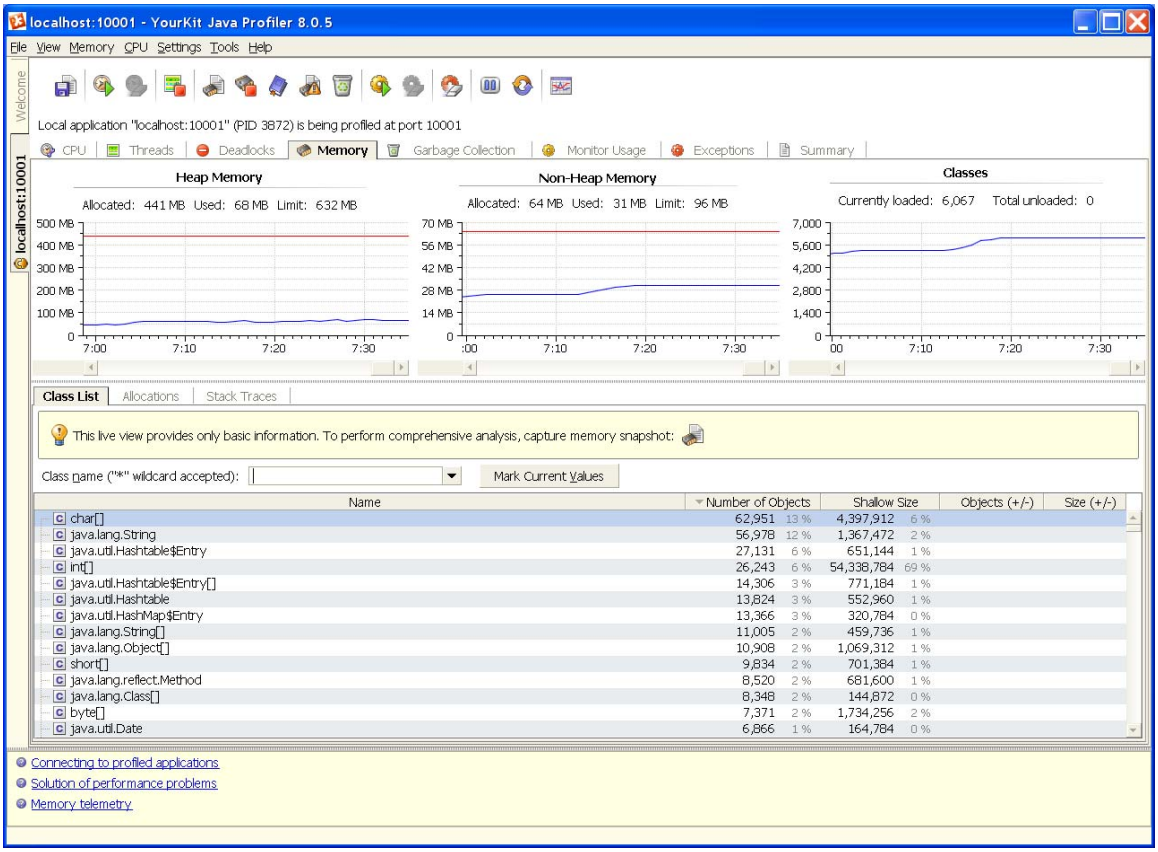
Click on 'Connect to remote application'.



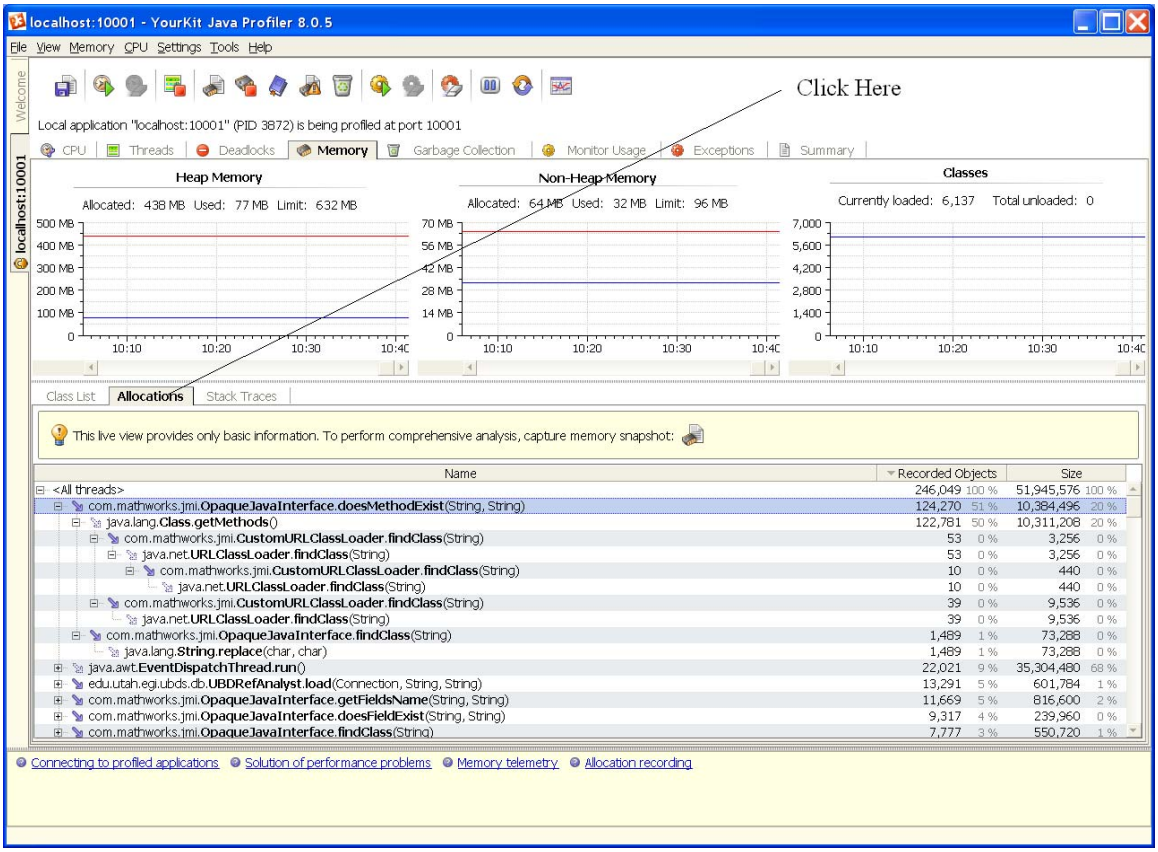
Fill in ‘localhost’, the port (according to the documentation) is automatically selected starting from 10001. If everything is ok, you should see:



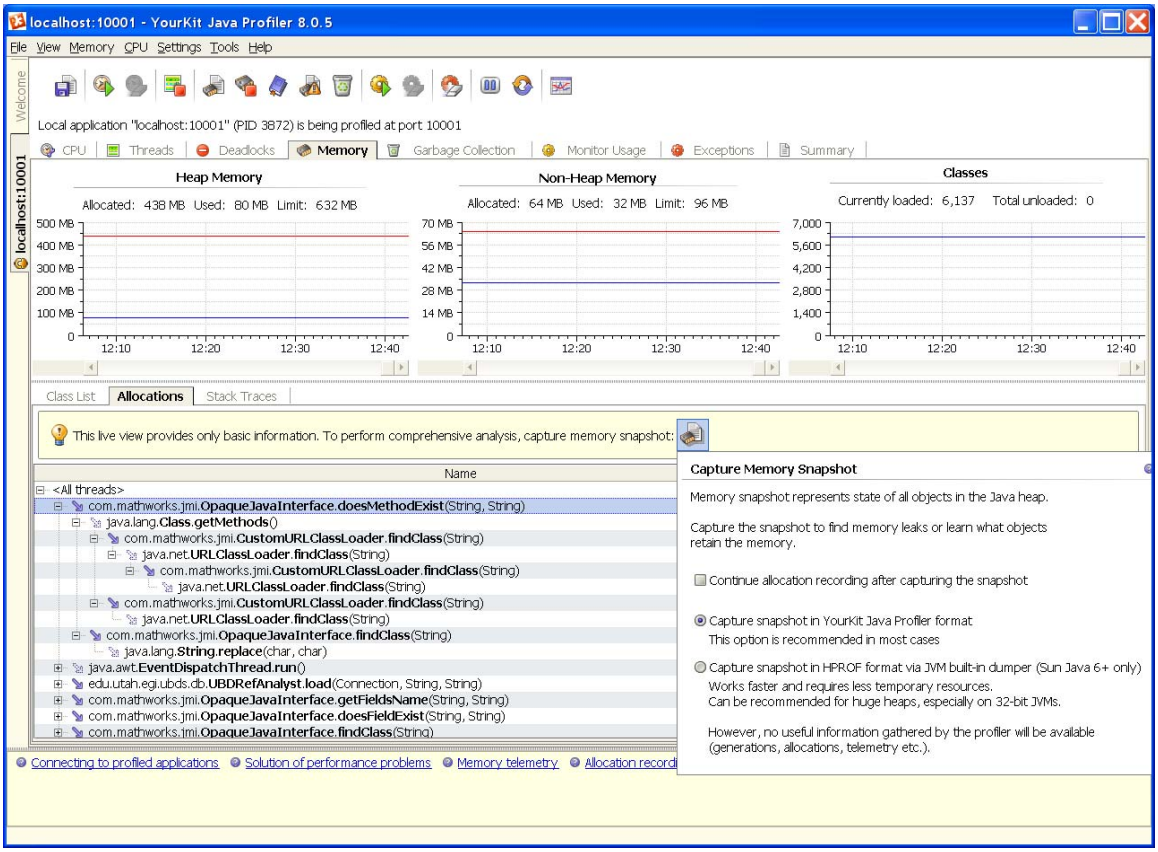
Now click on the ‘Start Object Allocation Recording’ button on the toolbar. You should then see message “[YourKit Java Profiler 8.0.5] Allocation recording is started” in your MATLAB command window.



Here you see the current memory allocations (I've selected the "Memory" tab). Now I highlighted the 'char[]' entry and then click on the 'Allocations' tab in the middle of the screen…

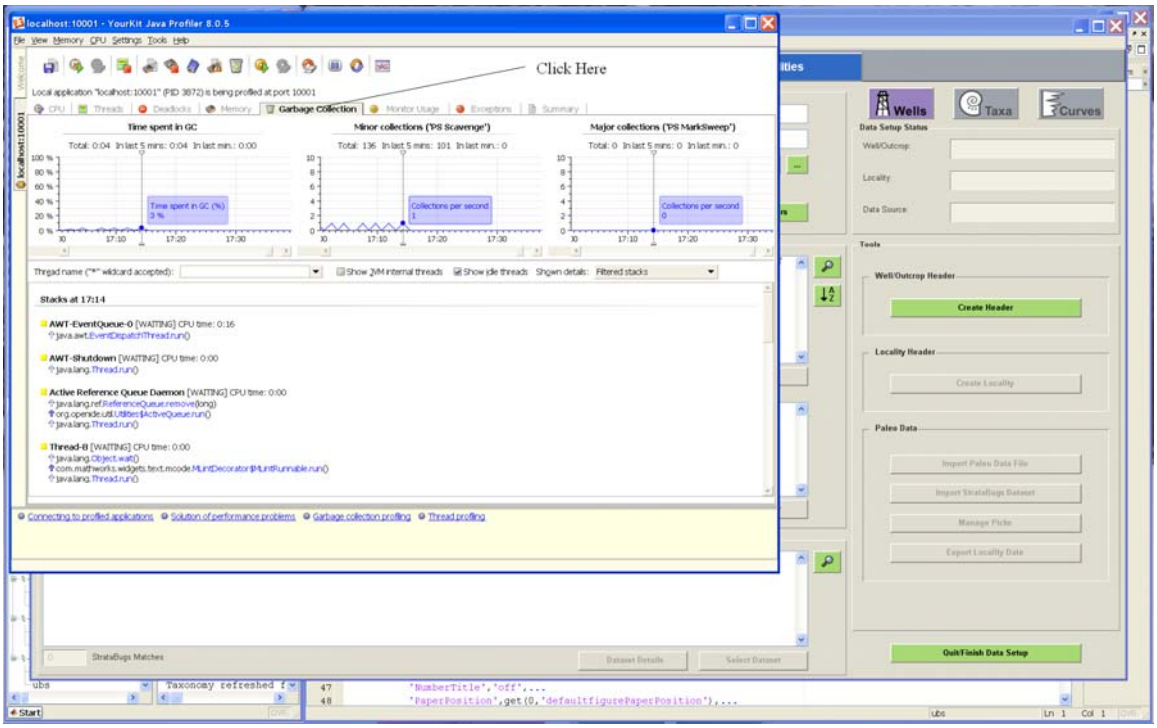


This shows the current object allocations (“char[]” objects selected before we clicked on the “Allocations” tab) sorted by ‘Recorded Objects’ grouped by the java class. You can see which class/method has created the selected object.

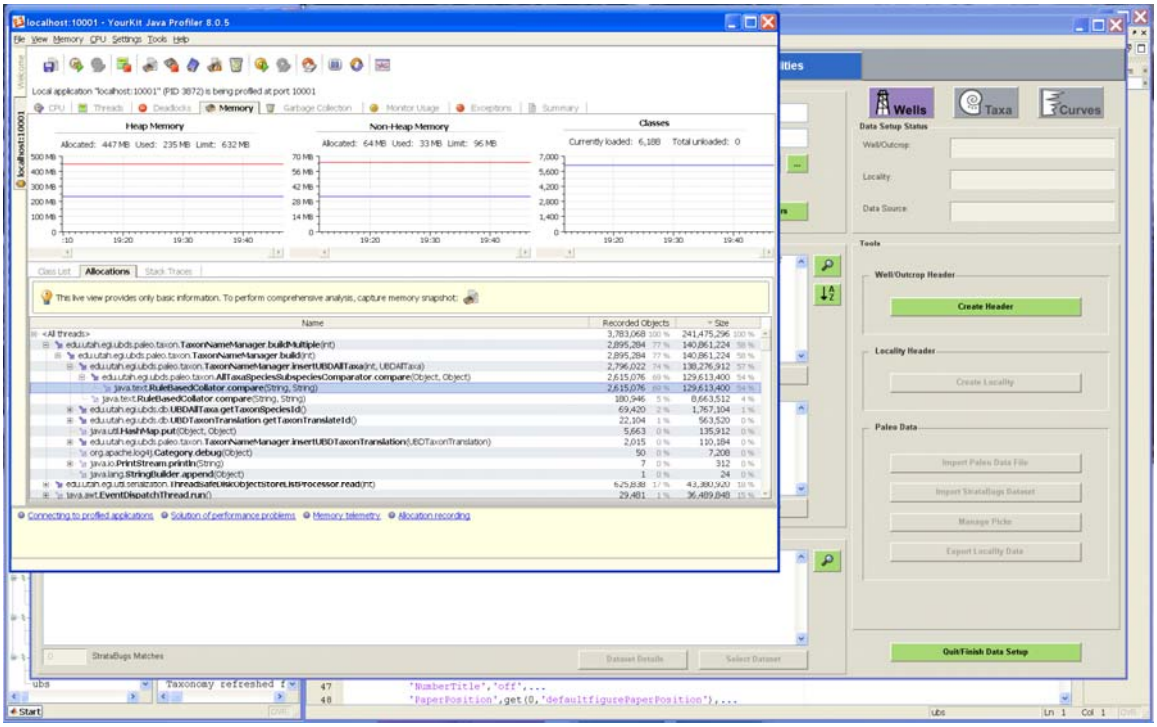


For a more in-depth memory analysis, click the icon in the middle of the screen to the right to capture memory snapshot.

Now, click on the ‘Garbage Collection’ tab on the top and you’ll see…



Here is the garbage collection statistics. Now after my routine has finished running, I clicked on the “Memory” tab on the top…



Here you’ll see most of the objects (in terms of their number and their size in bytes) created comes from the method java.text.RuleBasedCollator.compateTo().

I hope this little tutorial shows you what you can do with YourKit java profiler hooking into the JVM launched by MATLAB.

Note: Java profilers are far from perfect; they freeze at times and failed to update displays. Unless you have a beefy machine to profile your application, I think (although I haven't proved it myself) the best way is to perform profiling on a machine attaching to a remote machine running your application.