Step-by-step ProActive debugging

Arnaud Contes: *arnaud.contes@activeeon.com*

Maxime Menant: *maxime.menant@activeeon.com*

Etienne Vallette d'Osia: *etienne.vallette_d_osia@sophia.inria.fr*

Laurent Vanni: *laurent.vanni@sophia.inria.fr*

Debugging a ProActive application : Outline

- Context
- Solution:
  - StepByStep
  - Connecting debuggers to JVMs
- Demonstration
- Thereafter

Context

Context

1 - How are you debugging your ProActive application ?

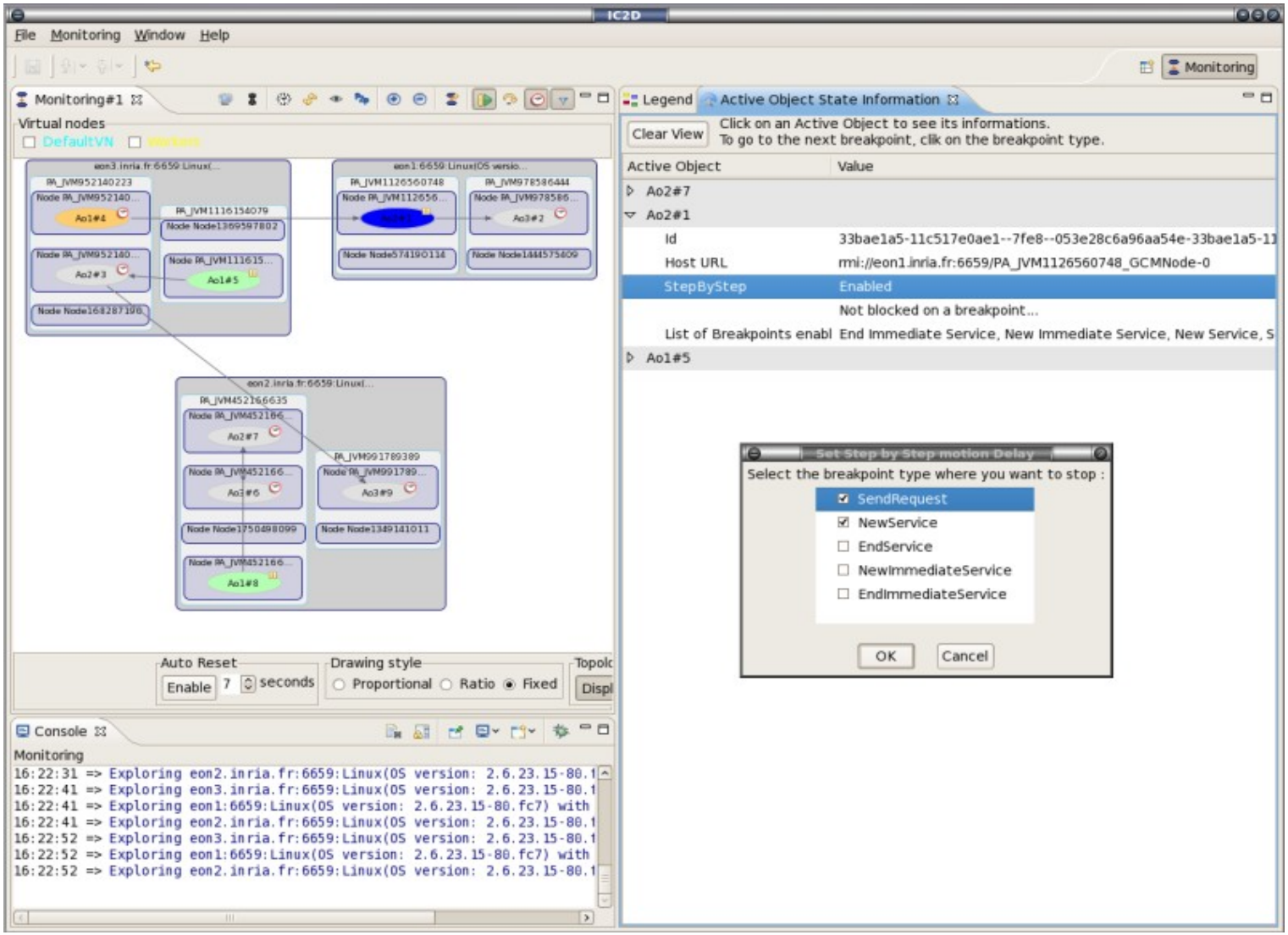
System.out.println() ?

2 - How to connect a debugger to a remote Runtime ?

Ssh tunneling, port forwarding, ... Complex, isn't it ?

Solution: Step-by-step

Step-by-step: controlling application at service level



Step-by-step: Controls

Differents levels of selection for Active Object present in:

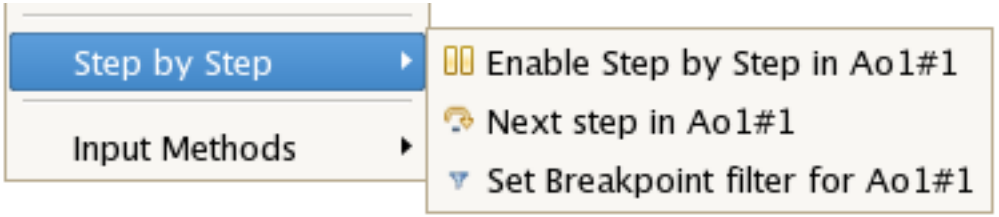
world / host / jvm / node

With the following features:

- Pause / Resume
- Next Step
- Slow Motion
- Filters

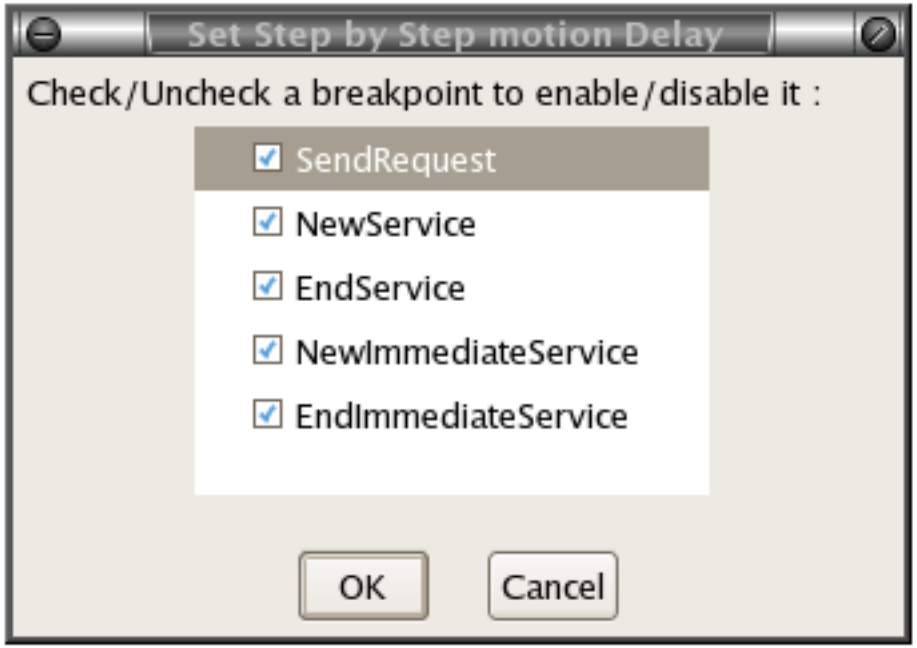
Toolbar for controlling application at world level:

Contextual menu for the others:



Step-by-step: Breakpoints overview

There are different kinds of breakpoints at service level.

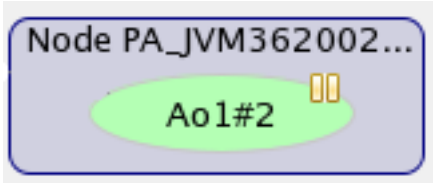


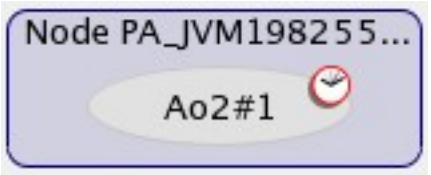
You can enable or disable each breakpoints throught this filters view.

Step-by-step: Visual feedback

Some flags are used to show the state of an Active Object durring the step by step:

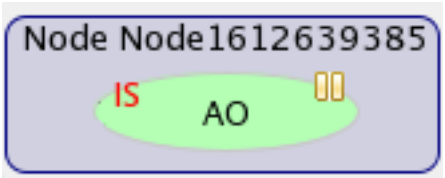
Pause



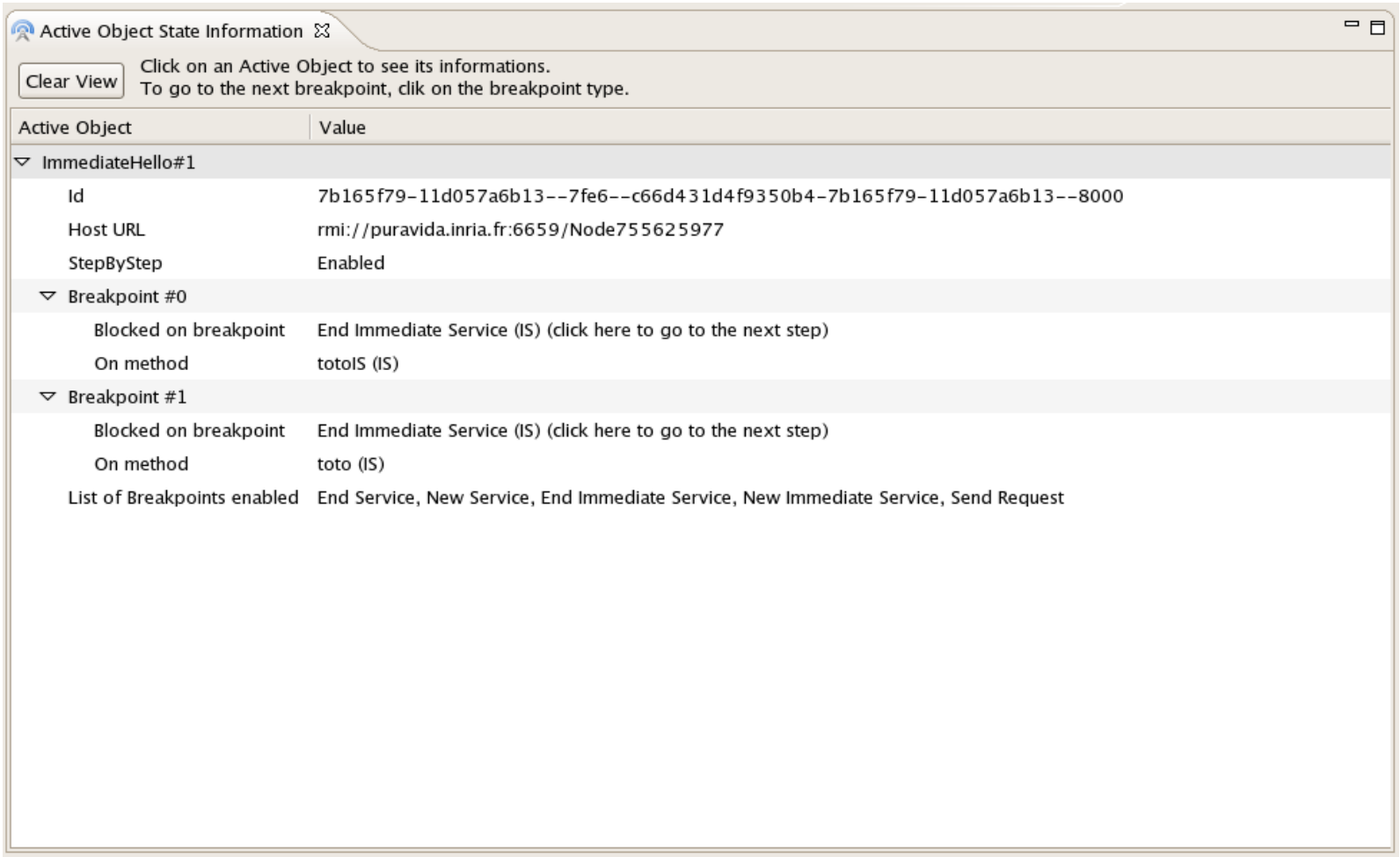


Slow Motion

Immediate Service



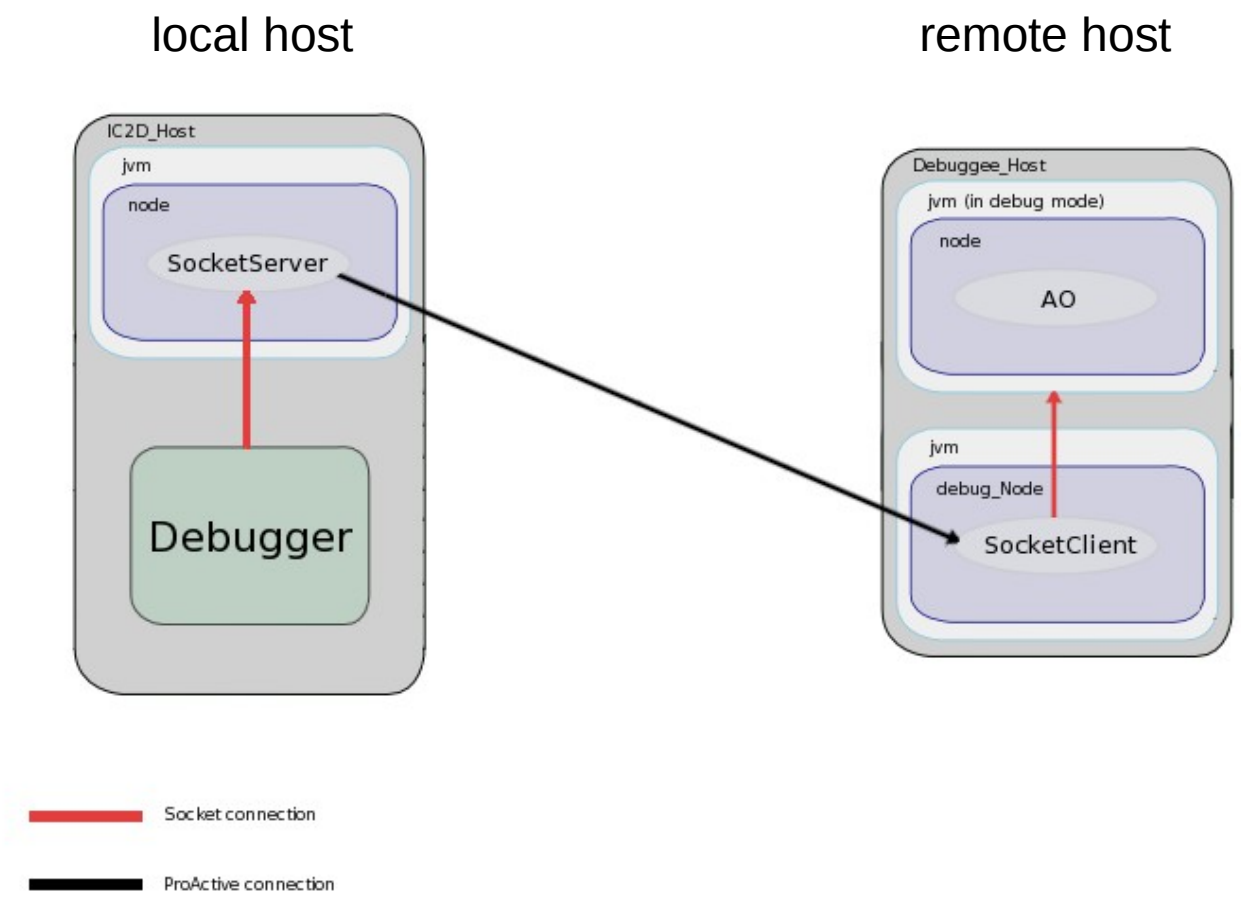
Step-by-step: Informations



Solution: Debugger connection

Debugger connection:

Connecting remote JVM to a local debugger using ProActive connection

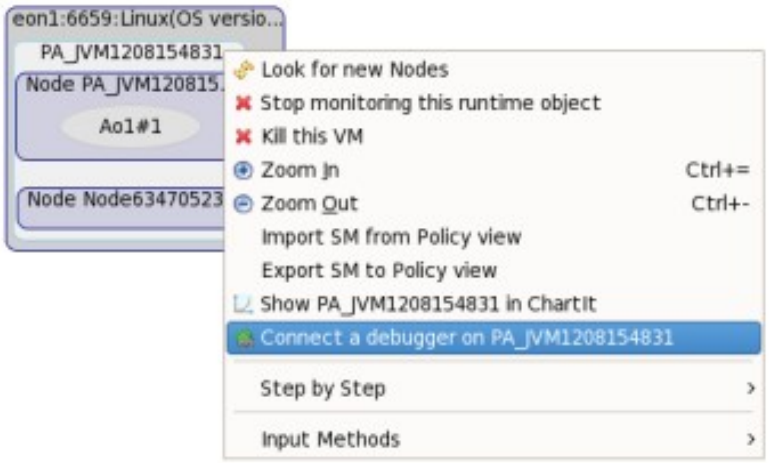


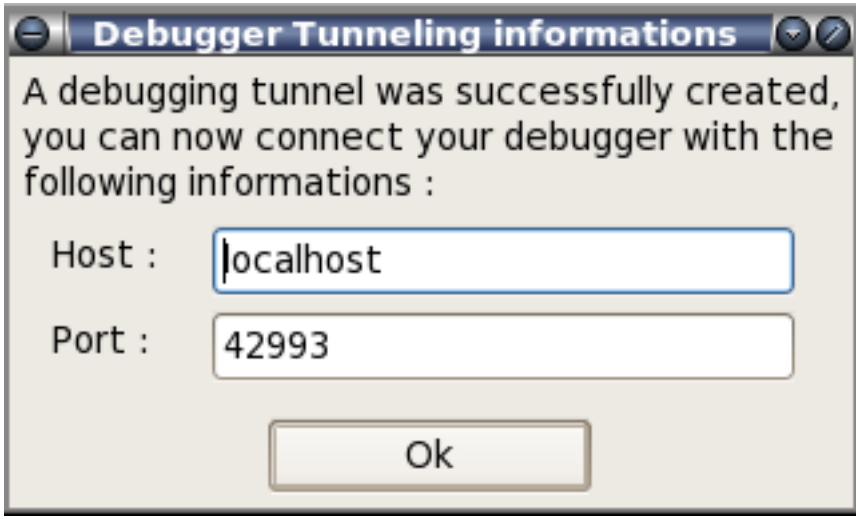
Debugger connection: Usage

First, enable the debug mode for all the JVMs used in the application within the XML deployement file by adding one line in the configuration of the application.

Debugger connection: Usage

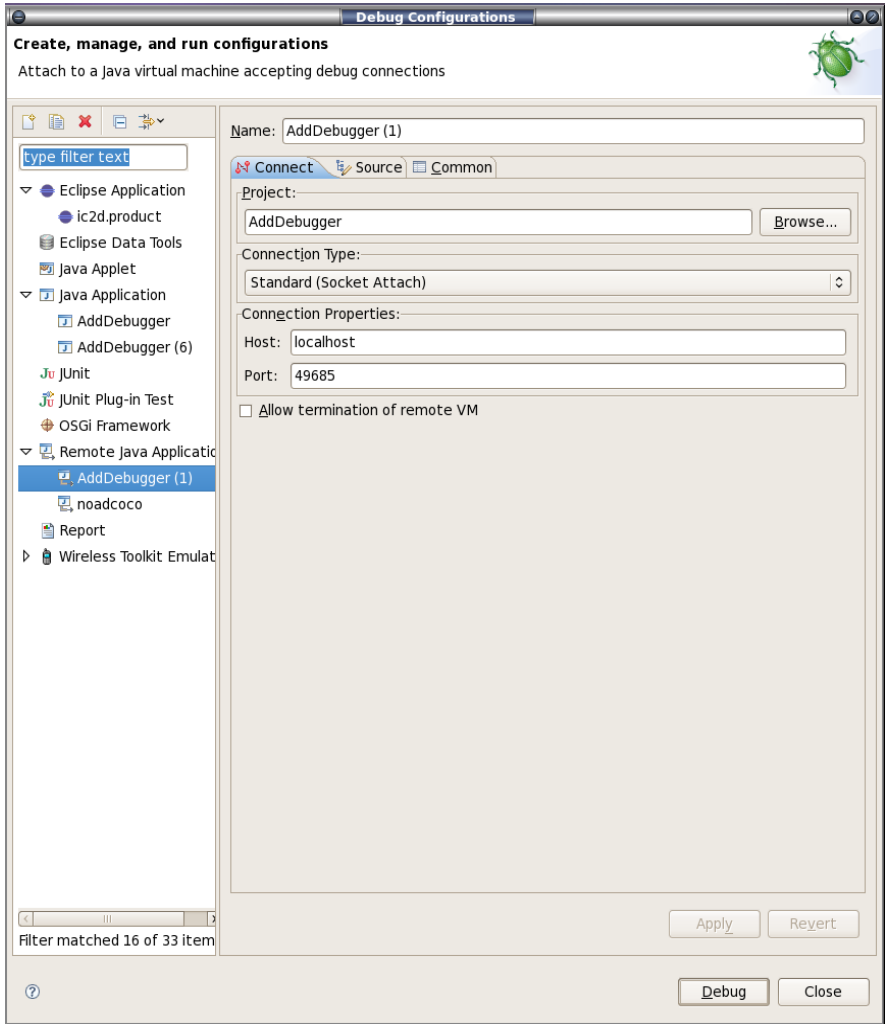
Then, select the JVM that you whant to debug

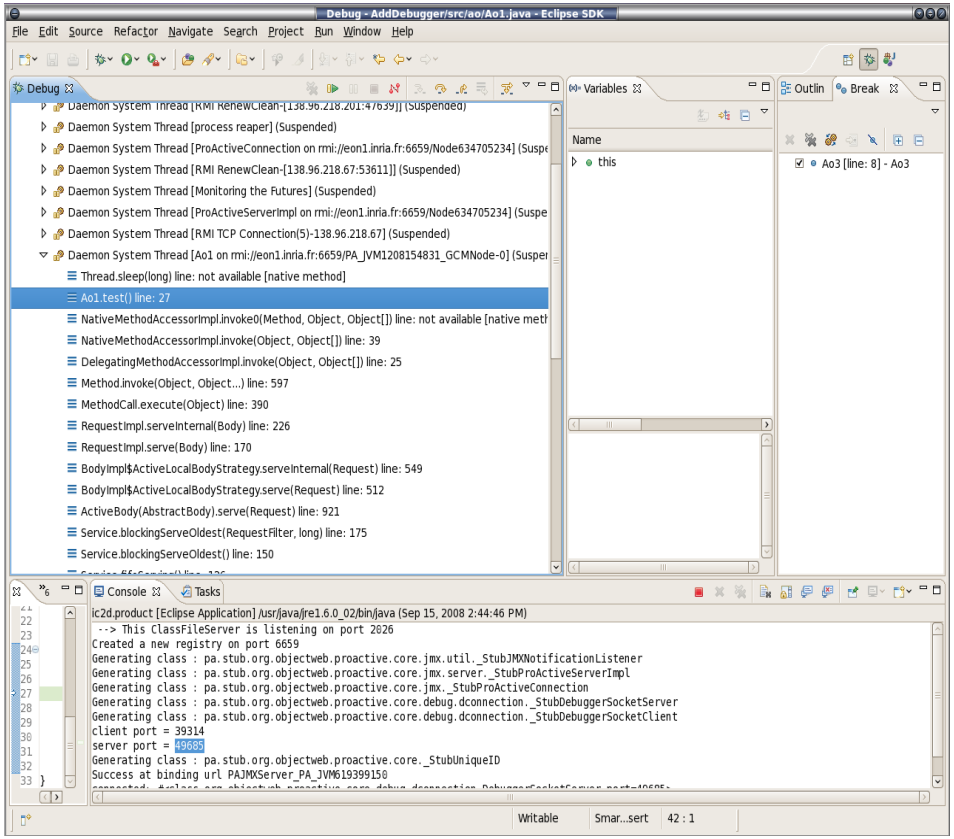




And connect your debugger to localhost with the given port...

Debugger connection: Example





Demonstration

Conclusion: Actual limitations

- Step-by-step
  - Breakpoints are parts of code
  - Partial control of the application (act only with services)
- Debugger connection
  - One debugger per JVM at the same time
  - Local binding performed by the user

Conclusion: Thereafter

The enhancement of ProActive debugging features :

- Distributed Debugging
  - A single interface to debug distributed JVMs (place breakpoint, etc.)
- Replay From Checkpointing
  - Including some work to upgrade if necessary the Fault Tolerance mechanism: Checkpoint Server etc. (stable memory, redundancy).
- Distributed Service Debugging
  - Capacity to view and breakpoint the flow of a given service, and consequent asynchronous calls.

Thanks for your attention!

Any Questions...