Tests

by Paul Hammant

1. Introduction

There are a number of examples that come with AltRMI. They are only present in the source download or the CVS depot, so we will assume that you have one or the other of these.

TODO: This entire page is out of date.

2. Tests using AltRMI

The tests all run under Ant control. Some tests are client and server, others are in a single VM. You may need two command shells for the client/server tests.

The majority of the tests transfer a primary interface, TestInterface, between server and client. It has a number of methods that test the passing of primatives and objects as parameters and return types. Apart from this testing of features, the speed of the transport type is tested. This is simply the counting of as many repetetive invocations of the same method in ten seconds as possible. It is used for an statistically incorrect comparison of transports.

2.1. ObjectStream Over Plain Sockets

The ObjectStream over plain sockets tests are launched from a build file called socketa.xml. You need two command shells. In the first launch **ant -buildfile socketa.xml server**, and in the second **ant -buildfile socketa.xml client**

2.2. CustomStream Over Plain Sockets

The CustomStream over plain sockets tests are launched from a build file called socketb.xml. You need two command shells. In the first launch **ant -buildfile socketb.xml server**, and in the second **ant -buildfile socketb.xml client**

2.3. CustomStream Over Plain Sockets, using callback handlers

The CustomStream over plain sockets tests are launched from a build file called socketc.xml. You need two command shells. In the first launch **ant -buildfile socketc.xml server**, and in the second **ant -buildfile socketc.xml client**. The callback capable layer is not used to its fullest capacity, in that no callbacks ae setup. This is most useful for a comparative speed test.

2.4. RMI

The RMI tests are launched from a build file called rmi.xml. You need two command shells. In the first launch **ant -buildfile rmi.xml server**, and in the second **ant -buildfile rmi.xml client**.

2.5. Piped

The Piped tests are launched from a build file called piped.xml. You need a single shell. To test the piped transport with generated proxies already in the client's classloader, launch **ant -buildfile piped.xml clientclasses**. To test the piped transport with generated proxies retrieved from the server by the client, launch **ant -buildfile piped.xml serverclasses**. To test the piped transport with generated proxies generated on demand by the server for the client, launch **ant -buildfile piped.xml dynamicclasses**.

2.6. Direct

The Direct tests are launched from a build file called direct.xml. You need a single shell. To test the direct connection of client and server launch **ant -buildfile direct.xml direct**. To test the direct connection of client and server with marshalling of communications launch **ant -buildfile direct.xml direct-marshalled**.

3. Tests not using AltRMI

These tests are used for a speed comparison of native Java techniques. This type of testing is possible because AltRMI uses normal Java interfaces. All are run from the proxies.xml Ant script in a single command shell.

3.1. Dynamic Proxy

This test uses dynamically generated proxies, which are normally used when you want to do implemetation hiding. To test, launch **ant -buildfile proxies.xml dynamic-proxy**.

3.2. Coded Proxy

This test uses human crafted proxy. To test, launch **ant -buildfile proxies.xml coded-proxy**.

3.3. Non Proxy

This test directly wires the server to the clint via TestInterface. To test, launch **ant -buildfile proxies.xml un-proxy**. This, of course, is the fastest possible connection of client and server.