Monitors

by Paul Hammant

1. Introduction

Both the server and client sides of AltRMI can be monitored. After instantiation, the Server of Factory have a setMonitor() methods.

2. Server Monitor

The interface for the ServerMonitor is :

```
public interface ServerMonitor {
    void closeError(Class clazz, String s, IOException e);
    void badConnection(Class clazz, String s, BadConnectionException bce);
    void classNotFound(Class clazz, ClassNotFoundException e);
    void unexpectedException(Class clazz, String s, Exception e);
    void stopServerError(Class clazz, String s, Exception e);
}
```

You get to choose from a number of implementations. NullServerMonitor consumes all monitored events. LogEnabledServerMonitor, CommonsLoggingServerMonitor and Log4JServerMonitor route through to the appropriate logging framework. You do not have to tie the application you develop (that needs to use AltRMI) to any particular logging framework. If you so desire, you do not need any logging jar in your classpath (or classloader tree for more complex deployments).

3. Client Monitor

The interface for the ClientMonitor is :

```
public interface ClientMonitor
{
    void methodCalled(Class clazz, String methodSignature, long duration, String annota
    boolean methodLogging();
    void serviceSuspended(Class clazz, Request altrmiRequest, int attempt, int suggeste
    void serviceAbend(Class clazz, int attempt, IOException cause);
    void invocationFailure(Class clazz, String name, InvocationException ie);
    void unexpectedClosedConnection(Class clazz, String name, ConnectionClosedException
    void unexpectedInterruption(Class clazz, String name, InterruptedException ie);
}
```

The ClientMonitor has a couple of novel feautes over ServerMonitor (which just listens). The first is that timings for method calls can be reported on. As timing costs time, the ClientMonitor reports whether it wants timing at all. The second is that serviceSuspend() and serviceAbend() encourages the implementor to join in with whether the pending request will fail or try again. It does this by throwing InvocationException. Different strategies (fail-fast, retry-forever) are possible, but clearly they affect the way client code works.

As with the ServerMonitor, you get to choose from a number of implementations. DumbClientMonitor consumes all monitored events without logging anything and fails-fast for the two abend() and suspend() methods. DefaultClientMonitor, still logs nothing, but tries for a few attempts to reestablish a connection.

Copyright (c) @year@ The Apache Incubator Project. All rights reserved. $Revision: 1.2 $ $Date: 2003/02/16 21:41:35 $