Towards Refactoring the DMF to Support Jini and JMS DMS in GIPSY

Yi Ji, Serguei A. Mokhov, and Joey Paquet
Computer Science and Software Engineering
Concordia University, Montreal, QC, Canada
{ji_yi,mokhov,paquet}@cse.concordia.ca

Abstract

We report on our re-engineering effort to refactor and unify two somewhat disjoint DMS (Demand Migration System) implementations – Jini and JMS by refactoring their parent framework – the DMF (Demand Migration Framework), within the General Intensional Programming System (GIPSY).

1 Introduction

The founding works prior this one that we base our integration effort on are cited here [14, 21, 20, 19, 15, 16, 17, 6, 13, 12]. They cover the initial design, and proof-of-concept implementations of the DMF – Jini- and JMS-based as well as the surrounding integration effort of them in to the General Intensional Programming System (GIPSY)'s run-time environment – the General Eduction Engine (GEE).

This document is structured as follows: first we briefly cite the middleware distributed technologies, such as JMS and Jini, the objectives of this work in Section 2.1, we then present the methodology and the approach in Section 2, and finally we conclude in Section 3.

1.1 Jini

The related work items discussing Jini aspects are covered in [1 4 5]. In Figure 1 we present.

1.1.1 Jini features

- JavaSpace [10]:
  - Object Space
  - Query through Entry fields
  - Middleware and/or Demand Store?
    * Currently more like a middleware: the access is not managed: i.e. demands are written into the space without checking if the corresponding results exist.
    * Accessed through DemandDispatcher
1.2 JMS

Related work on JMS is covered in works such as \[2, 9, 7\]. In Figure 2, we present our approach to the JMS use and unification within the project since the work of Pourteymour \[17\].
1.2.1 JMS features

- A middleware:
  - Senders deposit and receivers acquire.
- Similar to an Object Space, but:
  - Query (filter) is based on Message Properties, not the Entry’s properties.
  - Receivers get the first messages (if no filter applied).

2 Methodology

2.1 Objectives

- Switch the roles of Demand Dispatcher and the Transport Agent.

2.2 Approach: Making Jini and JMS similar

- If JavaSpace [10] is a Demand Store, then the DemandDispatcher can be viewed as Database Access Object.
- Make a Jini-like JMS DMS.
- Substitute the Jini DMS with JMS DMS in the Jini simulator.

3 Conclusion

We have successfully did the POC integration of the two middleware technologies implementations based on Jini and JMS available to the GIPSY run-time system. In the future work we plan to continue refactoring and cleaning up the other technologies within GIPSY to work together in unison. In Appendix A are step-wise instructions for running Jini.

References


A How to run the Jini DMS in the GIPSY project

This document introduces the steps to compile and run the Jini code in the GIPSY project in Windows XP. To use this guide, readers are required to have basic Java and Eclipse experience, basic Jini knowledge (for example, service, lookup, JavaSpace, etc), and the basic understanding of the GIPSY project structure.

1. Get and install the appropriate software, and import the GIPSY project [11].
   NOTE: If the software below cannot be found online, please check the GIPSY tools repository
   (a) JDK 6 update 14 or later (http://java.sun.com/javase/downloads/index.jsp). It is better to set the JAVA_HOME environment variable.
   (b) Eclipse IDE for Java Developers [3]. Unpack the IDE, open it and import the GIPSY project from the GIPSY CVS into the Eclipse.
   (c) Jini Technology Starter Kit v2.1 [8]. The installation should require no administrator accounts. The term JINI_HOME would be used in this manual to refer the installation directory.

2. Compile the Jini code of the GIPSY project NOTE: Point 1 and 2 should be done already in the GIPSY project when you get it.
   (a) Copy the jini-core.jar and jini-ext.jar from JINI_HOME/lib into the gipsy/lib
   (b) Configure the project Build Path by adding the above jars inside lib through “Add JARs”
   (c) Build the project automatically or manually.

3. Start the Jini service
   (a) Go to JINI_HOME/installverify and double-click the Launch All shortcut. The shortcut should launch a service window and a Service Browser window.
(b) In the Service Browser window, click the “Registrar”, and select the registrar representing your computer. Then there would be 6 services appear in the “Matching Services” area, including JavaSpace05, LookupDiscoveryService, ServiceRegistrar and TransactionManager.

(c) Leave the two windows open and do not touch them unless you want to shut down all the services.

4. Run the code requiring only JavaSpace. NOTE: Currently there are two groups of Jini scenarios. The first one consists of DemandDispatcher, DemandDispatcherClient and the DemandDispatcherAgent under the gipsy.GEE.IDP package. The second scenario consists of the gipsy.GEE.IDP.DemandDispatcher, and the DGT class in the gipsy.GEE.IDP.DemandGenerator.simulator package, and the Worker class in the gipsy.GEE.IDP.DemandGenerator.simulator.jini package.

(a) Open Eclipse and run the classes within the same groups mentioned above with the main() method.

5. Run the code requiring both JavaSpace and RMI

(a) Use command window to go to the gipsy/bin directory.

(b) Open the startJiniHTTPServer.bat in edit mode, and check if all the paths are correct.

(c) Double-click the startJiniHTTPServer.bat

(d) Double-click the startJiniRMID.bat

(e) Open Eclipse and run the gipsy.GEE.IDP.DemandGenerator.jini.rmi.JINITransportAgent, and the gipsy.GEE.IDP.DemandGenerator.simulator.jini.WorkerJTA, and the DGT in the gipsy.GEE.IDP.DemandGenerator.simulator package.

Note:
Please make sure that the startJiniHTTPServer.bat and the startJiniRMID.bat are in the gipsy/bin folder. If they are missing, please refer to the following content. Please make sure the settings in the content are consistent with your own JRE directories as shown in Figure 3 and Figure 4.

```
set RUNTIME_JAR="D:\Program Files\Java\jre6\lib\rt.jar"
set JINIHOME_BACKSLASH="D:\Program Files\jini2_1"
set JINI_CLASSPATH=.;%RUNTIME_JAR%;%JINIHOME_BACKSLASH%\lib\jini-core.jar;%JINIHOME_BACKSLASH%\lib-dl\reggie.jar;%JINIHOME_BACKSLASH%\lib-dl\outrigger.jar;%JINIHOME_BACKSLASH%\lib-dl\outrigger-dl.jar;%JINIHOME_BACKSLASH%\lib\tools.jar;%JINIHOME_BACKSLASH%\lib\sun-util.jar;
java -jar -classpath %JINI_CLASSPATH% %JINIHOME_BACKSLASH%\lib\tools.jar -port 8085 -dir . -verbose
```

Figure 3: startHTTPServer.bat
rmid -J-Djava.security.policy=gipsy/GEE/IDP/config/jini.policy

Figure 4: startJiniRMID.bat
Index

API
DemandDispatcher, 1, 3, 6
DemandDispatcherAgent, 6
DemandDispatcherClient, 6
DGT, 6
Entry, 1, 3
gipsy.GEE.IDP, 6
gipsy.GEE.IDP.DemandDispatcher, 6
gipsy.GEE.IDP.DemandGenerator.jini.rmi.JINITransportAgent, 6
gipsy.GEE.IDP.DemandGenerator.simulator, 6
gipsy.GEE.IDP.DemandGenerator.simulator.jini, 6
gipsy.GEE.IDP.DemandGenerator.simulator.jini.WorkerJTA, 6
JAVA_HOME, 5
JINI_HOME, 5
main(), 6
Worker, 6

DMF, 1
DMS, 1, 3

Files
  gipsy/bin, 6
  gipsy/lib, 5
  jini-core.jar, 5
  jini-ext.jar, 5
  JINI_HOME/installverify, 5
  JINI_HOME/lib, 5
  startHTTPServer.bat, 6
  start.JiniHTTPServer.bat, 6
  start.JiniRMI.bat, 6, 7

Frameworks
  DMF, 1
  GEE, 1
  GIPSY, 1, 5
  Jini, 1, 3, 5
  JMS, 1, 3