JADE Java Agent DEvelopment Framework

Ricardo J. Rabelo rabelo@das.ufsc.br Saulo Popov Zambiasi popov@gsigma.ufsc.br

Sumário

Parte 1 – Introdução

- Pré-requisitos
- Download
- Instalação e Configuração
- Interface Gráfica

Parte 2 – Utilização

- Prog. Hello World
- Eclipse
- Comunicação entre Agentes
- Agentes Distribuídos
- Integração com Jess
- Referências



Parte 1 – Introdução

(Pré-requisitos, download, instalação e configuração, interface gráfica)

Introdução

- Java Agent DEvelopment Framework.
- Framework implementado em Java.
- Simplifica a implementação de sistemas multiagentes.
- Utiliza um *middleware* e ferramentas gráficas que suportam debug e fases de desenvolvimento.
- Suporta mobilidade de agentes

Introdução

- A plataforma de agentes pode estar distribuída em diferentes máquinas (sem ser necessário o mesmo SO) e as configurações podem ser controladas via uma GUI remota.
- A configuração pode ser alterada em em tempo de execução, movendo agentes de uma máquina a outra, quando necessário.

Introdução

- JADE é free software (?), com copyright opensource LGPL desde Maio de 2003.
- Atuais membros do Projeto JADE:
 - Telecom Italia;
 - Motorola;
 - Whitestein Technologies AG;
 - Profactor GmbH;
 - France Telecom R&D.
- A última versão: JADE 4.0.1 (07/07/2010).

Pré-requisitos

- Para trabalhar com Jade, são necessários os seguintes requisitos:
 - Máquina virtual java 1.4 ou mais recente;
 - Kit de desenvolvimento JDK 1.4 ou mais recente.
 - A pasta *bin* do JDK deve ser incluida no *PATH* do sistema operacional;
 - Um editor de texto ou IDE para desenvolvimento (Notepad, gedit, Eclipse, Netbeans);
 - Download do JADE (cadastro gratuito)

http://jade.tilab.com

Download

File	~ File size	Description of the content	
jadeAll.zip	8.6 MB	This file contains all JADE, i.e. it is just composed of the 4 files below. If it is too large for downloading, the 4 files below might be downloaded instead.	
jadeBin.zip	2.0 MB	This file contains JADE already compiled and ready to be used, i.e. a set of JAVA archive JAR files.	
jadeDoc.zip	4.7 MB	This file contains all the JADE documentation included the Administrator's Guide and and the Programmer's Guide. NOTICE THAT all the documentation is also available on-line.	
jadeSrc.zip	jadeSrc.zip 1.8 MB This file contains all the JADE source code.		
jadeExamples.zip	270 KB	This file contains the source code of the examples and a simple demo. All the examples and demo must be compiled.	

Instalação e Configuração

 Descompactar o arquivo jadeAll.zip e descompactar cada um dos arquivos em um diretório chamado jade. Exemplo:

Windows: c:\jade

Linux: /home/usuario/jade

Instalação e Configuração

- No Windows, acrescentar os seguintes arquivos no CLASSPATH:
 - c:\jade\lib\jade.jar;
 - c:\jade\lib\jadeTools.jar;
 - c:\jade\lib\Base64.jar;
 - c:\jade\lib\http.jar;
 - c:\jade\lib\iiop.jar.

Instalação e Configuração

 No Linux, acrescentar as seguintes linhas no final do arquivo .bashrc:

```
export JADE_LIB="${HOME}/jade/lib"
export CLASSPATH=".:${JADE_LIB}/jade.jar:${JADE_LIB}/iiop.jar\
:${JADE_LIB}/http.jar\
:${JADE_LIB}/Base64.jar:${JADE_LIB}/jadeTools.jar"
```

Interface Gráfica

 Pelo prompt de comando Executar: *java jade.Boot -gui -local-host localhost*

\varTheta 🔿 🕙 RMA@isabeaut:1099/JADE - JAD	E Remote	Agent Managen	nent GUI 🗆
File Actions Tools Remote Platforms Help			
		💈 🗈 🍼 🐞 🚆	
🕈 🛅 AgentPlatforms	name	addresses state	e owner
🛉 🛅 "isabeaut:1099/JADE"		ADDRESSES STATE	OWNER
🕈 🗖 Main-Container			
– 📴 RMA@isabeaut:1099/JADE			
– 📴 ams@isabeaut:1099/JADE			
– 🙂 df@isabeaut:1099/JADE			
	S]		



Parte 2 – Utilização

Prog. Hello World, eclipse, comunicação entre agentes, agentes distribuídos, referências

Programa HelloWorld

```
import jade.core.Agent;
public class HelloWorld extends Agent {
    protected void setup() {
        System.out.println("Hello World! ");
        System.out.println("My name is: " + getLocalName());
    }
}
```

Compilar e Executar

Compilar:

javac HelloWorld.java **Executar:**

> // com a interface gráfica rodando java jade.Boot -<u>container</u> suzumiya:HelloWorld

// sem a interface gráfica rodando java jade.Boot suzumiya:HelloWorld

Eclipse – Build Path... Libraries...

ſ	0	Properties for HelloWorld	₽
1	type filter text	Java Build Path	\$\$ • \$\$ • \$
	ResourceBeanInfo PathBuildersJava Build Path▷Java Code Style▷Java Compiler▷Java EditorJavadoc LocationProject ReferencesRun/Debug Settings▷Task RepositoryTask Tags▷Validation	Source Projects Libraries Order and Export JARs and class folders on the build path: commons-codec-1.3.jar - /home/saulo/bin/jade/lib/commons-codec http.jar - /home/saulo/bin/jade/lib iop.jar - /home/saulo/bin/jade/lib jade.jar - /home/saulo/bin/jade/lib jade.jar	Add JARs Add External JARs Add Variable Add Libr <u>a</u> ry Add Class Folder Add External Class <u>F</u> older <u>E</u> dit <u>R</u> emove <u>Migrate JAR File</u>
	0		OK Cancel

Eclipse – Run Configuration: Main

0	Run Configurations	\bigcirc
Create, manage, and run co	nfigurations	
Run a Java application		
📑 🗈 🗶 🖻 🐎 🔹	Name: HelloWorld	
type filter text		\neg
🗄 Apache Tomcat	Broject	
Eclipse Application		
😝 Eclipse Data Tools	Helloworld	21
🗄 Generic Server	Main class:	
🗄 Generic Server(External I	jade.Boot	
HTTP Preview	Enclude system libraries when searching for a main class	
J2EE Preview	Include in <u>h</u> erited mains when searching for a main class	
🖭 Java Applet	St <u>o</u> p in main	
▼ 🗊 Java Application		
HelloWorld		
Ju JUnit		
ັງ _{ບີ} JUnit Plug-in Test		
OSGi Framework		
Jii Task Context Plug-in Test		
Ju Task Context Test		
	Apply Revert	
Filter matched 15 of 21 items		
0	<u>Bun</u> Close	

Eclipse – Run Configuration: Arguments

0	Run Configurations	
Create, manage, and run co	nfigurations	
Run a Java application		
		_
📑 🗎 🗶 🖻 🏇 🖌	Name: HelloWorld	
type filter text		
🗄 Apache Tomcat	G Main (%)= Arguments BRE Classpath Source Convolution Ment Common	
Eclipse Application	-container Gandelf:HelloWorld	
🗑 Eclipse Data Tools		
🗄 Generic Server		
🗄 Generic Server(External I	Variables	2
HTTP Preview	VM arguments:	
J2EE Preview		
题 Java Applet		,
▽ 🗾 Java Application	Variable <u>s</u>	
🗾 HelloWorld	Working directors	2
J <mark>u</mark> JUnit	Default: Efwarkspace locitialleWorld]	
Junit Plug-in Test	peragic. ptworkspace_loc.Helloworld}	- 11
OSGi Framework	Other:	
Jin Task Context Plug-in Test	Workspace) <u>File System</u> Variabl <u>e</u> s	
Ju lask Context lest		
)++	Apply Revert	
Filter matched 15 of 21 items		
Ô		
V	<u>H</u> un Close	

Um Exemplo Mais Real

```
⊖import jade.core.Agent;
 import jade.core.behaviours.*;
 public class MyAgent extends Agent {
     protected void setup() {
Θ
         addBehaviour(new myBehaviour(this));
     }
Θ
     class myBehaviour extends SimpleBehaviour {
         public myBehaviour(Agent a) {
Θ
             super(a);
         }
         public void action() {
Θ
             // local do programa do agente
         }
         private boolean finished = false;
         public boolean done() {
Θ
             return finished;
         }
     }
```

Comunicação: Sender.java

```
import jade.core.Agent;
 import jade.core.AID;
 import jade.core.behaviours.*;
 import jade.lang.acl.*;
 public class Sender extends Agent {
     protected void setup() {
Θ
         addBehaviour(new CyclicBehaviour(this) {
Θ
             public void action() {
                  ACLMessage msg = receive();
                 if (msq != null)
                      System.out.println("== Answer" + " <- " + msg.getContent()</pre>
                              + " from " + msq.qetSender().qetName());
                 block();
             }
         });
         ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
         msg.setContent("Ping");
         msg.addReceiver(new AID("a", AID.ISLOCALNAME));
         send(msq);
     }
```

Comunicação: Receiver.java

```
⊨import jade.core.Agent;
 import jade.core.behaviours.*;
 import jade.lang.acl.*;
 public class Receiver extends Agent {
     protected void setup() {
Θ
         addBehaviour(new CyclicBehaviour(this) {
Θ
Θ
             public void action() {
                 ACLMessage msg = receive();
                 if (msg != null) {
                     System.out.println("== " + myAgent.getLocalName()
                             + " received: " + msg.getContent());
                     ACLMessage reply = msg.createReply();
                      reply.setPerformative(ACLMessage.INFORM);
                      reply.setContent(" Pong");
                     send(reply);
                 }
                 block();
             }
         });
     }
```

Execução

Em uma janela execute o Receiver: java jade.Boot -local-host localhost a:Receiver Em outra janela execute o Sender:

java jade.Boot -local-host localhost -container main:Sender

Observe que como o primeiro comando já executa a plataforma, o segundo apenas acrescenta o Sender nesta, já em execução.

Agentes Distribuídos

Supondo que no computador1 tem como hostname "PC1", digite a seguinte linha de comando para carregar o maincontainer:

java jade.Boot -local-host localhost -gui

Execute a seguinte linha de comando em uma outra máquina a qual criará outro container de agentes e fará com que esse container se conecte ao main container no computador1 PC1:

java jade.Boot -host PC1 -container a:Receiver

Onde "PC1" é o host da máquina onde se encontra o maincontainer, "a" é o nome do agente e "Receiver" é o código que implementa o agente.

Agentes Distribuídos

Execute agora em uma terceira máquina o seguinte comando que cria dois agentes:

java jade.Boot -host PC1 -container Main:Sender

Neste caso teremos dois agentes distribuídos:

- Sender;
- Receiver.

Agentes conectados à plataforma JADE remota.

As classes destes agentes já devem estar compiladas.

Integração JADE + JESS

- Download Jess
 - http://www.jessrules.com/
- Instalação em um diretório
- Configuração do Eclipse
- Inserção do código em um agente Jade
- Execução

JADE + JESS Configuração

 No Linux, acrescentar as seguintes linhas no final do arquivo .bashrc:

export JESS_LIB="\${HOME}/bin/Jess/lib" export CLASSPATH="\${CLASSPATH}:\${JESS_LIB}/jess.jar\ :\${JESS_LIB}/jsr94.jar"

Jade & Jess – Configuração Eclipse

		Java Build Path	⇔ ↔ ◄
	Resource	A Source Projects Libraries & Order and Export	
	Builders Java Build Path	JARs and class folders on the build path:	
⊳	Java Code Style	commons-codec-1.3.jar - /home/saulo/bin/jade/lib/com	Add JARs
⊳	Java Compiler Java Editor	 million http.jar - /home/saulo/bin/jade/lib million.jar - /home/saulo/bin/jade/lib 	Add External JARs
ŕ	Javadoc Location	i jade.jar - /home/saulo/bin/jade/lib	Add <u>V</u> ariable
	Project References Run/Debug Settings	ight is a second state of the second state	Add Libr <u>a</u> ry
Þ	Task Repository	 ▷ ➡ JRE System Library [JavaSE-1.6] 	Add <u>C</u> lass Folder
Þ	Task Tags Validation		Add External Class <u>F</u> older
ſ	WikiText		<u>E</u> dit
			<u>R</u> emove
			Migrate JAR File
		III	
(?		Cancel OK

Jade & Jess – hello.clp e JessJadeAgent.java

```
;; Hello, world in Jess!
```

```
(printout t "Hello, world!" crlf)
```

```
import jess.*;
import jade.core.Agent;
import jade.core.behaviours.*;
```

}

public class JessJadeAgent extends Agent {
 protected void setup() {
 addBehaviour(new Behaviour(this));
 }

Jade & Jess – hello.clp e JessJadeAgent.java

```
class Behaviour extends SimpleBehaviour {
   private boolean finished = false;
    public Behaviour(Agent a) {
        super(a);
    }
    public void action() {
        System.out.println("Agent Name " + myAgent.getLocalName());
        try {
            runJessCode();
        } catch (JessException je) {
            System.out.println("Error: " + je);
        finished = true;
    }
    public void runJessCode() throws JessException {
        Rete engine = new Rete();
        engine.batch("hello.clp");
    }
   public boolean done() {
        return finished;
    }
```

JadeJessExpert - Supplier.java

```
public class Supplier {
   public String name;
   public int value;
    public int trust;
   public int quantity;
    public Supplier(String n, int v, int t, int q) {
        name = n;
        value = v;
        trust = t;
        quantity = q;
    }
   public String toString() {
        return name + "\t" + value + "\t" + trust + "\t" + quantity;
    }
```

JadeJessExpert.java

```
import java.util.Vector;
 import jade.core.Agent;
 import jade.core.behaviours.*;
 public class JadeJessExpert extends Agent {
     protected void setup() {
Θ
         addBehaviour(new Behaviour(this));
     }
 }
 class Behaviour extends SimpleBehaviour {
     private boolean finished = false;
     public Behaviour(Agent a) {
Θ
         super(a);
     }
     public void action() {
\oplus
     public boolean done() {
Θ
         return finished;
     }
 }
```

JadeJessExpert.java – metodo action()

```
public void action() {
   System.out.println("Agent Name " + myAgent.getLocalName());
   Vector<Supplier> suppliers = new Vector<Supplier>();
    // Name, value, trust, quantity
    suppliers.add(new Supplier("001", 100, 50, 10));
    suppliers.add(new Supplier("002", 105, 70, 90));
    suppliers.add(new Supplier("003", 90, 90, 35));
    suppliers.add(new Supplier("004", 50, 10, 90));
    for (int i = 0; i < suppliers.size(); i++) {</pre>
        System.out.println(suppliers.get(i).toString());
    }
    ExpertBuy eb = new ExpertBuy(suppliers, 40); // required quantity
    if (eb.winner != null) {
        System.out.println("Fornecedor selecionado: " + eb.winner.name);
    } else {
        System.out.println("Nenhum fornecedor selecionado");
    ŀ
    finished = true;
```

}

JadeJessExpert - ExpertBuy.java

```
□import java.util.Vector;
 import jess.*;
 public class ExpertBuy {
     public Supplier winner;
     private Vector<Supplier> suppliers = new Vector<Supplier>();
     private int quantity;
     public ExpertBuy (Vector<Supplier> sup, int qtd) {
Θ
         winner = null;
         suppliers = sup;
         quantity = qtd;
         run();
     }
     public void run() {
\oplus
 }
```

ExpertBuy.java – Metodo run()

```
public void run() {
   try {
        Rete r = new Rete();
        r.clear():
        r.batch("se.clp");
        r.assertString("(quantidade necessaria " + quantity + ")");
        for (int i = 0; i < suppliers.size(); i++) {</pre>
            Supplier s = suppliers.get(i);
            String str =
                "(fornecedor " +
                "(codigo \"" + s.name + "\") " +
                "(preco " + s.value + ") " +
                "(confianca " + s.trust + ") " +
                "(quantidade " + s.quantity + ") )";
            r.assertString(str);
        }
        r.run();
        Value v = r.getGlobalContext().getVariable("*vencedor*");
        String winnerName = v.toString();
        for (int i = 0; i < suppliers.size(); i++) {</pre>
            Supplier s = suppliers.get(i);
            if (winnerName.contains(s.name)) {
                winner = s;
        }
    } catch (JessException ex) {
        System.err.println(ex);
    }
```

Referências

- Vaucher, J; Ncho, A, Jade Tutorial and Primer http://www.iro.umontreal.ca/~vaucher/Agents/Jade/JadePrimer.html
- Caire, G, Jade Programming For Beginners http://jade.tilab.com/doc/JADEProgramming-Tutorial-for-beginners.pd
- Silva, L A M, Estudo e Desenvolvimento de Sistemas Multiagentes usando JADE: Java Agent Development framework http://jade.tilab.com/papers/2003/monografia.pdf
- Site oficial do Jade http://jade.cselt.it/
- Site oficial do Jess http://www.jessrules.com
- Sun Java JDK http://java.sun.com/javase/downloads/index.jsp
- Site Oficial do Eclipse http://www.eclipse.org/