

Les communications des agents JADE

Par Youghourta BENALI 

Date de publication : 10 août 2009


Après avoir vu **[comment créer un agent JADE](#)** et **[comment définir ses tâches](#)** ; alors pour parler d'un véritable système multi-agents et non pas d'un ensemble d'agents agissant d'une manière individuelle, il est temps de voir comment les agents communiquent et s'échangent les messages. Cet article a comme but de montrer comment les agents JADE s'échangent les messages, et comment créer des conversations entre les agents JADE.

I - Introduction.....	3
II - Format d'un message JADE.....	3
III - L'envoi d'un message.....	3
IV - La réception d'un message.....	3
V - L'attente d'un message.....	4
VI - Choisir un message de la boîte aux lettres.....	4
VII - Exemple.....	4
VIII - Conclusion.....	6
IX - Remerciement.....	7

I - Introduction

Pour que plusieurs agents JADE arrivent à collaborer, ils doivent s'échanger des messages. Chaque agent JADE possède une sorte de boîte aux lettres qui contient les messages qui lui sont envoyés par les autres agents. Ces boîtes aux lettres sont sous forme d'une liste qui contient les messages selon l'ordre chronologique de leur arrivée.

II - Format d'un message JADE

Les agents JADE utilisent des messages conformes aux spécifications de la FIPA ( **FIPA-ACL**). Les messages JADE sont des instances de la classe `ACLMessage` du package `jade.lang.acl`. Ces messages sont composés en général de :

- L'émetteur du message : un champ rempli automatiquement lors de l'envoi d'un message.
- L'ensemble des récepteurs du message : un message peut être envoyé à plusieurs agents simultanément.
- L'acte de communication : qui représente le but de l'envoi du message en cours (informer l'agent récepteur, appel d'offre, réponse à une requête,...)
- Le contenu du message.
- Un ensemble de champs facultatifs, comme la langue utilisée, l'ontologie, le `timeOut`, l'adresse de réponse...

III - L'envoi d'un message

Pour envoyer un message, il suffit de remplir les champs nécessaires (l'ensemble des récepteur et le contenu du message et l'acte de communication) d'un message JADE, puis d'appeler la méthode `send()` de la classe `Agent`. Voici un exemple d'envoi d'un message JADE.

```
ACLMessage message = new ACLMessage(ACLMessage.INFORM);
message.addReceiver(new AID("said", AID.ISLOCALNAME));
message.setContent("bonjour...");
send(message);
```

ce message est envoyé à l'agent appelé Said pour lui dire bonjour.

IV - La réception d'un message

La réception d'un message est aussi simple que l'envoi. Il suffit d'appeler la méthode `receive()` de la classe `Agent` pour récupérer le premier message non encore lu de l'agent.

Exemple :

```
ACLMessage messageRecu = receive();
```

Pour répondre à un message reçu on récupère l'identité de l'émetteur du message par la méthode `getSender()` ; Voici un exemple :

```
ACLMessage message = new ACLMessage(ACLMessage.INFORM);
message.addReceiver(messageRecu.getSender());
message.setContent("la réponse");
send(message);
```

V - L'attente d'un message

Il se peut qu'un agent doive effectuer un certain traitement ou lancer quelques tâches après avoir reçu un message d'un autre agent. Il est possible de faire une attente active jusqu'à l'arrivée du message de la manière suivante :

```
ACLMessage message =null ;
While (message == null){
Message = receive() ;
}
//traitement à faire après avoir reçu le message.
```

Mais ce genre d'attente active consomme énormément les ressources de la machine sur laquelle l'agent s'exécute. On peut aussi bloquer un Behaviour d'un agent jusqu'à la réception du message, et ceci grâce à la méthode block() de la classe Behaviour.

```
Message = receive() ;
If (message == null) block();
//traitement à faire après avoir reçu le message.
```

Le Behaviour se bloque jusqu'à la réception d'un message.



A la réception d'un message tous les Behaviour bloqués de l'agent continuent leur exécution.

VI - Choisir un message de la boîte aux lettres

Lorsqu'un agent communique simultanément avec plusieurs agents, il peut décider un moment de ne lire que les messages de la provenance d'un agent particulier ou les messages ayant un acte de communication particulier. Pour cela un agent peut définir un modèle de message à recevoir.

Exemple :

```
MessageTemplate modele = MessageTemplate.MatchPerformative(ACLMessage.INFORM) ;
ACLMessage msg = myAgent.receive(modele) ;
```

VII - Exemple

Voici un exemple de communication entre deux agents. Nous avons deux agents : AgentA et AgentB. AgentA produit des nombres aléatoires puis les envoie à AgentB qui cumule ces nombre aléatoires. Tant que la somme ne dépasse pas un certain seuil (50 dans l'exemple), il informe AgentA qu'il est prêt à recevoir d'autres nombres aléatoires. Dès que le seuil est dépassé, AgentB informe AgentA et s'arrête. Dès qu'AgentA reçoit le message d'arrêt, il s'arrête également.

AgentA.java

```
import jade.core.AID;
import jade.core.Agent;
import jade.core.behaviours.FSMBehaviour;
import jade.core.behaviours.OneShotBehaviour;
import jade.lang.acl.ACLMessage;

public class AgentA extends Agent {

protected void setup() {

    System.out.println("-----");
```

AgentA.java

```

System.out.println("-----agent A-----");
System.out.println("-----");

FSMBehaviour agentA_beh= new FSMBehaviour();

agentA_beh.registerFirstState(new attendreAgentB(), "attendreAgentB");
agentA_beh.registerState(new envoiChiffre(), "envoiChiffre");
agentA_beh.registerLastState(new fin(), "fin");

agentA_beh.registerDefaultTransition("attendreAgentB", "envoiChiffre");
agentA_beh.registerTransition("envoiChiffre", "attendreAgentB",0);
agentA_beh.registerTransition("envoiChiffre", "fin", 1);

addBehaviour(agentA_beh);
}

private class attendreAgentB extends OneShotBehaviour{

    @Override
    public void action() {

        System.out.println("en attente de l agent B");
        block();
    }
}
/*****/
private class envoiChiffre extends OneShotBehaviour{

    int valeurRetour = 0;
    @Override
    public void action() {

        ACLMessage messageRecu = receive();
        if (messageRecu.getContent().equalsIgnoreCase("pret") ) valeurRetour=0;
        else    valeurRetour=1;

        int chiffre = (int) (Math.random()*10);
        System.out.println("envoi de la valeur "+ chiffre);
        ACLMessage message = new ACLMessage(ACLMessage.INFORM);
        message.addReceiver(messageRecu.getSender());
        message.setContent(chiffre+"");
        send(message);
    }

    public int onEnd(){
        return valeurRetour;
    }
}
/*****/
private class fin extends OneShotBehaviour{

    @Override
    public void action() {
        System.out.println("arret de l'agent");
        myAgent.delete();
    }
}
}

```

AgentB.java

```

import java.security.acl.Acl;
import jade.core.AID;
import jade.core.Agent;
import jade.core.behaviours.FSMBehaviour;
import jade.core.behaviours.OneShotBehaviour;
import jade.lang.acl.ACLMessage;

public class AgentB extends Agent {

```

AgentB.java

```
int somme = 0;
boolean stop = false;
protected void setup() {
    System.out.println("-----agent B-----");
    FSMBehaviour agentB_beh= new FSMBehaviour();

    agentB_beh.registerFirstState(new attendrechiffre(), "attendrechiffre");
    agentB_beh.registerState(new afficher(), "afficher");
    agentB_beh.registerState(new fin(), "fin");

    agentB_beh.registerTransition("attendrechiffre", "afficher",0);
    agentB_beh.registerTransition("attendrechiffre", "fin",1);
    agentB_beh.registerDefaultTransition("afficher", "attendrechiffre");

    addBehaviour(agentB_beh);
}
private class attendrechiffre extends OneShotBehaviour{

    int valeurRetour = 0;

    public void action() {
        ACLMessage message = new ACLMessage(ACLMessage.INFORM);
        message.addReceiver(new AID("AgentA", AID.ISLOCALNAME));

        if(!stop){
            message.setContent("pret");
            send(message);
            valeurRetour=0;
            block();
        }else{
            message.setContent("arret");
            send(message);
            valeurRetour=1;
        }
    }
    public int onEnd(){
        return valeurRetour;
    }
}
/*****/
private class afficher extends OneShotBehaviour{
    @Override
    public void action() {
        ACLMessage messageRecu = receive();
        somme+= Integer.parseInt(messageRecu.getContent());
        System.out.println("message recu= "+ messageRecu.getContent());
        System.out.println("la somme actuelle = "+somme);
        if (somme > 50) stop = true;
    }
}
/*****/
private class fin extends OneShotBehaviour{
    @Override
    public void action() {
        System.out.println("fin de l'agent");
        myAgent.delete();
    }
}
}
```

VIII - Conclusion

A travers cet article nous avons pu voir les briques de base de la communication des agents JADE. J'espère que cet article va vous aider à créer un véritable système multi-agents.

IX - Remerciement

Je remercie **Baptiste Wicht** pour l'aide et la correction orthographique.