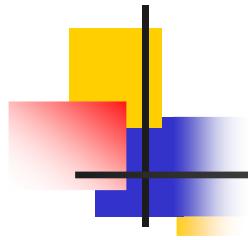


JADE

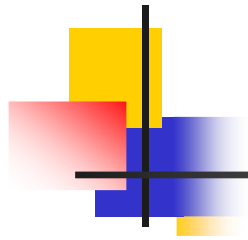


Introducción

JADE (Java Agent DEvelopment Framework) es una plataforma desarrollada íntegramente en Java para la creación de sistemas multi-agente.

Además de proporcionar un API para la creación de agentes y elementos relacionados, pone a nuestra disposición una interfaz gráfica y una serie de herramientas para el control y la depuración de nuestro sistema durante el desarrollo del mismo.

Una de las principales características de la plataforma es que los desarrollos estan dentro del estándar FIPA. El intercambio de mensajes entre agentes, así como las performativas empleadas se corresponderán con lo especificado en este estándar.



Instalación

Al ser una plataforma desarrollada en Java será posible utilizarla en cualquier sistema operativo que disponga de una máquina virtual de Java

La página del proyecto la podemos encontrar en:

<http://sharon.cselt.it/projects/jade>

La última versión oficial de JADE es la 3.7

Se debe añadir a la variable de entorno CLASSPATH los archivos .jar del directorio *lib*, así como el *directorio actual*.

```
set CLASSPATH=%CLASSPATH%.;c:\jade\lib\jade.jar;  
c:\jade\lib\jadeTools.jar; c:\jade\lib\http.jar;  
c:\jade\lib\iiop.jar
```



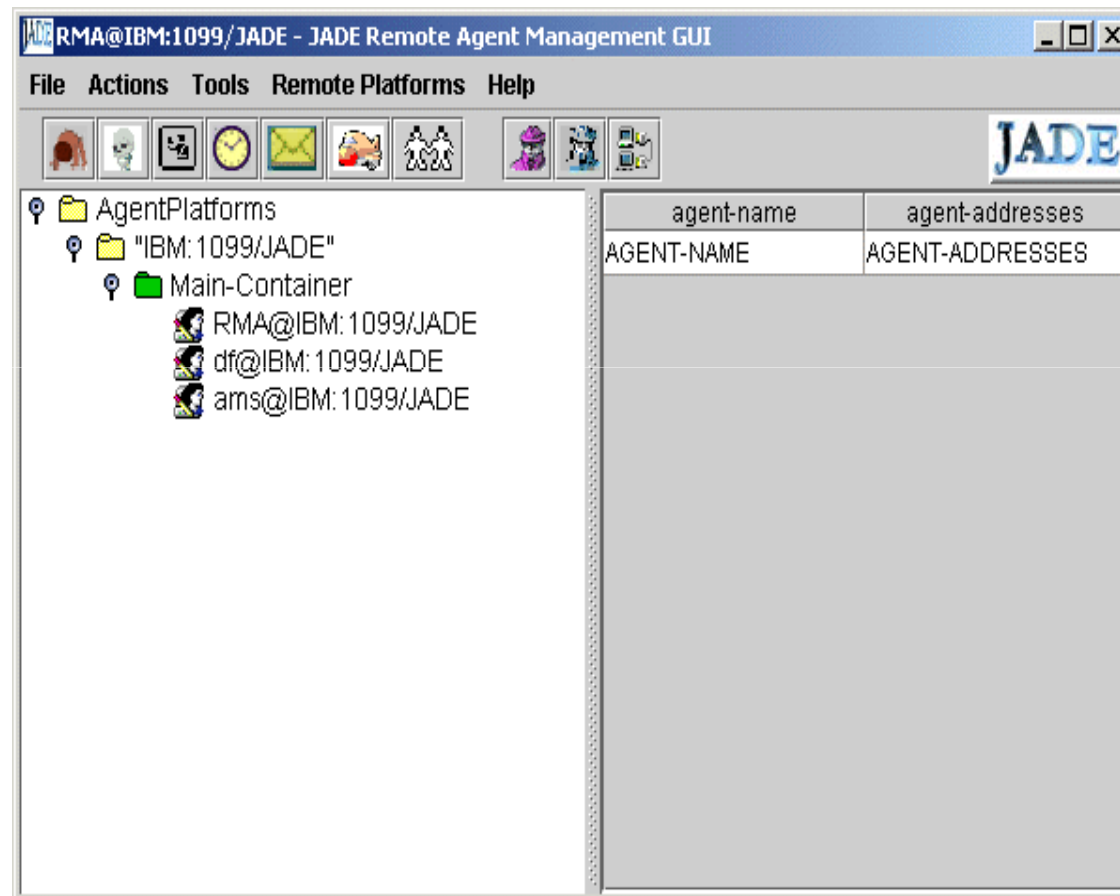
Iniciar Jade

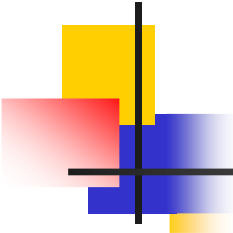
`java jade.Boot [opciones] [Lista de Agentes]`

Opciones principales:

- container (si es distinto del principal)
- host (nombre del host)
- port (puerto, por defecto es el 1099)
- name (nombre simbólico de la plataforma)
- gui (lanza el RMA)
- mtp (permite añadir protocolos externos además del iiop)

Entorno





Entorno

Deberemos tener en cuenta una serie de conceptos:

Las *plataformas de agentes* (AgentPlatforms) se corresponden con máquinas y constan de *contenedores* cuya función es almacenar agentes, no necesariamente en el mismo ordenador. Los agentes utilizan el protocolo RMI para comunicarse entre varios contenedores.

El RMA (*Remote Monitoring Agent*) permite:

- Iniciar, suspender, reiniciar agentes
- Matar agentes o contenedores
- Mandar mensajes
- Clonar agentes
- Añadir o quitar plataformas remotas

muestra la presencia de otros dos agentes en el Main Container.



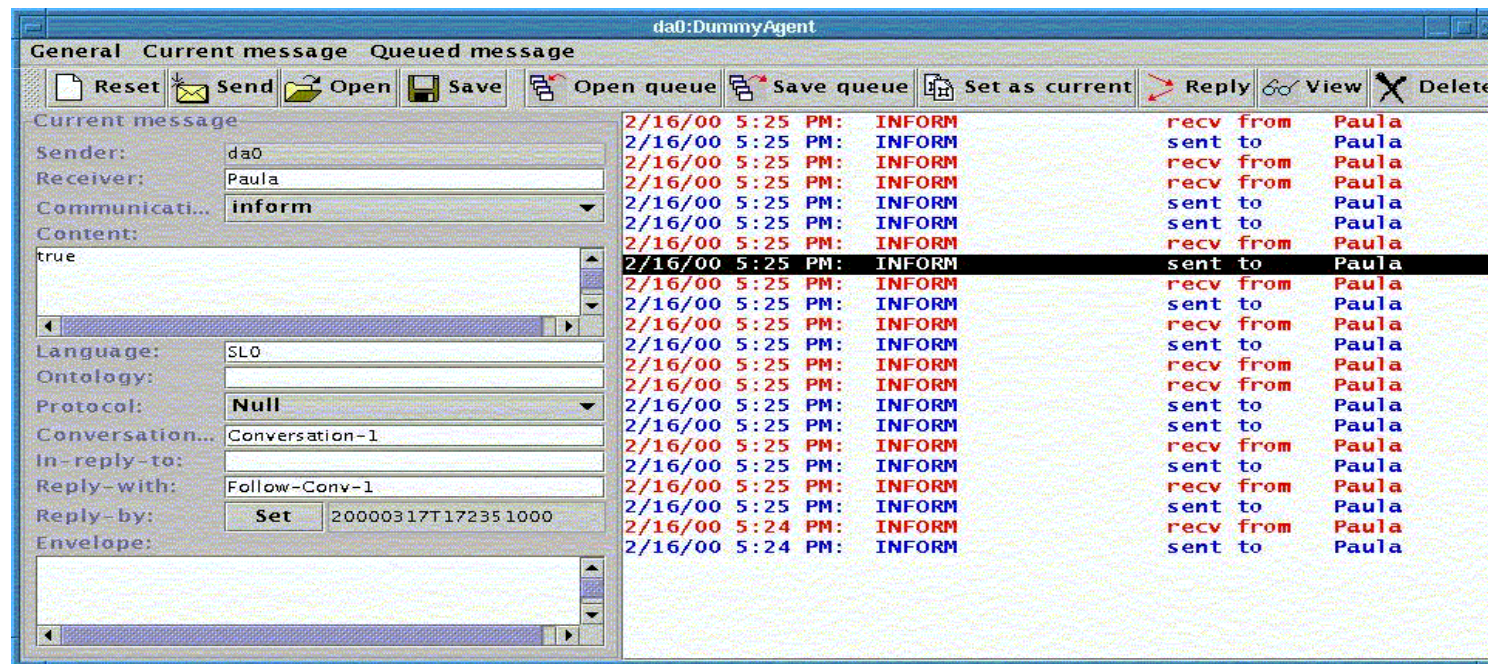
Entorno

El **ams** es el *Agent Management System*. Proporciona un entorno con varios servicios para los agentes de la plataforma.

El **df** es el *Directory Facilitator*. Es un agente que proporciona un servicio de 'páginas amarillas' para los agentes conocidos para la plataforma.

Los agentes deben tener nombres únicos.

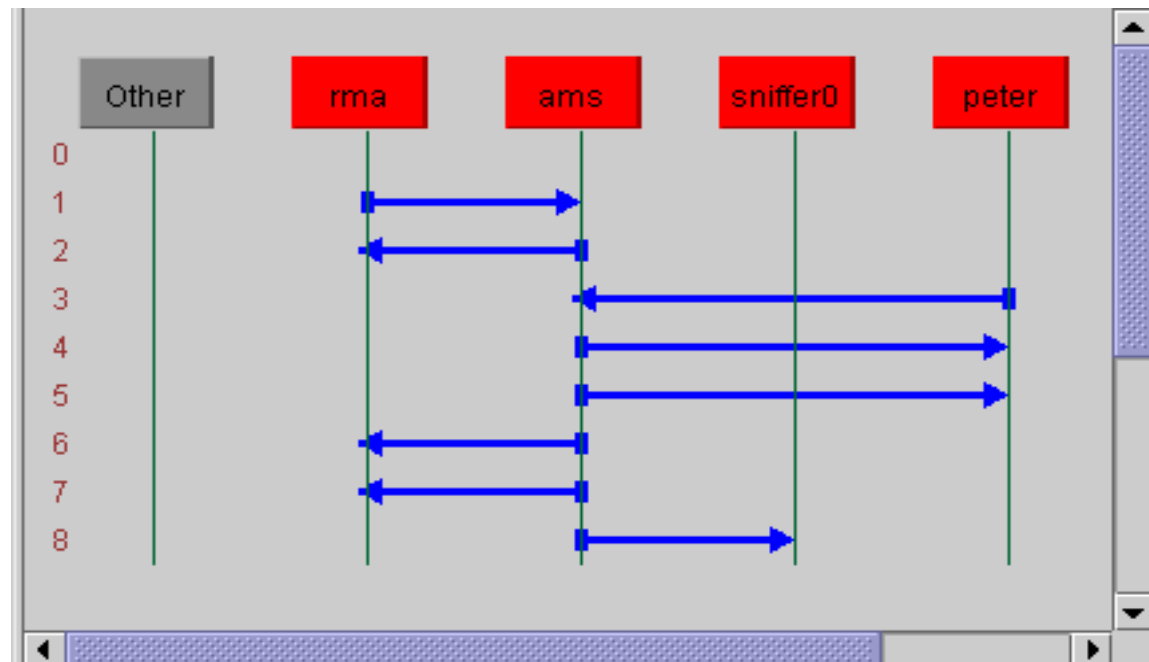
Se trata de un agente envióador de mensajes que tiene asociada una interfaz gráfica que nos permite indicar los distintos campos del mensaje a enviar, así como visualizar los mensajes enviados y recibidos.



Entorno

Sniffer Agent

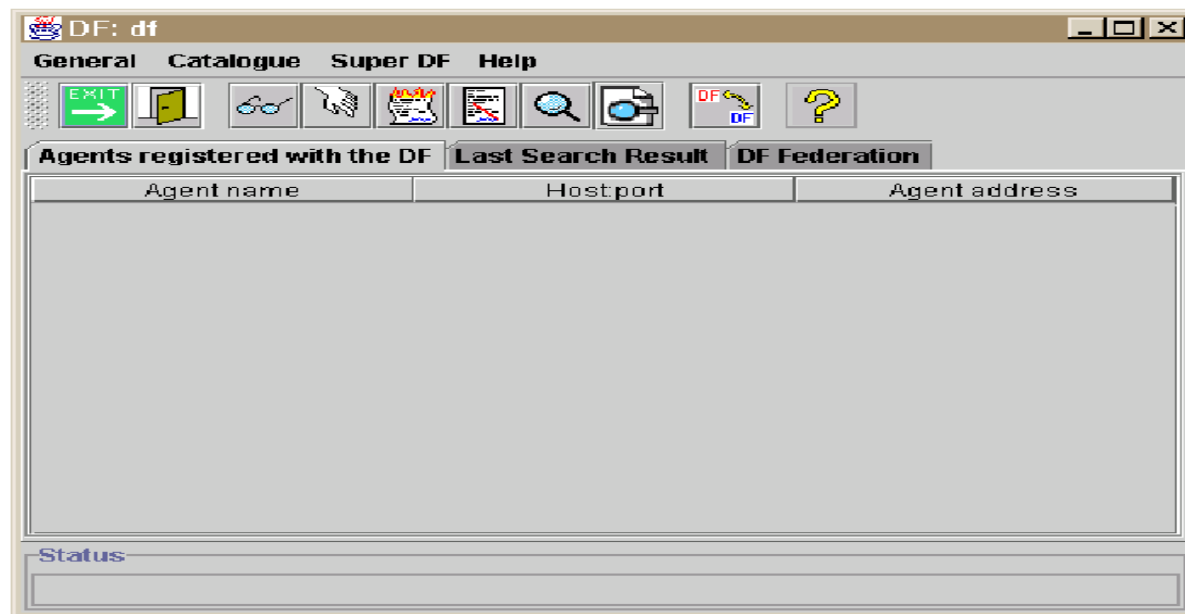
El agente Sniffer es una de las herramientas que nos proporciona la plataforma JADE para la depuración durante el proceso de desarrollo de nuestro sistema multi-agente. Básicamente, nos permite comprobar como se produce la interacción e intercambio de mensajes entre los agentes que indiquemos.



Entorno

DF GUI: Es un interfaz del Directory Facilitator. Permite:

- Ver descripciones de los agentes registrados
- Registrar y desregistrar agentes
- Modificar registros
- Buscar descripciones
- Puede ser iniciado desde el RMA





Estructura de un Agente

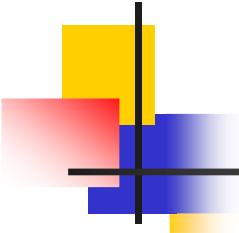
Un agente JADE es una instancia de una clase de JAVA definida por el usuario que extiende la clase AGENTE básica.

```
public class MI_AGENTE extends Agent { ...
```

El ciclo de vida de un agente JADE sigue el ciclo de FIPA

El método setup es donde se inicializa el agente

Las tareas que realiza un agente en JADE se estructuran en comportamientos



Ejecutar un Agente

Para que un agente o un conjunto de agentes sean creados al inicializar la plataforma JADE deberemos ejecutar jade.Boot de la siguiente forma:

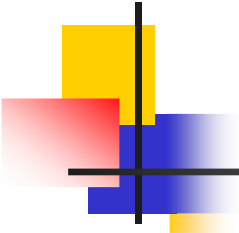
java jade.Boot nombre1:clase1 nombre2:clase2 ...

Para cada agente deberemos indicar su nombre y la clase a la que instancia

java jade.Boot agenteEjemplo:ClaseAgente

Si además quisieramos que se mostrara la interfaz del agente RMA :

java jade.Boot -gui agenteEjemplo:ClaseAgente



Ejemplo: Agente Hola Mundo

```
import jade.core.Agent;
import java.util.*;
public class Hola extends Agent {
    protected void setup() {
        System.out.println("Hola mundo. Soy un agente!");
        System.out.println("Mi nombre local es "+getAID().getLocalName() );
        System.out.println("Mi GUID es "+getAID().getName());
        System.out.println("Mi dirección es: ");
        Iterator it = getAID().getAllAddresses();
        while (!it.hasNext());{
            System.out.println( it.next() );
        }
    }
}
```



Ejemplo: Agente Comportamiento

SimpleAgent

```
java jade.Boot SA:examples.behaviours.SimpleAgent
```

TimeAgent

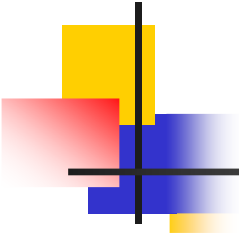
```
java jade.Boot TA:examples.behaviours.TimeAgent
```

ComplexBehaviourAgent

```
java jade.Boot CBA:examples.behaviours.ComplexBehaviourAgent
```

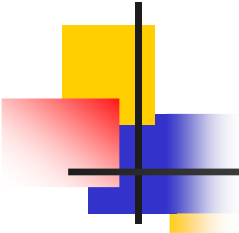
FSMAgent

```
java jade.Boot FSMA:examples.behaviours.FSMAgent
```



Ejemplo: Ping Agente

1. crear un nuevo agente en un container: nombre: PA clase: `examples.PingAgent.PingAgent`
2. lanzar el Dummy agent desde el menu tools
3. desde el dummy agent, mandarle un mensaje al agente creado, PA (nombre local), con performative: `QUERY_REF`, y contenido: ping



Ejemplo: Book Trading

1. crear dos sellers desde el GUI, darles unos cuantos libros. No cerrar su GUI o el agente muere (observar la ventana msdos)

BS1 `examples.bookTrading.BookSellerAgent`

BS2 `examples.bookTrading.BookSellerAgent`

2. crear el agent comprador y darle como parametro el nombre de un libro que los vendedores tengan

BB1 `examples.bookTrading.BookBuyerAgent`

3. idem, dandole el nombre de un libro que los vendedores NO tengan

BB2 `examples.bookTrading.BookBuyerAgent`

4. Introducir en al menos uno de los vendedores, el libro que necesita el segundo comprador, etc...



Ejemplo: Paginas Amarillas

DFRegisterAgent, DFSearchAgent y DFSubscribeAgent

1. desde una ventana msdos: `java jade.Boot -gui`
`provider:examples.yellowPages.DFRegisterAgent(my-forecast)`
2. desde otra ventana: `java jade.Boot -container`
`searcher:examples.yellowPages.DFSearchAgent`
3. desde el GUI crear un agente: nombre: subscriber clase:
`examples.yellowPages.DFSubscribeAgent`
4. desde el GUI crear un agente: nombre: provider-1 clase:
`examples.yellowPages.DFRegisterAgent` parametro: forecast-1 observar
como el subscriber detecta la existencia de los providers