Упражнение 5: Регистър на помощни средства (Directory Facilitator)

Регистърът на помощни средства (Directory Facilitator) е централизиран регистър на вписванията, който свързва описанието на услуга с идентификатор на агент (AID).

1. Регистриране и търсене с DFAgentDescription

DFAgentDescription (DFD) се използва както за ***добавяне*** на вписване, така и за ***търсене*** на услуги и има една и съща основна структура на данните.

Търсенето връща масив от пълни записи (с AIDs), чиито атрибути съответстват на описанието и от тези записи може да бъдат извлечени идентификационните номера на подходящите агенти.

Структурата на описанието на агент в регистъра DF е следната:

```
DFAgentDescription
      Name:       AID        //  Required for registration
      Protocols:   set of Strings
      Ontologies:  set of Strings
      Languages:   set of Strings
      Services:    set of
      {   Name:  String //Required
         Type:  String        //  Required
         Owner: String
         Protocols:   set of Strings
         Ontologies:  set of Strings
         Languages:   set of Strings
         Properties: set of
            { Name:   String
              Value:  String
            }
      }
```

Ако даден агент иска да публикува една или няколко услуги, трябва да предостави описание DFAgentDescription, което да съдържа неговият уникален номер (AID), списък с предоставените услуги и списък на езиците и онтологията, които другите агенти трябва да използват, за да осъществят контакт с него.

Например в приложение, свързано с продажби, при което има купувачи и продавачи, могат да бъдат описани чрез използване на множество типове на услуги (service type), като например "buyer" или "seller".

2. Регистрация на услуга в DF

Използва се DFD (DFAgentDescription) с име (тук *name* означава Agent ID). Регистриращият метод изисква два параметъра: референция към агента, който ще извърши регистрацията и DFD.

```
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.*;
import jade.domain.FIPAException;
....
   DFAgentDescription dfd = new DFAgentDescription();
   dfd.setName(getAID());
   try {
      DFService.register(this, dfd);
```

```
    }
    catch (FIPAException fe) {
       fe.printStackTrace();
    }
```

Добавя се услугата описание на тип "buyer". Също така трябва да се зададе име на услугата, защото всички описания на услуги изискват име. Често се използва локалното име на агента.

```
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(getAID());
ServiceDescription sd  = new ServiceDescription();
sd.setType("buyer");
sd.setName(getLocalName());
dfd.addServices(sd);
try {
   DFService.register(this, dfd);
}
catch (FIPAException fe) {
         fe.printStackTrace();
}
```

Когато се регистрират различни агенти, единственото нещо, което обикновено се променя е *Service Description*, така че може да се създаде метод *register()*, в който се записва многократно използваният код.

```
void register(ServiceDescription sd)
{
   DFAgentDescription dfd = new DFAgentDescription();
   dfd.setName(getAID());
   dfd.addServices(sd);
   try {
      DFService.register(this, dfd );
   }
   catch (FIPAException fe) { fe.printStackTrace(); }
}
```

Използване на метода за регистриране *register()*:

```
ServiceDescription sd  = new ServiceDescription();
sd.setType("buyer");
sd.setName(getLocalName());
register(sd);
```

Методът *register()* може да бъде разширен така, че първо да тества и изтрива дублираните записи, преди да добави нов.

```
void register(ServiceDescription sd)
{
         DFAgentDescription dfd = new DFAgentDescription();
         dfd.setName(getAID());
    try {
    DFAgentDescription list[] = DFService.search(this, dfd);
           if (list.length>0)
         DFService.deregister(this);
      dfd.addServices(sd);
           DFService.register(this,dfd);
```

```
        }
    catch (FIPAException fe) { fe.printStackTrace(); }
}
```

3. Дерегистриране на агент от DF

Добра практика е да се изтрие вписването на агент в регистъра, когато приключи. Обичайният начин да се дерегистрира агент е метод с името *takeDown()*. Средата JADE извиква метода *takeDown()* автоматично, когато се унищожава агент. Пример за използване:

```
protected void takeDown()
{
  try { DFService.deregister(this); }
  catch (Exception e) {}
}
```

4. Търсене на услуга в DF

За търсене на услуга в DF трябва да се създаде DFD без AID, където съответните полета са инициализирани със свойствата, които се изискват. Търсенето връща масив от DFDs (с AIDs), чиито свойства съвпадат с описанието и може да се извлекат идентификационните номера на подходящите агенти от вписванията.

На следващия пример е даден типичен код за намиране на купувач.

```
DFAgentDescription dfd = new DFAgentDescription();
ServiceDescription sd  = new ServiceDescription();
sd.setType("buyer");
dfd.addServices(sd);
DFAgentDescription[] result = DFService.search(this, dfd);
System.out.println(result.length + " results" );
if (result.length>0)
   System.out.println(" " + result[0].getName() );
```

Следващият пример съдържа два метода за търсене, базирани на *ServiceType String*, които връщат или един или всички съвпадащи агенти.

```
import jade.core.*;
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.*;
import jade.domain.FIPAException;
public class SearchDF2 extends Agent
{
  protected void setup()
  {
    ServiceDescription sd  = new ServiceDescription();
    sd.setType("buyer");
    sd.setName(getLocalName());
    register(sd);
    AID agent = getService("buyer");
    System.out.println("\nBuyer: "
      +(agent==null ? "not Found" : agent.getName()));
    agent = getService("auction");
    System.out.println("\nAuction: "
```

```
        +(agent==null ? "not Found" : agent.getName()));
      AID [] buyers = searchDF("buyer");
      System.out.print("\nBUYERS: ");
      for (int i=0; i<buyers.length; i++)
         System.out.print(buyers[i].getLocalName() + ",");
      System.out.println();
      doDelete();
      System.exit(0);
   }
   protected void takeDown()
   {
         try { DFService.deregister(this); }
         catch (Exception e) {}
   }
// ------------- Utility methods to access DF ----------------
   void register(ServiceDescription sd)
   {
         DFAgentDescription dfd = new DFAgentDescription();
         dfd.setName(getAID());
     try {
      DFAgentDescription list[]=DFService.search(this,dfd);
               if (list.length>0)
               DFService.deregister(this);
       dfd.addServices(sd);
               DFService.register(this,dfd);
     }
      catch (FIPAException fe) { fe.printStackTrace(); }
   }

   AID getService(String service)
   {
      DFAgentDescription dfd = new DFAgentDescription();
      ServiceDescription sd = new ServiceDescription();
      sd.setType( service );
      dfd.addServices(sd);
      try
      {
      DFAgentDescription[] result=FService.search(this,dfd);
      if (result.length>0)
          return result[0].getName() ;
      }
      catch (FIPAException fe) { fe.printStackTrace(); }
      return null;
   }

   AID [] searchDF(String service)
   {
      DFAgentDescription dfd = new DFAgentDescription();
      ServiceDescription sd = new ServiceDescription();
      sd.setType(service);
```

```
        dfd.addServices(sd);
        SearchConstraints ALL = new SearchConstraints();
        ALL.setMaxResults(new Long(-1));
        try
        {
           DFAgentDescription[] result=
                DFService.search(this, dfd, ALL);
           AID[] agents = new AID[result.length];
           for (int i=0; i<result.length; i++)
              agents[i] = result[i].getName();
           return agents;

        }
        catch (FIPAException fe) { fe.printStackTrace(); }
        return null;
    }
}
```