Javaプログラミングの Tip集

第４回

2007年7月17日
アイティエス

エージェント(Agent)

- 「代理人」、「動作主体」の意味を持つ人工知能(AI：Artificial Intelligence)に由来している
  - ➢1958年
- ネット上にソフトウェアでエージェントを実現する
  - ➢モバイルエージェント

人間が役割を担うエージェント

- 保険の代理人
- プロスポーツの代理人
- 国家レベルの諜報部員

など ⇒ ソフトウェアで擬人化

エージェントの擬人性要素

- インテリジェンス(intelligence)
- 自律性(autonomy)
- コラボレイティブ(collaborative)

モバイルエージェントの持つべき特性

- 自律的である・・・・・・・・・自律性(autonomy)がある
- 計画的に推論できる・・・インテリジェンス(intelligence)がある
- 協調的である・・・・・・・・・コラボレイティブ(collaborative)なシナジーがある

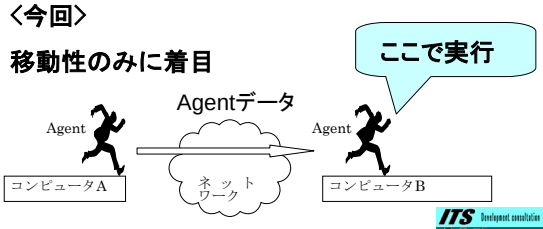
- 移動できる・・・・・・・・・・・**移動性(mobility)**がある
- 通信負荷が減らせる・・・**負荷分散(load balancing)機能**がある

モバイル性に必要な技術

Agentデータが他のコンピュータに移動する

〈今回〉

移動性のみに着目



エージェントの操作イメージ

Agent転送の指示を与え自分の所で実行結果を得る

存在するプログラム：Agent

プログラムは複数のクライアントを処理するためにはスレッド構成にする

クライアントをスレッドで構成

```
try{
   //ahオブジェクト生成
   AgLoaderHandler agLoaderh = new AgLoaderHandler(sock);
   //スレッド生成
   Thread th1 = new Thread(agLoaderh);
   //スレッド起動
   th1.start();
}
catch(Exception e){
    System.err.print(e);
    System.exit(1);
}
```

サーバをスレッドで構成

```
while(true){
    try{
        //接続要求待機
        Socket sock = servsock.accept();
        System.out.println("accept now");
        clientProc cp = new clientProc(sock);
        //Threadクラスのインスタンス生成
        Thread th = new Thread(cp);
        //スレッド起動
        th.start();
    }catch (IOException e){
        System.err.println("accept error.");
        System.exit(1);
    }
}
```

エージェントの動き

指示はクライアントで行う

① プログラムを指定(クラスファイル名:Agent)

② 実行したいメソッドを指定(例:getSeason())

④ Agentクラスファイルを自分の所で実行

応答をサーバが担当する

③ Agentクラスファイルをクライアントに転送

クライアントの機能

① サーバにある情報をクライアントに持ってきてクライアントで実行する

②「例:季節データ」を得るクラスファイルを転送してもらう指示をサーバに与える

③ クライアントで貰ったあとは実行して「季節」を表示する

クライアントにはAgent.javaやAgent.classは存在しない

サーバのRunnableに用いたコンストラクタ定義

```
public clientProc(Socket s) throws IOException{
    this.s = s;
    //入出力ストリーム宣言
    out = s.getOutputStream();
    bout = new BufferedOutputStream(out);
    in = new BufferedReader(new InputStreamReader(s.getInputStream()));
}
```

サーバでのスレッド本体

```
public void run(){
   try{
       ByteArrayOutputStream byteout = new ByteArrayOutputStream();
       className = in.readLine();              //１行読み込み
       classFile = className + ".class";       //クラスファイル名
       System.out.println("クラスファイル名:"+ classFile);
       bin = new BufferedInputStream(new FileInputStream(classFile));
       int len;
       byte[] buff = new byte[LEN];
       while((len = bin.read(buff,0,LEN)) != -1){
           bout.write(buff, 0, len);
       }
       System.out.println("出力完了");
       bout.flush();
       bout.close();
       bin.close();
   }catch(Exception e){
       e.printStackTrace();
   }
}
```

クライアントのRunnable に用いたコンストラクタ定義

```
public AgLoaderHandler(Socket sock) throws IOException{
    this.sock = sock;
    out = new BufferedOutputStream(sock.getOutputStream());
}
```

クライアントでのスレッド本体

```
sendData = className + CRLF;
sendbuf = sendData.getBytes();
System.out.println("------取得内容出力------");
send(sendbuf);
loader = new AgClassLoader(sock)
cl = loader.findClass(className);
obj = cl.newInstance();//クラスをロードして実行

retobj = invokeMethod(methodName,obj);
System.out.println(retobj);
```

invokeMethodの構成

```
private Object invokeMethod(String methodName,Object obj){
        Method method;
        try{
            Object[] argsObj = {};
            Class[]  paramTypes  = {};
            //メソッド取得
            method = obj.getClass().getMethod(methodName, paramTypes);
            //メソッド呼び出し
            return method.invoke(obj,argsObj);
        }catch(NoSuchMethodException e){
            e.printStackTrace();
        }catch(IllegalAccessException e){
            e.printStackTrace();
        }catch(IllegalArgumentException e){
            e.printStackTrace();
        }catch(InvocationTargetException e){
            e.printStackTrace();
        }
        return null;
    }
```

サーバの操作

サーバを先に起動しておく

>java classServer 5000
Connections on port 5000
accept now
クラスファイル名:Agent.class
出力完了

クライアントの操作１

>**java AgLoaderClient 192.168.1.1 5000**
Connected to 192.168.1.1
Remote ClassName:**Agent**
Remote methodName:**getSeason**
------取得内容出力------
Agent: an instance created. 7
7月夏です ← Agentの実行結果
>

クライアントの操作２

>**java AgLoaderClient 192.168.1.1 5000**
Connected to 192.168.1.1
Remote ClassName:**Agent1**
Remote methodName:**getGreeting**
------取得内容出力------
Agent1: an instance created. 20
こんばんは ← Agent1の実行結果
>