Distributed Systems – TD8 : JXTA - first steps (2)

Bogdan.Pasca @ens-lyon.fr

20 November 2009

If you did not finished Lab #7, please download the source code of the HelloWorld application from :
`http://perso.ens-lyon.fr/bogdan.pasca/ds/resources/td08/HelloWorld.tar.gz`

In the previous laboratory we tested that the `discoveryEvent` method is executed by starting the $JXTA - Shell$ so that the $JXTA - Shell$ replies the discovery request message.

Request

Create the server side of the application which publishes advertisements for the client side of the application. Test the functionality of your applications.

Hints

A pipe advertisement can be created in the following way :

```
PipeAdvertisement advertisement = (PipeAdvertisement)
        AdvertisementFactory.newAdvertisement(PipeAdvertisement.getAdvertisementType());

advertisement.setPipeID(IDFactory.newPipeID(PeerGroupID.defaultNetPeerGroupID));
advertisement.setType(PipeService.UnicastType);
advertisement.setName("HelloWorldServerPipeAdvertisement");
```

Once you have created an advertisement you need to publish it :

```
discovery.publish(pipeAdv, lifetime, expiration);
discovery.remotePublish(pipeAdv, expiration);
```

Request

Print the details of the message received on the *Client* side of the application.

```
JXTA Started
Peer name       :HelloWorld_n
Peer ID         :urn:jxta:uuid−59616261646162614E50472050325033EF0F9491DB7D418DB39FC0AB9802EE6A03
Peer group name :NetPeerGroup
Peer group ID   :urn:jxta:jxta−NetGroup
Waiting for a rendezvous connection
Connected : true
Sending discovery message
Sending discovery message
Nov 19, 2009 9:47:05 PM net.jxta.impl.endpoint.tcp.TcpMessenger <init>
INFO: Connection from 140.77.13.23:41630
Discovery event ..... your code here
```

```
 [  Got a Discovery Response [1 elements]  from peer :
    jxta://uuid-59616261646162614E50472050325033D0281BF49CFE49688F04C2EF5337BEE803   ]
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:MIA>
<jxta:MIA xml:space="default" xmlns:jxta="http://jxta.org">
  <MSID>
    urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000010206
  </MSID>
  <Desc>
    Default Network PeerGroup reference implementation
  </Desc>
  <Comp>
    <Efmt>
      JDK1.4.1
    </Efmt>
    <Bind>
      V2.0 Ref Impl
    </Bind>
  </Comp>
  <Code>
    net.jxta.impl.peergroup.ShadowPeerGroup
  </Code>
  <PURI>
    http://jxta-jxse.dev.java.net/download/jxta.jar
  </PURI>
  <Prov>
    sun.com
  </Prov>
  <Parm>
    <Svc>
      urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000100106
    </Svc>
    <Svc>
      urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000E0106
    </Svc>
    <Svc>
      urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000060106
    </Svc>
    <Svc>
      urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000070106
    </Svc>
    <Svc>
      urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000080106
    </Svc>
    <Svc>
      urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000050306
    </Svc>
    <Svc>
      urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000020106
    </Svc>
    <Svc>
      urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000030106
    </Svc>
    <Svc>
      urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000040106
    </Svc>
    <Proto>
      urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000110106
    </Proto>
    <Proto>
      urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000F0106
    </Proto>
```

Hint

See page 65 of JXSE 2.5 Programmers Guide.

Sending messages

Next we will learn how to use pipes to send messages between two JXTA peers, and also shows how to implement the RendezvousListener interface.

We will need to create two separate applications [1] :

PipeClient : creates an input pipe with a pre-created pipe ID, and listens for messages on this pipe

PipeServer : creates an output pipe, and sends a message on this pipe

You will need to instantiate a PipeService which is instantiated using the `getPipeService()` method on a `PeerGroup` object.

```
  PipeService pipeService = manager.getNetPeerGroup().getPipeService();
```

You for this example you will create the pipe advertisement from an existent ID. The ID is defined in the following way :

```
public final static String PIPEIDSTR =
  "urn:jxta:uuid-59616261646162614E50472050325033C0C1DE89719B456691A596B983BA0E1004";
```

We need to create a pipeID from the previously defined string :

```
PipeID pipeID = null;
try {
    pipeID = (PipeID) IDFactory.fromURI(new URI(PIPEIDSTR));
} catch (URISyntaxException use) {
    use.printStackTrace();
}
```

One the pipe advertisement is created, we need to assign the it the proper ID :

```
advertisement.setPipeID(pipeID);
```

In order to send messages out the output pipe, the client class must implement the OutputPipeListener interface.

```
  public class PipeClient implements OutputPipeListener
  ...
  pipeService.createOutputPipe(pipeAdv, this);

  public void outputPipeEvent(OutputPipeEvent event) {
```

Once the pipe has been resolved, the outputPipeEvent method is called. Here, we get the output pipe :

```
private OutputPipe outputPipe;
...
public void outputPipeEvent(OutputPipeEvent event) {
  outputPipe = event.getOutputPipe();
```

[1] If you are running both applications on the same system, you will need to run each application from a separate subdirectory so that they can be configured to use separate ports.

Then we create a new message.

```
  Message msg = new Message();
```

We will add to this message a string message element containing the current date and finally we will send the message.

```
Date date = new Date(System.currentTimeMillis());
StringMessageElement sme = new StringMessageElement("MyMessage", date.toString(), null);
msg.addMessageElement(null, sme);
outputPipe.send(msg);
```

The server side has to create an input pipe :

```
  inputPipe = pipeService.createInputPipe(pipeAdv, this);
```

The server side implements the PipeMsgListener interface. It has to implement the method `public void pipeMsgEvent(PipeMsgEvent event)`. We need to process the event. So we create a message and print its contents on the screen :

```
Message msg = event.getMessage();
printMessageStats(msg, true);
```

Requests

– Implement these two applications.

– Extend the applications so that the pipe advertisement is created and exchanged between the two peers using the discovery service.