Distributed Systems – TD6 : Introduction to JXTA
The JXTA Shell

Bogdan.Pasca @ens-lyon.fr

30 October 2009

The JXTA shell is a command-line interface to Sun's JXTA peer-to-peer framework. It is used to prove the concepts behind JXTA **without writing one line of code**.

Fetching the JXTA Shell

Download and unpack the archive :
`http://download.java.net/jxta/jxta-jxse/2.4/jxta-shell-2.4.zip`

Using your preferred shell, enter the `shell` folder of the newly extracted archive. Make the file `run.sh` executable by passing the command `chmod +x run.sh` to your shell.

Launch the JXTA Shell by executing `run.sh` : `./run.sh`. The sequence of commands you should have used :

```
bogdan@pulse:~/Desktop/jxta-shell-2.4$ ls -l
total 0
drwxr-xr-x 2 bogdan bogdan 248 2006-06-20 15:34 lib
drwxr-xr-x 2 bogdan bogdan 184 2006-06-20 15:34 shell
bogdan@pulse:~/Desktop/jxta-shell-2.4$ cd shell/
bogdan@pulse:~/Desktop/jxta-shell-2.4/shell$ ls -l
total 24
-rw-r--r-- 1 bogdan bogdan 7680 2003-02-14 12:09 jxta.exe
-rw-r--r-- 1 bogdan bogdan 2398 2003-12-11 13:50 Jxta_Readme.html
-rw-r--r-- 1 bogdan bogdan  172 2005-06-03 12:17 run.bat
-rw-r--r-- 1 bogdan bogdan  195 2005-06-03 12:17 runjdk.bat
-rw-r--r-- 1 bogdan bogdan  377 2005-06-03 12:17 run.sh
bogdan@pulse:~/Desktop/jxta-shell-2.4/shell$ chmod +x run.sh
bogdan@pulse:~/Desktop/jxta-shell-2.4/shell$ ls -l
total 24
-rw-r--r-- 1 bogdan bogdan 7680 2003-02-14 12:09 jxta.exe
-rw-r--r-- 1 bogdan bogdan 2398 2003-12-11 13:50 Jxta_Readme.html
-rw-r--r-- 1 bogdan bogdan  172 2005-06-03 12:17 run.bat
-rw-r--r-- 1 bogdan bogdan  195 2005-06-03 12:17 runjdk.bat
-rwxr-xr-x 1 bogdan bogdan  377 2005-06-03 12:17 run.sh
bogdan@pulse:~/Desktop/jxta-shell-2.4/shell$ ./run.sh
```

The following JXTA configuration window (Figure 1) should appear.

FIG. 1 – JXTA Configurator

Choose a peer name, and a password having a minimum of 8 characters. Then go to the third tab, the Rendezvous/Relays tab (Figure 2), and add seeding URIs for both rendezvous, `http://rdv.jxtahosts.net/cgi-bin/rendezvous.cgi?2` and relays `http://rdv.jxtahosts.net/cgi-bin/relays.cgi?2`.

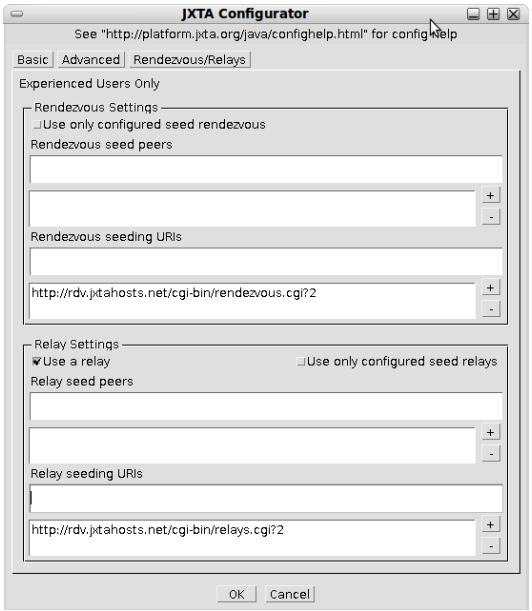

FIG. 2 – JXTA Configurator

You now have to type-in the password you selected for you peer (Figure 3)

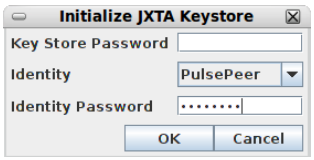

FIG. 3 – JXTA Login

The JXTA Shell (Figure 4) should appear :

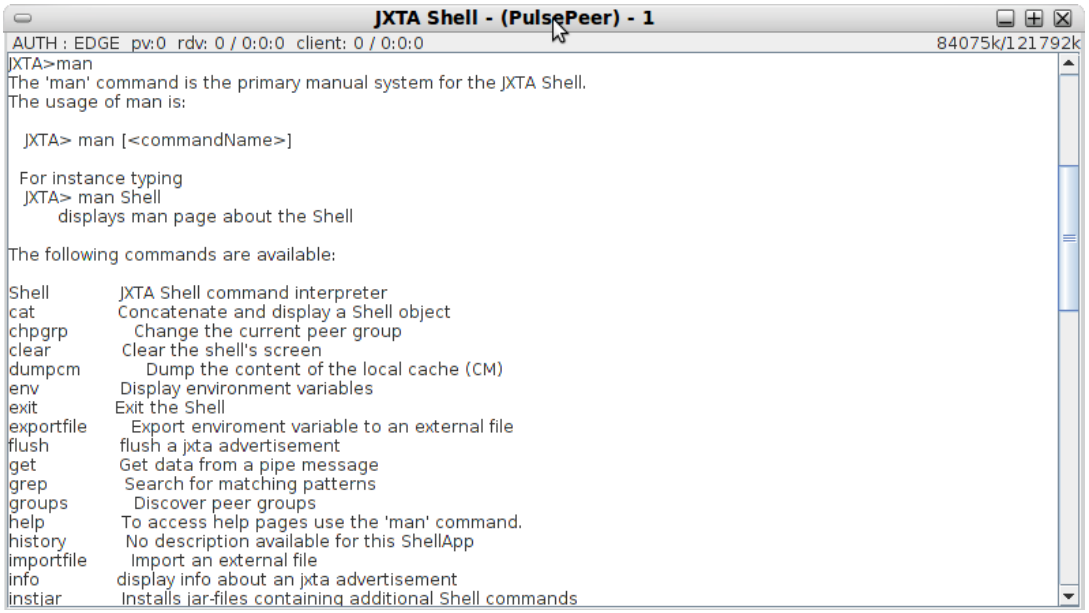

FIG. 4 – JXTA Shell

JXTA commands

For a complete list, take a look at the documentation
`http://download.java.net/jxta/jxta-jxse/2.4/jxta-shell-doc-2.4.zip`.

Peer reconfiguration

The `peerconfig` command is used to re-configure the local peer. The command forces the platform to show a configuration dialog the next time it is started. After the command is run the shell needs to be restarted in order to display the configuration dialog.

Peer Discovery

The `peers` command is used to discover other peers within a peer group or at a specified peer location. Running `peers` command with no options lists only the peers already known by the peer (cached). The '-r' option is used to send a propagate request to find new peers. The command stores results in the local cache, and inserts advertisement(s) into the environment, using the default naming : peerX where X is a growing integer number.

Local peer information

The `whoami` displays information about a peer or a peergroup. With no option, `whoami` returns information about the local peer. The '-g' option returns information about the current peer group joined.

Local environment

Shell environment variables are defined as a result of executing Shell commands or using the 'set' command. The '=' operator can be used to assign the result value of a command to a particular variable. For example 'myenv = mkmsg' will assign a new message object to the 'myenv' environment variable.

Use the `env` command to visualize the status of the environment variables.

Local variable contents

The `cat` command displays on stdout the contents of objects stored in environment variables. The `cat` command knows how to display a limited set of JXTA object types :

- Advertisement
- Credentials
- Document
- StructuredDocument
- Message
- PeerInfoResponseMessage

If you are not sure, try to cat the object anyway – 'cat' will try to display the object as best it can.

The Rendezvous status

The `rdvstatus` displays information about the rendezvous service in the current group. The command shows the current peerview and any rendezvous or client connections.

Try the `peers -f` command. At some point, it might be appropriate to remove the Peer Advertisements from the local cache, eliminating the local peer's knowledge of other peers on the network. To flush the local cache of Peer Advertisements, use this command. The only remaining Peer Advertisement will be that of your own local peer.

Peer Groups

A new peer group can be created from within the JXTA Shell :
`JXTA>mygroupadv = newpgrp -n mygroup`
The `newpgrp` command creates a new peer group advertisement with a random group id which uses the same implementation as the current peer group. You can find the services of the current peergroup via the command 'whoami -g'.

The `join` command is used to instantiate and join a peergroup that was created via the `newpgrp` command or using an advertisement that was previously discovered.
If no argument is given, join lists all the existing groups and the current group on the local peer. When a group is joined successfully, an environment variable is created. This variable holds the group object.
Upon joining the peer group, depending on the membership authentication required, the user will be asked for the identity he/she wants to have in the peer group. An identity is used to assign credentials to users when accessing peer group resources. Each peer group can define their own set of identities available in the peer group.

To join the newly created group :
`grp = join -d mygroupadv`

Pipes and Messages

Creating Pipes

The `newpipe` command creates a new pipe advertisement with a random pipe id.
For example : `JXTA>mypipeadv = newpipe -n mypipe`
This creates a new pipe advertisement of the default type. The new pipe is given the name 'mypipe'. Before you can do anything with the pipe you need to instantiate it via the 'mkpipe' command.
You can check the advertisement using the `cat mypipeadv` command.
You will publish the advertisement with `publish mypipeadv`.
Finding advertisements is done using the command `search`.

The 'mkpipe' creates an input pipe or an output pipe from a given pipe advertisement document. In order for pipes to communicate an input and output pipe needs to be created with the same pipe advertisement. PipeService advertisements are structured documents that contains at least the unique pipe Id. The pipe Id uniquely identifies a pipe in the JXTA world. Pipes are not localized or binded to a physical peer. PipeService connections are established by searching for pipe advertisements and resolving dynamically the location of an input pipe object binded to that advertisement. An input pipe can be binded to the same pipe advertisement on multiple peers transparently to the output pipe. The output pipe does not need to known on which physical peer the input pipe is located. To communicate with the pipe, the output pipe needs to search for the input pipe that binds that advertisement.
The two options are :
– '-i' create an input pipe
– '-o' create an output pipe

For example :

```
JXTA>myinpipe = mkpipe -i mypipeadv
JXTA>myoutpipe = mkpipe -o mypipeadv
```

Creating Messages

Communication between an input and an output pipe relies on the capability to form a message object to exchange. If you import a text file into the Shell, you can pack it inside a message :

```
JXTA>theDataObject = importfile -f yourStructuredXML.txt
JXTA>msg = mkmsg
JXTA>put msg myDesiredTag theDataObject
```

To send the message :

```
JXTA>send myoutpipe msg
```

To receive the message :

```
JXTA>rcvMessage = recv -t 5000 myinpipe
```

To see the received message :

```
JXTA>tmp = get rcvMessage myDesiredTag
JXTA>cat tmp
```

Talking between peers

The `talk` command implements a simple instant messaging command where two users on two remote peers can exchange messages. Messages are displayed on the Shell stdout. In order to use 'talk' the user needs to register himself. This is done via the following steps :

1. Register via `'talk -register <username>'` command. This command creates a Talk advertisement for that user. This has to be done only once, the first time the user registers with talk. The system remembers it across reboot. -secure can be added in order to establish a secure talk session. -propagate can be added in order to establish a chatroom style talk session.
2. Login via `'talk -login <username>'` command. This command logs the user and starts a listener daemon. This has to to be done every time the peer is restarted.
3. User can talk to another user via the command `'talk -u <myusername> <destusername>'`. This command will prompt the user to enter the message he/she wants to send
   JXTA>talk -u me you # talk : Connected to user you Type your message. To exit, type '.' at beginning of line

To stop receiving any more talk messages. The user can stop the talk listener daemon by entering the command 'talk -logout <username>'

Example :

```
JXTA>talk -register me
JXTA>talk -login me
JXTA>talk -search
JXTA>talk -u me you
```

This example shows how a new user 'me' can register and log into talk, an talk to the user 'you'. User 'you' needs to be registered and logged on.

1 Requirements

Use the JXTA Shell to explore the JXTA world.
- Check the information related to the peer currently running the shell.
- Check the group information for the current shell.
- Visualize peer/group/pipe advertisements
- Create pipe advertisements
- Bind the pipe advertisements to input and output pipes
- fetch data from structured XML file
- create a message containing the data read from the file
- send/receive data
- display received data
- explore chatting using the `talk` command