Setting Up The NextBus JMS Pump

For the Wildly Curious and Impatient

http://sourceforge.net/projects/nextbusapi/

Jim Doyle <rockymtnmagic@gmail.com> 5 Sep 2010

1. Download and install Apache ActiveMQ 5.6.0. For the completely impatient, you can simply download and unzip the distribution and start the MQ server. There are a few tiny things you need to configure first:

- Edit `activemq.xml` and enable the ActiveMQ to listen on all network interfaces. While in this file, also enable the STOMP protocol connector. STOMP will allow non-Java message driven clients to play with the toys as well. Find the connector for Openwire and do this:

```
<transportConnector name="openwire" uri="tcp://0.0.0.0:61616"/>
```

Next, add this just underneath add the stomp transport definition:

```
<transportConnector name="stomp" uri="stomp://0.0.0.0:61613"/>
```

The Pumper will send JMS Messages, including Serialized POJOs from the NextBus XML API into the message broker. It will also attach Headers that largely contain all the attributes of the full POJO. Since things like PHP, Ruby, C#, Javascript or Python cant eat Java POJOs, they can eat the Header information instead. The message header fields have pretty much all the infornation that is inside of the Java POJO (Route, Direction, GPS location, Time). Thus, the Headers are not only useful for implementing Subscriber Message Selection rules, but also for simply getting the data without the need of using the payloaded Serialized Java class.

- Start the ActiveMQ Broker:

```
$ ./bin/activemq start
```

- Go to the ActiveMQ Admin Console (http://localhost:8161/) and create a target Topic. Use the links: Manage The Broker → Topics → enter the Topic name (i.e. nextbus) ; Click Create

2. Download the pumper JAR, the sample `nbpumper.properties` file and the sample `log4j.properties` file from the Sourceforge release directory and put them together in some empty directory.

3. Edit **`nbpumper.properties`** and set the following to your preference:
   * The transit agency of your interest and liking (i.e. mbta)
   * The URL to your newly setup ActiveMQ server using port 61616 (OpenWire protocol)

4. Run the Pumper Super JAR like this ; note there are no Classpath or environment variables to worry about : this is the beauty of Maven Shade packaging:

```
C:\...>java -jar nextbus-jms-pump-1.0.1-RELEASE.jar
```

Give the pumper a good 30 seconds to get off the runway... Watch for any log messages at the ERROR Level.

5. Now go watch the message traffic arrive on the ActiveMQ Web Admin Console. Click <Topics> in the navigation bar of the Admin Webapp every 5 seconds or so and watch the message count rise each time the Pumper's Scheduler wakes up and probes NextBus.

6. Now lets become a client and tune into the message traffic using the STOMP Protocol and a simple TELNET client. This is what application code would do in say a non-Java smart-phone or some other clever scripting-language application that consumes transit data.

Open a TELNET connection to the STOMP port of the AMQ Server, and issue these commands. Note how there are blank lines with just a carriage-return, those are important, just hit Return. CTRL-ALT-@ sends the ASCII 'NUL' character into the Telnet Stream. That's what STOMP needs to see in order to execute a command.

```
$ telnet localhost 61613
CONNECT

CTRL-ALT-@
SUBSCRIBE
destination:/topic/nextbus
ack:auto

CTRL-ALT-@

-----------------------------------------------------------------
```

You should see a STORM of message traffic arrive in your Telnet window. Just sit back and watch. You can also disconnect from the Message broker. At that point the Broker will stop copying messages for you to receive, and will stop holding messages that you haven't acknowledged.

```
DISCONNECT

CTRL-ALT-@
```

Finally, you can CTRL-C the Pumper Daemon to shut it down gracefully.

Next Steps!

- Go read up and learn how to setup Passwords and Authorization Control on ActiveMQ. A good first implementation would be to password-protect the PUBLISH side of the Topic for the Pumper, but permit open, no-password-required SUBSCRIBE from the topic by any client.
- Try building a simple client application that Subscribes to the Topic. There are myriad ways to do this ranging from an Ajax library driven tool for HTML page, to server side stuff in PHP and Ruby. If you are a Java guy like me, then Spring JMS support will let you build support into an existing Web App. More exciting would be to build additional Spring Integration workflows that subscribe to the real-time datafeed and do other exciting things: Persist to a database, feed JFC Swing graphing widgets to make a Radar Display based on vehicle location, etc.
- Try using Message Selectors in the SUBSCRIBE phase of the STOMP sessions to tune-in to a limited slice of the traffic stream... For instance, you can select on a particular Route, or even a geographical region by Selecting on a range of lattitudes and longitudes.
- Understand the difference between Topics and Queues, aka Pub-Sub versus Point-to-Point. The Pumper supports both messaging destination type with a simple one-line edit of the properties file.

References

The Stomp Protocol Specification
http://stomp.github.com/stomp-specification-1.1.html#SUBSCRIBE

ActiveMQ Cross Language Support
http://activemq.apache.org/cross-language-clients.html

Spring Integration Reference
http://static.springsource.org/spring-integration/reference/htmlsingle/

Source Release Area (SVN)
https://sourceforge.net/p/nextbusapi/code/61/tree/trunk/jms-pump/
https://sourceforge.net/p/nextbusapi/code/61/tree/trunk/adapter/

Binary Release Area
https://sourceforge.net/projects/nextbusapi/files/releases/jms-pump-releases/