Enterprise Systems & Architecture

Lab 10 (Week 12): JMS Publish & Subscribe

Note: Complete over the next two lab sessions.

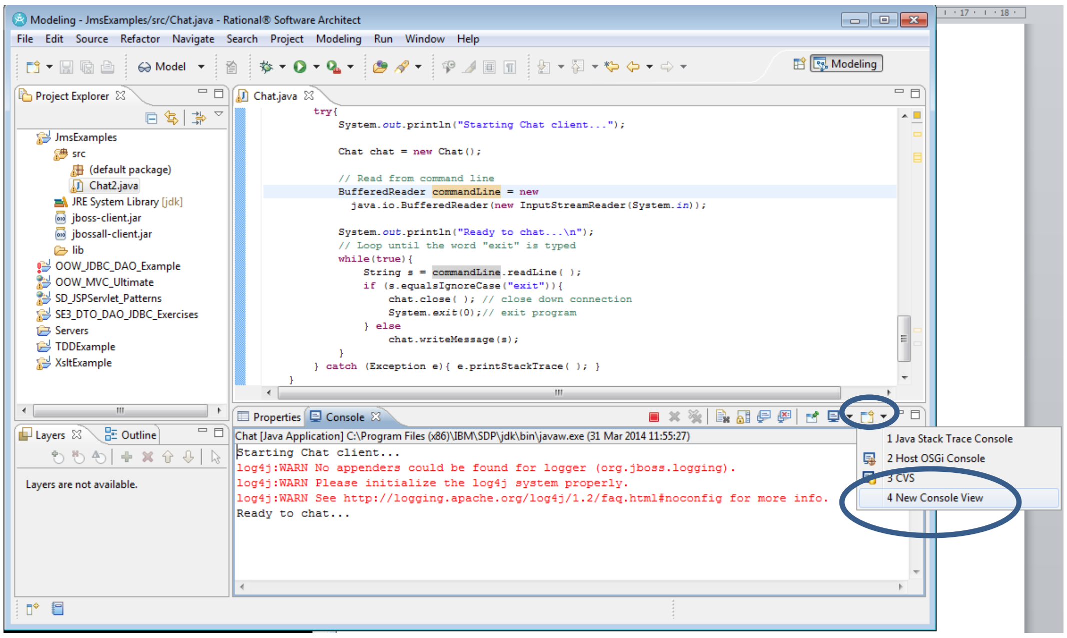
In this lab you can use the Java project you created and worked on in the last lab.

Demonstration of Publish & Subscribe Messaging using a Simplified Chat Application

- Using file explorer, copy the *Chat.java* source file from the Webcourses lab folder into your project *src* folder.
- Study the code to see how it uses the Publish / Subscribe mechanism to send messages between instances of the program. Note the *InputStreamReader*, when your code is running, you can enter text using the *Console View*.
- Ensure you have started the JBoss Application Server as in last week's lab.
- From eclipse, run the code twice (or three times). Ignore the warnings for now. Please see next note.

Note:

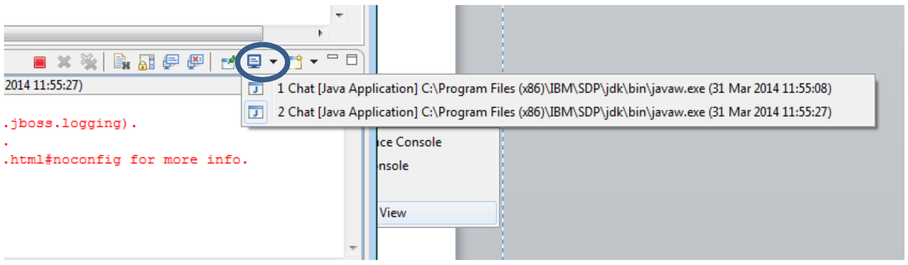
When you run two (or three) instances of the code, they will be running simultaneously and will keep running until the user types "exit". In eclipse, you will probably only see one *Console View*. You can open multiple *Console* views and also select which running process uses each *Console*. You can also *Detach* views and arrange them side by side if you want (see below). In this way you can test out the messages being sent and received by each instance of your code. Please ask if you need help with the *Console* views.



```
        try{
            System.out.println("Starting Chat client...");

            Chat chat = new Chat();

            // Read from command line
            BufferedReader commandLine = new
              java.io.BufferedReader(new InputStreamReader(System.in));

            System.out.println("Ready to chat...\n");
            // Loop until the word "exit" is typed
            while(true){
                String s = commandLine.readLine( );
                if (s.equalsIgnoreCase("exit")){
                    chat.close( ); // close down connection
                    System.exit(0);// exit program
                } else
                    chat.writeMessage(s);
            }
        } catch (Exception e){ e.printStackTrace( ); }
    }
```

```
Chat [Java Application] C:\Program Files (x86)\IBM\SDP\jdk\bin\javaw.exe (31 Mar 2014 11:55:27)
Starting Chat client...
log4j:WARN No appenders could be found for logger (org.jboss.logging).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Ready to chat...
```

Modeling - JmsExamples/src/Chat.java - Rational® Software Architect
File Edit Source Refactor Navigate Search Project Modeling Run Window Help
Project Explorer
JmsExamples
src
(default package)
Chat2.java
JRE System Library [jdk]
jboss-client.jar
jbossall-client.jar
lib
OOW_JDBC_DAO_Example
OOW_MVC_Ultimate
SD_JSPServlet_Patterns
SE3_DTO_DAO_JDBC_Exercises
Servers
TDDExample
XsltExample
Layers
Outline
Layers are not available.
Properties
Console
1 Java Stack Trace Console
2 Host OSGi Console
3 CVS
4 New Console View



1 Chat [Java Application] C:\Program Files (x86)\IBM\SDP\jdk\bin\javaw.exe (31 Mar 2014 11:55:08)
2 Chat [Java Application] C:\Program Files (x86)\IBM\SDP\jdk\bin\javaw.exe (31 Mar 2014 11:55:27)
2014 11:55:27)
.jboss.logging).
.html#noconfig for more info.

Exercise 1

- Test out the chat clients. Note that when a message is sent by one client, all clients receive that message (including the one that sent it).
- Modify the code so that the client that sent the message does not print it out on the console when it receives it.

Hint: Have a look at the methods on the JMS Message Interface specification (link below). In JMS, you can set properties into the message before sending them and read them from the message after receiving them. See if you can use some of the methods to achieve the above.

http://docs.oracle.com/javaee/6/api/javax/jms/Message.html

Exercise 2

- Modify the code so that the user could enter a name at the Console which would subsequently be used to prefix any messages sent by the client (e.g. *Ciaran> message text…*).

Suggestion: With the existing code, the user can enter “exit” to shut down the client. In a similar way, the user could enter “myname=Ciaran” to tell the client to prefix any messages with that name.